URBAN TRANSPORTATION NETWORKS

YOSEF SHEFFI
Massachusetts Institute of Technology

URBAN TRANSPORTATION NETWORKS:

Equilibrium Analysis with Mathematical Programming Methods

PRENTICE-HALL, INC., *Englewood Cliffs, New Jersey 07632*

Library of Congress Cataloging in Publication Data

Sheffi, Yosef. (date)
Urban transportation networks.

Bibliography: p.
Includes index.
1. Urban transportation—Mathematical models.
I. Title.
HE305.S54 1984 388.4'0724 84-9913
ISBN 0-13-939729-9

Editorial/production supervision:
Gretchen K. Chenenko, Ros Herion, and *Nancy DeWolfe*
Interior design: *Gretchen K. Chenenko*
Cover design: *Diane Saxe*
Manufacturing buyer: *Anthony Caruso*
Cover photo: *Ken Karp*

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-939729-9 01

Prentice-Hall International, Inc., *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Whitehall Books Limited, *Wellington, New Zealand*

To Anat

Contents

Preface

The flow pattern throughout an urban network can be looked upon as the result of two competing mechanisms. On the one hand, users of the system (drivers, passengers, pedestrians) try to travel in a way that minimizes the disutility associated with transportation. For example, motorists driving between a given origin and a given destination are likely to choose the route with the shortest travel time. On the other hand, the disutility associated with travel is not fixed but rather depends in part on the usage of the transportation system. Thus, in the previous example, the travel time on each of the paths connecting the origin and the destination is a function of the total traffic flow due to congestion. It is therefore not clear a priori which path through the network has the shortest travel time. Consequently, it may not be obvious what the flow pattern throughout the network will be under various conditions. This book describes how this flow pattern can be determined for an urban road network by modeling these two mechanisms (travel decisions and congestion).

The analytical approach described in this text draws on analogies between the two mechanisms mentioned here and the interaction of supply and demand in the marketplace. Instead of analyzing the price of a product and the quantity consumed, the analysis here looks at transportation level of service (or its inverse, travel disutility) and flows. The results of the analysis include a set of flow and a set of level-of-service measures that are at equilibrium with each other.

The book looks at many dimensions of travel choice, including the decision to take a trip, the choice of travel mode, the distribution of trips among various possible destinations, and the choice of route between an origin and a destination. All these decisions, when aggregated and analyzed simultaneously with the congested effects, result in the flow pattern through the network. The analysis of all these travel choices is carried out by using a unified framework that builds on graphical and network representation.

The problem of finding the equilibrium flow pattern over a given urban transportation network is also known as traffic assignment. The basic solution methodology is based on formulating the problem as a nonlinear optimization and solving it as such. This book, however, does not require any prerequisites in mathematical programming or graph theory. All the necessary background is reviewed at an introductory level. The level of mathematics assumed includes college calculus and (in the last parts of the book) introductory probability concepts. The book uses extensively intuitive arguments and network structures, which are utilized to illustrate many situations graphically.

This book grew out of a set of course notes used for two courses at M.I.T., one taken by graduate students and the other by undergraduates. Currently, the courses (and the notes) are combined, drawing both upper-level undergraduate and graduate students. The book was designed as a text for the following classroom environments:

1. A primary text for an intensive one-semester (or two-trimester) course(s).
2. A primary text for a slower one-semester or a one-trimester course (Chapters 1–6, 7, and/or 8, possibly 9, and parts of 13). Such a course would cover only deterministic equilibrium methods.
3. A supporting text in an introductory urban transportation planning course (Chapters 1, 3, 5, 6, and 7 with emphasis on the algorithms).
4. A supporting text in a course on operations research and mathematical programming techniques, demonstrating the applications of the methodology.

The book should also serve practicing transportation engineers and urban planners as a reference guide on issues of traffic assignment and urban transportation networks analysis.

Many people have contributed to the development of this book. Warren Powell of Princeton University and Carlos Daganzo of the University of California at Berkeley joined me in most of my research in this area. Joel Horowitz of the University of Iowa and Hani Mahmassani of the University of Texas at Austin helped in the preparation of the manuscript by commenting extensively on its early versions. Useful comments were also provided by Brian Brademeyer, Stella Dafermos, Mark Daskin, Patrick Harker, Haris Koutsopoulos, Larry LeBlanc, Joffre Swait, and all the students who took my networks course at M.I.T. during the years in which the manuscript was in preparation. The contributions of Karen and Jonathan are also greatly appreciated.

YOSEF ("YOSSI") SHEFFI

I

Urban Networks and Equilibrium

1

Urban Transportation Networks Analysis

The amount of travel taking place at a given moment on any street, intersection, or transit line in an urban area is the result of many individuals' decisions. Travelers choose if and when to take a trip for some purpose, which mode of transportation to use (e.g., whether to drive or take public transit), where to go, and which way to get there. These decisions depend, in part, on how congested the transportation system is and where the congested points are. Congestion at any point of the transportation system, however, depends on the amount of travel through that point. This book describes how these interactions between congestion and travel decisions can be modeled and solved simultaneously to obtain the flow pattern throughout the urban transportation network.

One of the major problems facing transportation engineers and urban planners is that of predicting the impact of given transportation scenarios. The analytical part of this problem can be dealt with in two main stages. First, the scenario is specified mathematically as a set of inputs that are used to predict the flow pattern resulting in from such a scenario. Second, the flow pattern is used to calculate an array of measures that characterize the scenario under study. The inputs to the analysis typically include a description of the following (either existing or projected):

1. The transportation infrastructure and services, including streets, intersections, and transit lines.
2. The transportation system operating and control policies
3. The demand for travel, including the activity and land-use patterns

The first stage of the analysis uses these inputs to calculate the flow through each component of the urban network. Flow is measured in terms of the number of travel units (vehicles, passengers, pedestrians) that traverse a given transportation facility in a unit time. This part of the analysis builds on functional relationships between flows and congestion as well as relationships between congestion and travel decisions.

The second stage of the analysis involves the calculation of an array of measures of interest, given the flow pattern. These may include the following:

1. Level of service measures, such as travel time and travel costs. These affect the users of the transportation system directly.
2. Operating characteristics, such as revenues and profits. These are of primary concern to the operators of the system.
3. Flow by-products, such as pollution and changes in land values. These affect the environment in which a transportation system is operated.
4. Welfare measures, such as accessibility and equity. These may indirectly affect various population groups.

The focus of this book is on the first stage of the analysis, that of determining the flow pattern, given the inputs. Many of the aforementioned measures can be calculated from these flows in a straightforward manner. Others (such as pollution level or equity measures) may require sophisticated methodologies and complex models which are not within the scope of this book. These analyses, however, use the flow pattern as a major input. In addition, a complete analysis of a given scenario may involve important considerations which are not based on the flow, such as construction costs and political and institutional factors.

The perspective of the modeling approach presented in this text is *descriptive* rather than *normative*. In other words, it describes how individuals travel through an urban transportation system given the components of that system (which are specified as inputs). It does not attempt to determine the optimal system configuration (in terms, say, of which project should be built or what kind of control policies should be utilized). In many cases, however, the analysis is motivated by the need to decide on a given transportation investment, regulation, or policy from among a set of alternative courses of action. In this case, each of the alternative scenarios should be specified mathematically as a set of inputs to the analysis. The models described in this book are then formulated and solved for each alternative scenario in order to predict the flow pattern that is used, in turn, to develop many of the measures of interest. The full set of measures associated with each alternative is then used to compare their respective impacts. These measures are, typically, also compared to the set of measures characterizing a base case (which can be the current system or a projection of how the current system will operate under the "do nothing" alternative). These comparisons serve as a basis for recommendation regarding the preferred course of action.

This book presents the various assumptions, models, and algorithms used to calculate the flow pattern from the inputs. The thrust of the book is the interaction of congestion and travel decisions that result in this flow. Since congestion increases with flow and trips are discouraged by congestion, this interaction can be modeled as a process of reaching an *equilibrium* between congestion and travel decisions.

The remainder of this chapter introduces and explains the notion of equilibrium in the context of urban transportation analysis. Section 1.1 presents this equilibrium in general terms, and contrasts it with the more familiar economic equilibrium framework. Section 1.2 describes the mathematical representation of the urban transportation system as a network. The network structure is a fundamental characteristic of the equilibrium discussed in this text. Section 1.3 gives the actual definition of several equilibria between congestion and travel decisions. These equilibria are defined specifically for urban transportation networks. The section ends with an outline of the text. Finally, Section 1.4 summarizes the first chapter and Section 1.5 suggests some further readings.

1.1 EQUILIBRIUM ANALYSIS OF TRANSPORTATION SYSTEMS

The concept of equilibrium analysis is tied to the systems-oriented view of urban transportation taken in this text. This section discusses three related issues that explain this view: (1) the need for the systems-based approach to the analysis of urban transportation, (2) the general notion of equilibrium and the various types of equilibria, and (3) the applications of these notions to transportation systems analysis.

Systems Approach to Urban Transportation

The traditional approach taken in many engineering analyses is to isolate a component of a system and analyze this component individually. This approach is also used in analyzing the effect of small changes in the urban transportation system. For example, traffic lights are typically timed by considering only the traffic using the intersection under consideration. The effects of the change on adjacent intersections are usually assumed to be negligible. Similarly, parking regulations, intersection designs, and other changes in the urban transportation system are typically analyzed by considering only the immediate environment.

If the impact of a particular policy or design is likely to be small, this type of analysis may be sufficient. When the change being analyzed is more substantial, however, it will affect not only the component that is being changed but other parts of the system as well. As an illustration of this type of ripple effect, consider a congested segment of an urban arterial. In order to reduce congestion, the local transportation authority considers the costs and benefits of widening this segment of road. Given the existing traffic flow, the addition of one lane is judged to be sufficient, since the added lane will mean that the existing traffic will be able to flow smoothly and with only minor delays. The project benefits calculated in this manner may, however, be misleading since travel decisions were not taken into account. For example, it is likely that motorists who did not use the arterial segment before, will subsequently make use of the improved facility. Thus the traffic conditions on the widened facility will not be as good as anticipated due to the increased flow, which will cause increased congestion. In turn, some of the roads leading into the improved facility may get congested as motorists try to access and exit from the widened segment. In addition, the flow conditions on roads parallel to the widened facility may improve as a result of these shifts since the traffic flow on them may decrease. Drivers on other parts of the system may then realize a change in their flow conditions and alter their route accordingly. Each route change will lead to further changes in congestion levels and consequently more route changes. Ultimately, however, these ripple effects will lessen and, after a short while (several days or a few weeks), the system will stabilize at a new *equilibrium* point with no more significant changes occurring.

The preceding example illustrated the flow changes resulting from a change to the transportation infrastructure. Similar changes may occur if transportation control policies are changed or if new transporation services are introduced. Furthermore, new flow patterns may result from changes not directly tied to the transportation system but rather changes in the general activities in the urban area. These activities create the need and the demand for transportation. For example, consider the opening of a new shopping center. The center attracts shopping trips which are partially new (generated by people who now shop more frequently) and partially diverted from other destinations. The streets leading to the new center will get congested, causing some travelers who are only passing through the area to change route, mode of travel, or the timing of their trips. These diversions will change the flow and congestion throughout the system, causing further changes in travel decisions. Other changes will occur around other shopping centers, which may experience some decline in the level of flow. Again, after a while the flows will reach a new equilibrium point. At this point, the frequency of trips, the trip destinations, the modes of travel, and the chosen routes are stable throughout the transportation network.

The notion of equilibrium in this context parallels the physical notion of stable equilibrium, which is the state in which there are no (net) forces that try to push the system to some other state. Furthermore, when the system is in disequilibrium, there are forces that tend to direct the system toward the equilibrium state. In the case under consideration, the flows are being "pushed" toward equilibrium by the route-switching mechanism. At equilibrium, the flows will be such that there is no incentive for route switching. It is this state of equilibrium that is the focus of this text.

As the examples above indicate, the equilibrium flow pattern associated with a given scenario may involve unanticipated flow levels and congestion in various parts of the urban network. This bears directly on the question of who is affected under this scenario. Whereas some travelers will be better off, others may be worse off. A consideration of the entire system is therefore neccessary in order to account for the wide spectrum of effects associated with a particular scenario. In other words, the equilibrium flow pattern that would prevail under each scenario can be found only by analyzing simultaneously all elements of the urban transportation network.

The next section explains the equilibrium concept in several contexts, leading to the role of such analysis in the study of urban transportation systems.

Equilibrium in Markets

The classical view of a (perfectly competitive) economic market for a certain good includes two interacting groups: the producers and the consumers. The behavior of the producers is characterized by a supply function and the behavior of the consumers is characterized by a demand function. The supply function expresses the amount of goods that the suppliers produce as a function of the price of the product. As the price increases, it becomes profitable to produce more and the quantity supplied increases. The demand function describes the aggregate behavior of consumers by relating the amount of the product consumed to its price. As the price increases, the amount consumed decreases.

Figure 1.1 depicts simple demand and supply functions for a certain product. The point where the two curves intersect is characterized by the "market clearing" price, P^*, and quantity produced, Q^*. This is the point where the price of the product is just right, so that the entire quantity produced is consumed. If the price is higher than P^*, production will be higher than consumption, as shown in Figure 1.1b. Such an imbalance cannot be sustained, since not all of the product sells and inventories will grow indefinitely. Prices will eventually fall and consumption will increase accordingly. If the price is lower than P^*, the quantity demanded is higher than the production. Such a situation is again unstable since producers will try to increase prices in order to capture the consumers' willingness to pay more. Such price increases will lead to higher production and lower demand. Thus, if the prices are either lower or higher than P^*, market forces will tend to push the price toward its "market clearing" level. At this point the price will be stable and thus the point (P^*, Q^*) is known as the *equilibrium* point.

The monetary price of a product is not always the only determinant of the quantity consumed. Products can be characterized by many attributes that influence consumption. Furthermore, some of these characteristics are not constant but rather a function of the quantity consumed.

Consider, for example, the demand for gasoline at a certain gas station.

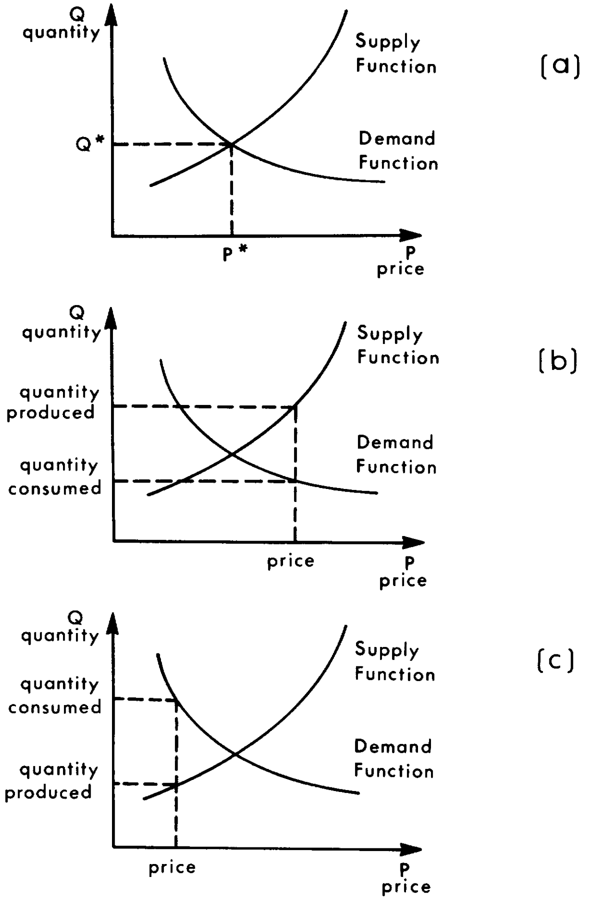


Figure 1.1 Demand/supply equilibrium: (a) market-clearing quantity and price; (b) price too high; (c) price too low.

The lower the price of gasoline at this station, the larger the number of motorists who would want to purchase gasoline there. As the number of customers grows, a queue appears and customers have to wait in line. At this point, the number of additional customers is influenced by two elements: price and waiting time. The station operator can set only the price. For a given price, the volume of sales will be determined by two factors: the waiting time, which is influenced by the number of customers in the queue, and the willingness of customers to wait, which determines the demand for gasoline at the station.

For a given gasoline price,† then, the situation can be characterized by two functions. The first is a demand function that relates the customers' arrival

†Assuming also that the price and waiting time in all other stations in the area is fixed and known.

rate to the delay (in other words, it gives the number of customers entering the station per unit time as a function of the waiting time). The second function relates the waiting time to the arrival rate of customers. This second function, unlike a supply function, does not capture the behavior of any economic agent. Instead, it depicts merely a physical phenomenon, which in this case is the delay associated with queueing. This function is termed a *performance function.* Unlike the supply and demand functions, the performance function does not have the price or any other service characteristic as its argument, and it does not give the quantity consumed or produced. Instead, its argument is the quantity consumed (or the arrival rate in the case of the example considered here) and it gives the service characteristic (which is the delay in this case).

Figure 1.2a depicts a hypothetical performance curve showing how the

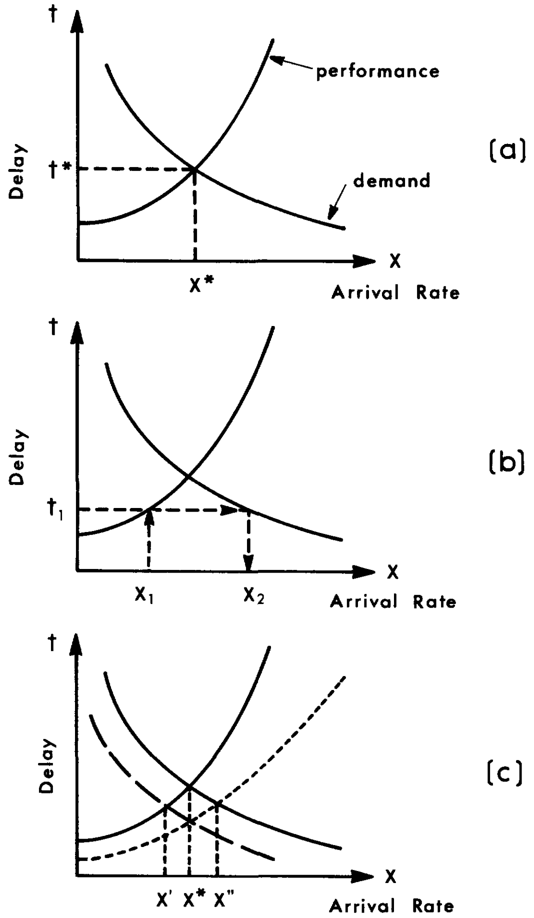


Figure 1.2 Demand/performance equilibrium for the gas station example: (a) equilibrium arrival rate and waiting time; (b) a waiting time that is too low; (c) a shifted demand curve (higher price) and a shifted performance curve (higher service rate).

delay, t, increases with the arrival rate, x. The figure also depicts a demand curve which shows the arrival rate associated with any given delay (this curve should be read by entering the vertical axis). When the arrival rate is low, there will be little queueing and the delay will be low, attracting more customers. When the arrival rate is very high, the delays are very high and no customers will join the queue. The only stable situation will be when the arrival rate is such that it creates a delay that corresponds to the same arrival rate. This equilibrium point is denoted by (x^*, t^*) in Figure 1.2a. Figure 1.2b shows a situation in which the arrival rate, x_1, is too low (i.e., $x_1 < x^*$). The delay associated with this arrival rate is t_1, in accordance with the performance function. This delay, however, generates an arrival rate of x_2 (where $x_2 > x^*$), in accordance with the demand function. As the arrival rate increases, the delay increases toward the equilibrium point where the delay matches the arrival rate.

Note that the demand function given in Figure 1.2 holds only when everything else (price, number of pumps, delay and price at other stations, etc.) is fixed. Since the demand is a function of both price and delay, a change in the price would show as a different demand function in the arrival rate/delay space. The demand curve associated with a higher price is depicted by the dashed line in Figure 1.2c. The equilibrium quantity here, x', will be lower than x^*, as expected. The dotted line in Figure 1.2c depicts a performance function associated with more pumps operating. In this case the delay associated with each arrival rate is smaller and the equilibrium quantity, x'', will be higher than x^*, again as expected.

The economics of the gas station operation can still be analyzed in the framework of supply/demand equilibrium. In this case the demand function gives the quantity consumed (or the customers' arrival rate) as a function of both price and delay. A supply function will give the price charged by the operator as a function of the quantity consumed. This supply/demand equilibrium, however, cannot be solved in isolation since the delay depends on the arrival rate. The latter relationships are given by a performance function. A complete economic analysis utilizes all three functions to solve, simultaneously, for the arrival rate, the delay, and the price. The demand/performance analysis given in the preceding paragraphs assumed that the price is fixed and known.

This example should clarify the differences between a supply/demand and performance/demand equilibrium. In a sense, the performance/demand analysis is a special case of the supply/demand equilibrium. It is obtained by fixing some of the variables in the supply/demand equilibrium problem.

The Transportation Arena

Transportation is a service that the transportation industry (including road builders, vehicle manufacturers, transit operators, traffic managers, etc.) offers travelers. As in many other service industries, the transportation products can be characterized by the level of service offered in addition to the price charged. Transportation level of service can be measured in terms of travel time, schedule convenience, reliability, safety, comfort, spatial coverage, accessibility to the service, and many other factors.

Many of the components of transportation level of service are not fixed but rather are flow dependent, in that their value depends on the usage of the transportation system. Consider, for example, a transit line connecting two points in an urban area. If many individuals choose to use this line, the buses will get congested, the waiting time will increase, the probability of obtaining a seat will be lower, and the driver is less likely to be friendly. The level of service, then, will be lower with increased patronage even if the fare remains fixed. In response, some patrons may drive instead of taking transit, others may change the timing of the trips, and yet others may forgo the trip altogether. This situation can be analyzed by using two functions: (1) a performance function that describes how the level of service deteriorates with increasing volume; and (2) a demand function that describes how the volume of passengers increases with improved level of service. These functions are defined for a given situation in terms of fare structure, schedule, equipment, and so on. As was the case with the aforementioned gas station example, the framework for a performance/demand equilibrium analysis is set by fixing those service characteristics that are under the operator's control. In the case of the gas station, this includes only the price. In the case of a transit line, these characteristics include a range of variables.

The dependence of the level of service on the flow is a fundamental characteristic of the transportation market. In the context of urban transportation, performance functions are rooted in the congestion phenomenon, which causes increased delays and costs with increased flow. The focus of this book is on the calculation of the demand/performance equilibrium in the urban passenger transportation market. The notion of equilibrium in the remainder of this book refers solely to this type of equilibrium.

The equilibrium in the urban transportation market is necessarily reached over the urban network of streets and transit lines. To define the notion of equilibrium over a transportation network, the network concept has to be presented and discussed. That is the topic of the next section.

1.2 NETWORK REPRESENTATION

The term "network" is commonly used to describe a structure that can be either physical (e.g., streets and intersections or telephone lines and exchanges, etc.) or conceptual (e.g., information lines and people, affiliation relationships and television stations, etc.). Each of these networks includes two types of elements: a set of points and a set of line segments connecting these points. This observation leads to the mathematical definition of a network as a set of *nodes* (or *vertices* or *points*) and a set of *links* (or *arcs* or *edges*) connecting

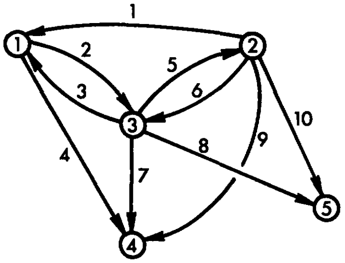


Figure 1.3 Directed network of links and nodes.

these nodes.† Figure 1.3 depicts a network including five nodes connected by 10 links. Each link in this network is associated with a direction of flow. For example, link 5 represents flow from node 3 to node 2, while link 6 represents the reverse flow, 2→3. This text deals exclusively with *directed* networks (in which all links are directed).

Each network link is typically associated with some impedance that affects the flow using it.‡ The units of measurement of this impedance depend on the nature of the network and the link flows. Impedance can represent electrical resistance, time, costs, utility, or any other relevant measure. When the flow involves people, the term "level of service" is usually used instead of "impedance." (The magnitudes of these terms are the opposite of each other; that is, a high level of service means low impedance.) Only links can be associated with impedance. Nodes represent merely the intersection of links and are not associated with any impedance to flow.

The networks discussed in this text are typically "connected." In other words, it is possible to get from any node to any other node by following a *path* (or a *route*) through the network. A path is a sequence of directed links leading from one node to another. The impedance along a path is the sum of the impedances along the links comprising that path. Note that a pair of nodes is usually connected by more than one path. For example, to get from node 2 in Figure 1.3 to node 4, the following paths (designated by the link numbers along the path) can be used: 9; 6, 7; 6, 3, 4; 1, 2, 7; 1, 4. In large networks, the number of paths connecting each pair of nodes can be extremely large.

Representation of the Urban Road Network

The focus of this text is on urban transportation. The roadway network in an urban area includes intersections and streets through which traffic moves. These elements can be translated naturally into a structure of nodes and links (including impedance measures), respectively. For a measure to

†The term "network" is used, then, to represent both a physical structure and its mathematical representation (known also as a graph).

‡Formally, the difference between a network and a graph is that network links are associated with impedance, whereas the links of a graph are not (they only represent connection and/or direction).

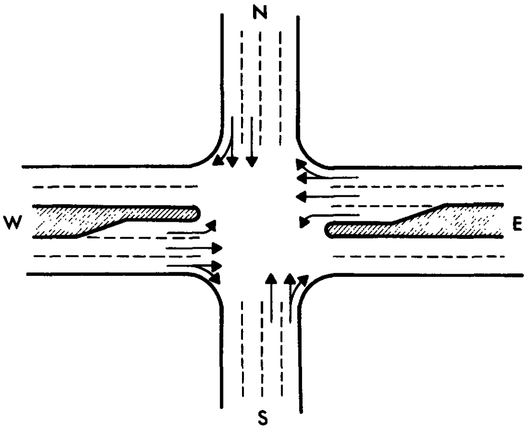


Figure 1.4 Four-approaches intersection layout.

properly represent link impedance it has to be a deterrent to flow. A particularly relevant measure for links representing urban streets is the time required to traverse the street. This impedance measure is discussed in detail in the last part of this section.

The graph representation of a physical network is not unique. In other words, there are many networks that can be used to represent the same physical structure. Consider, for example, the intersection shown in Figure 1.4. This intersection can be represented simply as a node with the streets leading to and from it represented by links connected to that node, as shown in Figure 1.5a. Note that a two-way street is represented by two opposite-direction links. The impedance on the links entering the intersection (the approaches) represents the intersection delay as well as the travel time along the approaching street. The impedance on the outbound links represents the travel time on the streets in that direction and the delay at downstream intersections.

The drawbacks of this representation are twofold. First, it cannot be

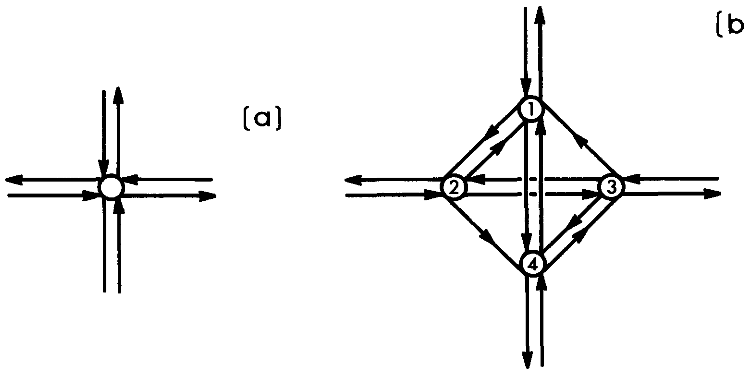


Figure 1.5 Network representation of the intersection in Figure 1.4: (a) representing the intersection as a node; (b) a detailed intersection representation.

used to represent turning restrictions. Second, it assumes that all flow entering the intersection from a particular direction will experience the same travel time, regardless of where it is destined. Clearly, right turns are easier to execute than straight-through movements, which, in turn, are easier to execute than left turns. These different delays can be accounted for by a more elaborate representation of the same intersection, shown in Figure 1.5b. In this figure the single node representation of the intersection is replaced by a network of four nodes and 10 links. Each permissible movement through the intersection is represented by a separate link. (Note that the graph is not completely symmetric due to the lack of left-turning movements from the northern and southern intersection approaches.) In this representation, each intersection movement can be associated with the appropriate delay. Link 3 → 4, for example, will be associated with left-turning delay that would be experienced by the traffic turning from the eastern approach into the southern one. Link 3 → 1, on the other hand, will represent right-turning maneuvers for flow from the east to the north, with the appropriate delay. In such a representation the links leading into the intersection will be associated only with the travel time on the incoming approaches, not with the intersection delay itself. The intersection delay is captured by the links connecting the four nodes in the figure.

Representation of the Transit Network

The movement of vehicular traffic through streets and intersections is not the only flow in the urban area; this text looks at the flow of transit passengers as well. A transit link can be represented by a simple linear network in which the transit stations (or bus stops) are represented by nodes and the line-haul portion by links. Such a graph is shown in Figure 1.6a. The impedance on each of the links in this network includes "in-vehicle" travel disutility elements such as travel time. Other measures that are not associated with the travel in the transit vehicle, however, serve also as travel impedance. These include, for example, the waiting time at the station and the fare charged. Furthermore, the transit station may be located away from the actual destination, requiring a walking trip from the point of alighting to the destination node. Figure 1.6b depicts the transit network associated with the flow between a particular origin node to a particular destination node. The loading link, in this figure, corresponds to the waiting time and fare, while the unloading link corresponds to the walking time. Note that in order for the various types of links to be compatible with each other, the impedance on all of the network links has to be expressed in the same units. If this were not the case, the impedance of a path could not be calculated.

A representation of a complete transit network would include not only loading and unloading links but also transfer links. The impedance on such links would be associated with line transfer charges (which may be lower than the fare at the start of the trip) and the waiting time at the transfer point. Figure 1.6c depicts a network representation of a transit station serving two

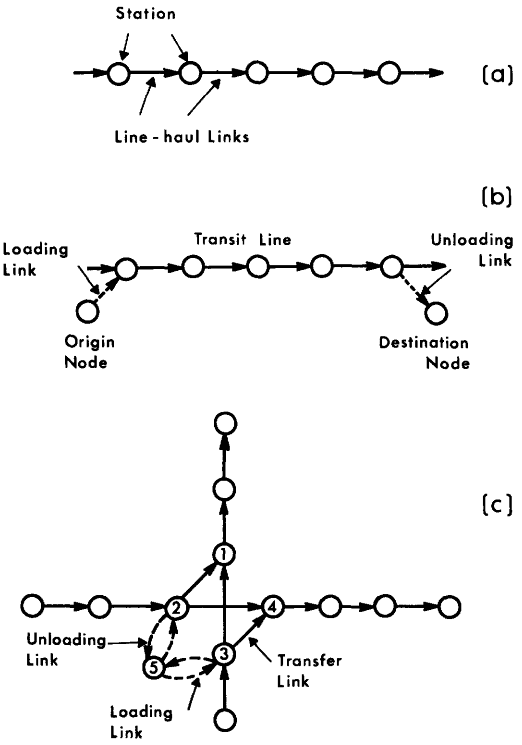


Figure 1.6 Transit-line representation: (a) a representation including only the "in-vehicle" movement; (b) a representation including boarding and alighting movements; (c) a detailed representation of a transit station, including transfer, boarding, and alighting links.

lines (west → east and south → north). The station itself is represented by the four nodes 1, 2, 3, and 4. Links 3 → 1 and 2 → 4 represent the continuation of a trip on the same line and are therefore associated with zero impedance. (Note that all four nodes represent the same physical facility.) Links 3 → 4 and 2 → 1 represent the transfer movements between the two lines. The impedance on these links include the transfer costs and delays expressed in appropriate units. Node 5 represents both a source for trips (i.e., it is an origin node) and a terminus for trips (i.e., it is also a destination node). This node is connected to both transit lines by loading and unloading links (shown as dashed lines in Figure 1.6c). It represents a set of trip origins and trip destinations which are located in the vicinity of the transfer station.

Centroids and Connectors

The transportation planning process for urban areas is typically based on a partition of the area into traffic zones. The size of each traffic zone can vary from a city block to a whole neighborhood or a town within the urban area. The number of traffic zones can vary from several dozens to several thousands. Each traffic zone is represented by a node known as *centroid* (the name stems from the practice of placing the node at the geometrical center of gravity of the traffic zone). The network representation of the urban area will include many other nodes, representing intersections, bus stops, and other transportation facilities. The centroids, however, are those "source" and "sink" nodes where traffic originates and to which traffic is destined. (Node 5 in Figure 1.6c is a centroid.) Once the set of centroids is defined, the desired movements over an urban network can be expressed in terms of an *origin–destination matrix.* This matrix specifies the flow between every origin centroid and every destination centroid in the network.

Figure 1.7 shows a traffic zone that is surrounded by four two-way streets. The node in the middle of the zone is the centroid. It is connected to the roadway network by special *centroid connector* links (known also as "connectors" or "dummy links"). The centroid shown in the figure is a trip origin and therefore all the connectors are directed away from the centroid and into the nodes of the road network. Connectors directed toward the centroid would be added if trips can also terminate there.

Each centroid represents an aggregation of all the actual origins and all the actual destinations in its traffic zone. The centroid connectors represent the ubiquitous street network within a traffic zone. If the actual origin and destination points are uniformly distributed in the traffic zone, the connectors can be associated with the travel time between the geometrical centroid of each zone and the appropriate nodes. If the distribution of origin and destination points is not uniform, the connector travel times should be weighed accordingly.

The centroids can be connected directly to nodes that are part of the transit network. In this case, the connectors represent loading and unloading links, as shown in Figure 1.6c. The impedance on the loading centroid connector represents the walking time to the station, waiting at the station, and the transit fare. The impedance on the unloading connectors would represent the walking time from the transit station to the centroids.

The level of detail at which an urban area is represented is determined in large part by the size of the traffic zones. The analysis of the flows in the urban area does not focus on the flows within each traffic zone; the centroid and centroid connectors representation has to ensure only that the flows *between* the traffic zones are correct. If the flows within a certain zone are of interest to the transportation analyst, the zone under consideration should be divided

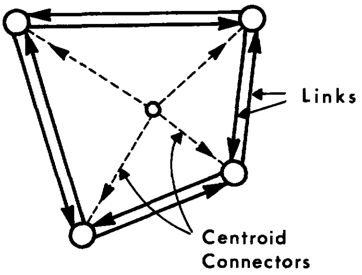


Figure 1.7 Network representation of a traffic zone, including a centroid node and centroid connector links.

into smaller zones and the road network between these zones modeled explicitly. Such detailed analysis is likely to be more accurate. Costs, however (including both direct analysis costs and data collection costs), escalate dramatically as the number of traffic zones and the size of the network increase. The size of the network to be analyzed is typically determined by this trade-off between the required accuracy and the available budget.

Similar considerations apply to the level of detail in which the road network itself is represented. Although it is important to show all the major streets and arterials, smaller streets may not have to be represented at all. The issue of network representation is as much an art as it is a science; practice and experience are required to carry it out successfully.

Link Performance Functions

The travel impedance, or level of service, associated with the links representing an urban network can include many components, reflecting travel time, safety, cost of travel, stability of flow, and others. The primary component of this impedance, however, is travel time, which is often used as the sole measure of link impedance. The reasons for using travel time in this context are threefold. First, empirical studies seem to indicate that it is a primary deterrent for flow. Second, almost all other possible measures of travel impedance are highly correlated with travel time and thus exhibit the same trends. Third, it is easier to measure than many of the other possible impedance components. Generalized impedance, which combines several measures, can, however, be used and in the remainder of this book, the term "travel time" can be understood as such a combined impedance. Furthermore, impedance measures other than travel times can be used explicitly in some parts of an urban network. For example, the appropriate impedance measure for links in the transit network are transit (in vehicle) time, waiting time, fare, and so on. To be compatible with the impedance over the vehicular traffic-carrying links of an urban network, these impedance measures should all be expressed in travel-time units.†

As mentioned in Section 1.1, the level of service (or impedance) offered by many transportation systems is a function of the usage of these systems. Because of congestion, travel time on urban streets and intersections is an increasing function of flow. Consequently, a *performance function* rather than a constant travel-time measure should be associated with each of the links representing the urban network. The performance function relates the travel time (which, as mentioned above, may stand for generalized impedance) on each link to the flow traversing this link.

A performance function for a typical approach to a signalized intersection is shown in Figure 1.8. This function captures both the time spent in

†The conversion coefficient should be rooted in travelers' values. The estimation of such coefficients is discussed in Chapter 13.

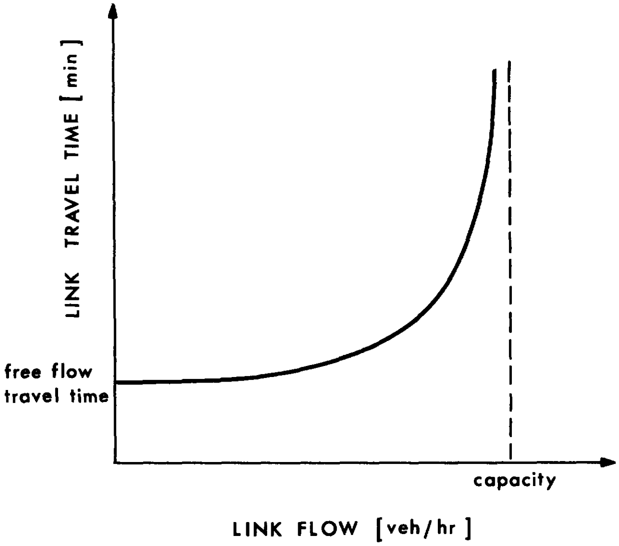


Figure 1.8 Typical link performance function for an approach to a signalized intersection.

traveling along the approach under consideration and the delay at the downstream intersection. The shape of the curve shown in the figure is typical of performance functions for links representing urban network components.

The travel time at zero flow is known as the free-flow travel time. At this point, a traveling car would not be delayed because of interaction with any other car moving along the link. The only source of delay at this point is the time associated with traversing the link and the expected delay associated with the probability of being stopped by a red signal indication. As the flow increases, the travel time monotonically increases since both the travel time along the approach increases (because of vehicle interactions at higher traffic densities) and the intersection delay increases (because of queueing phenomena) with the flow. Characteristically, the performance function is asymptotic to a certain level of flow known as the *capacity* of the transportation facility under consideration. The capacity is the maximum flow that can go through any transportation facility. The performance function is undefined for higher values of flow, since such flows cannot be observed. When the flow approaches capacity, the queues at the intersection will start growing, clogging upstream intersections and finally causing traffic to come to a halt.

The general shape of the performance functions is similar for links representing most types of urban streets. The physical characteristics of each street (e.g., length, width, parking restrictions, turning pockets, or signal green time) determine the exact parameters of the function for each street.

The centroid connectors are typically modeled as fixed travel-time links. In other words, the travel time on the connectors does not vary with the flow since these links represent a ubiquitous local street network and not a particular facility. Similarly, loading and unloading links represent time and cost components that do not vary with the flow.

It is important to remember that the physical network and its graph representation do not have to be similar. As shown in later chapters, links can be used to represent many types of flow, some of which do not correspond to physical movements. For example, in Chapter 6 the individuals who choose not to travel at all are represented as a flow on an imaginary link added to the network. It is always true, however, that each link represents a distinct flow.

The next section discusses equilibrium over urban transportation networks, building on the system analytic approach and the equilibrium framework introduced in Section 1.1.

1.3 EQUILIBRIUM OVER URBAN TRANSPORTATION NETWORKS

The notion of equilibrium in the analysis of urban transportation networks stems from the dependence of the link travel times on the link flows. Assume that the number of motorists who wish to travel between a given origin point and a given destination point is known. Furthermore, assume that these points are connected by several possible paths. The question of interest here is how these motorists will be distributed among the possible paths. If all of them were to take the same path (which may initially be the shortest one in terms of travel time), congestion would develop on it. As a result, the travel time on this path might increase to a point where it is no longer the minimum travel-time path. Some of these motorists would then use an alternative path. The alternative path can, however, also be congested, and so on.

The determination of the flows on each of these paths involves a solution of a demand/performance equilibrium problem. The flow on each link is the sum of the flows on many paths between many origins and destinations. A performance function is defined independently for each link, relating its travel time to this flow. The demand for travel, however, is rooted in motorists' behavior and is not defined for each link separately. Instead, it specifies how motorists choose among the alternative paths (routes) connecting each origin–destination (O–D) pair. This dichotomy in the definition of the performance and the demand functions gives this performance/demand equilibrium analysis its "systems" nature. In other words, this is why no link, path, or origin–destination pair can be analyzed in isolation.

This section discusses the route choice rule and the associated definitions of equilibrium used in this text.

Definitions of Equilibria

This text analyzes a variety of urban network equilibrium problems, including both the transit and automobile mode and encompassing several possible travel decisions. The basic problem, however, is presented here for a simplified case, including a network representing only the automobile flows. The only travel choice examined here is the motorists' choice of routes between their origin and their destination. This problem can be put as follows:

Given:

1. A graph representation of the urban transportation network
2. The associated link performance functions
3. An origin–destination matrix

Find the flow (and travel time) on each of the network links.

This problem is known as that of *traffic assignment* since the issue is how to assign the O–D matrix onto the network. The resulting link flows are then used to calculate an array of measures, which can be used, in turn, to evaluate the network. Particular designs of transportation infrastructure or transportation control policies usually enter the analysis through the specification of the network itself and the performance functions.

To solve the traffic assignment problem, it is required that the rule by which motorists choose a route be specified. As explained above, this rule can be viewed as the function or the procedure that specifies the demand for travel over paths. The interaction between the routes chosen between all O–D pairs, on the one hand, and the performance functions on all the network links, on the other, determines the equilibrium flows and corresponding travel times throughout the network.

It is reasonable to assume that every motorist will try to minimize his or her own travel time when traveling from origin to destination. As explained above, this does not mean that all travelers between each origin and destination pair should be assigned to a single path. The travel time on each link changes with the flow and therefore, the travel time on several of the network paths changes as the link flows change. A stable condition is reached only when *no traveler can improve his travel time by unilaterally changing routes.* This is the characterization of the *user-equilibrium* (UE) condition.

Since individual motorists can be expected to behave independently, the UE situation ensures that at this point there is no force that tends to move the flows out of the equilibrium situation. Consequently, this point will be stable and, in fact, a true equilibrium.

The user-equilibrium condition, however, is not the only possible definition of equilibrium. The assumption that each motorist chooses the minimum-travel-time route may be reasonable in some cases, but it includes several presumptions that cannot always be assumed to hold even approximately. For example, the UE definition implies that motorists have full information (i.e., they know the travel time on every possible route) and that they consistently make the correct decisions regarding route choice. Furthermore, it assumes that all individuals are identical in their behavior. These presumptions can be relaxed by making a distinction between the travel time that individuals *perceive* and the actual travel time. The perceived travel time can then be looked upon as a random variable distributed across the population of drivers. In other words, each motorist may perceive a different travel time, over the same link. Equilibrium will be reached when *no traveler* believes *that his travel time can be improved by unilaterally changing routes.* This definition characterizes the *stochastic-user-equilibrium* (SUE) condition.

The stochastic user equilibrium is a generalization of the user-equilibrium definition. If the perceived travel times are assumed to be entirely accurate, all motorists would perceive the same travel time and the stochastic user equilibrium will be identical to the (deterministic) user equilibrium. In other words, the flow patterns resulting from both models would be identical.

The definitions of equilibria given above are not easy to use in an operational manner to solve for equilibrium flow patterns. To be useful, the equilibrium definitions have to be characterized and formulated mathematically. The following section discusses the user equilibrium, while the more general stochastic user equilibrium is treated only in Part IV.

An important point, however, should be mentioned here with regard to both types of equilibria. The demand for urban travel is derived from the pattern of activity in the urban area. The timing and locations of these activities mean that travel demand is not uniform throughout the day. Steady-state equilibrium analyses of the type discussed here, however, are applicable only if the flows can be considered stable over the period of analysis. Consequently, transportation planners analyze urban transportation systems for certain time periods such as the "morning peak," "evening peak," or "midday," depending on the purpose of the analysis. The origin–destination flows within each of these respective periods is considered constant in order for steady-state analyses to apply. The larger the period of analysis, the less accurate this assumption is. The period of analysis cannot be very small, however, since each such period has to be appreciably longer than the typical duration of trips at this time.

A Simple User-Equilibrium Example

Consider the two-link network shown in Figure 1.9a. This network represents one origin–destination pair connected by two alternative routes. Let t_1 and t_2 represent the travel time on links 1 and 2, respectively, and let x_1 and x_2 represent the traffic flow on these links. The total origin–destination flow is designated by q, where

$$q = x_1 + x_2 \tag{1.1}$$

The performance functions on these links $t_1(x_1)$ and $t_2(x_2)$ are shown in Figure 1.9b. For each link, the performance function gives the travel time on that link as a function of the flow on the link.

Assume now that the trip rate between O and D is very small. In other

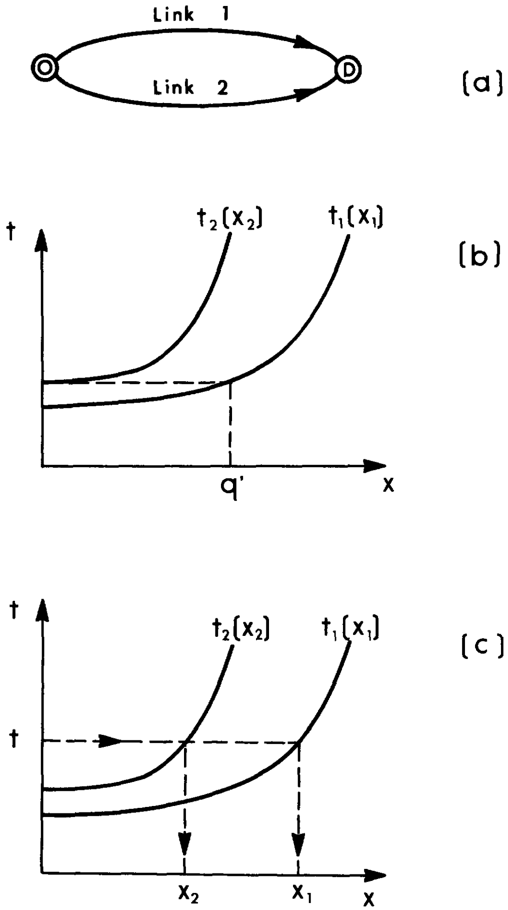


Figure 1.9 Equilibrium in a simple network: (a) a two-link network; (b) the two performance functions—if the flow is less than $q' = x_1 + x_2$, it is assigned to link 1; (c) the travel time on both links is equal if the flow is higher than q'.

words, q is very small. If all motorists are trying to minimize their own travel time, each of the q motorists should choose to travel over link 1. As shown in the figure, this link is associated with a lower free-flow travel time than link 2. If q is small, the increased delay due to the traffic on link 1 is not sufficient to increase the travel time on this link even to the point where it is equal to the free-flow travel time on link 2. Thus, all q motorists will use link 1 and no one will use link 2. This is an equilibrium situation since none of the motorists using link 1 has an incentive to switch routes to the longer link. Such an equilibrium will hold as long as $q < q'$, where q' is the flow that causes the travel time on link 1 to equal the free-flow travel time on link 2. At this point, an additional motorist may choose either link. If the additional motorist chooses link 2, the travel time on it will increase and the next motorist will choose link 1. If on the other hand, the first motorist (above q') chooses link 1, the next one will choose link 2.

Looking at the flow of traffic as a continuous flow, it is clear that beyond the point $q = q'$, equilibrium can be maintained only if the travel time on both links is equal. Beyond this point both links are used, and if the travel times are not equal, some motorists can change route and lower their own travel time. The route-switching process will not occur only if the travel time on both routes is equal, giving motorists no incentive to switch.

The two characterizations of equilibrium that can occur in this two-link example (the first for $q < q'$ and the second for $q > q'$) give rise to an operational definition of user equilibrium over transportation networks:

UE Definition. For each O–D pair, at user equilibrium, the travel time on all used paths is equal, and (also) less than or equal to the travel time that would be experienced by a single vehicle on any unused path.

This definition means that at equilibrium, the paths connecting each O–D pair can be divided into two groups. The first group includes paths that carry flow. The travel time on all these paths will be the same. The other group includes paths that do not carry any flow. The travel time on each of these paths will be at least as large as the travel time on the paths in the first group.

Using this definition, the two-link example in Figure 1.9 can now be solved for any value of q. The assignment of any amount of flow that is less than q' is obvious—it should all be assigned to link 1. The only problem is to ensure that the traffic assignment is such that beyond the point of $q = q'$, both links are assigned at a rate that keeps the travel time on these links equal. If the equilibrium travel time, t, between the origin and the destination is known (for the case in which $q > q'$), the equilibrium definition above means that $t = t_1 = t_2$. Consequently, the appropriate link flows can be determined by the inverse link performance functions [i.e., $x_1 = t_1^{-1}(t)$ and $x_2 = t_2^{-1}(t)$]. Graphically, this determination can be accomplished by entering the link performance curves horizontally, given t, as shown in Figure 1.9c. This method is applicable at any value of t, even those associated with $q < q'$. The problem then is to determine t.

The equilibrium travel time can be determined by creating a new "performance curve" which depicts the O–D travel time as a function of the O–D flow. This curve can be constructed by summing up the link performance function horizontally, as shown in Figure 1.10. The curve $t(q)$ in this figure is an O–D performance function giving the equilibrium O–D travel time as a function of the total O–D flow. The construction of the O–D performance function in this fashion ensures that for each value of the travel time, the O–D flow is the sum of the flows on the individual links. Once this curve is constructed, the equilibrium travel time can be determined by simply entering this function with the O–D flow. Given the equilibrium travel times, the flows on the individual links can be determined graphically, as shown in the figure. Note that the O–D performance function coincides with link 1 performance function for $q \leq q'$. This means that for $q \leq q'$ the equilibrium travel time will

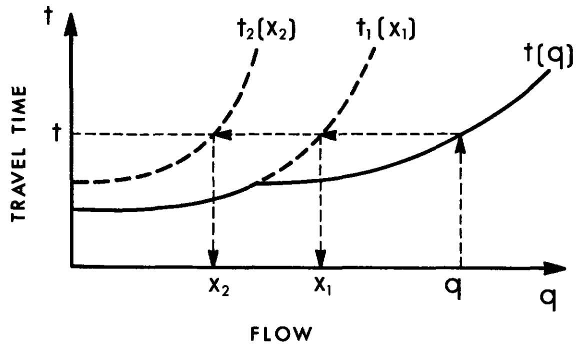


Figure 1.10 The O–D performance function, $t(q)$, is used to find the O–D travel time, t and equilibrium flows, x_1 and x_2.

be given by the travel time on link 1. The travel time on link 2 will be higher for this range of O–D flow, as required by the user equilibrium definition.

The graphical method used to solve this small two-link network example cannot be used to solve large networks. In such networks, the number of paths connecting each O–D pair may be so large that it will be prohibitively expensive to enumerate them all, even by using modern computers. Furthermore, the flow traversing each link results from the assignment of trips between many origins and many destinations. Consequently, as mentioned before, the entire network has to be solved simultaneously.

Outline

The methods used to determine the equilibrium flows and travel times described in this book are based on nonlinear optimization techniques. Part II includes a review of the mathematical background needed for such analysis. At the same time, it presents an application of these methods to the formulation and solution of user equilibrium problems over large networks.

Part III extends the basic user-equilibrium framework in two main dimensions. First, it includes travel choices other than route choice in the analysis. These include the choice of mode of travel, trip destination, and trip frequency. Second, it applies the analysis to cases in which the performance of each link depends on the flows on several (and possibly all) of the network links.

Part IV extends the analysis in yet another dimension by removing a number of the simplifying assumptions leading to the definition of the user equilibrium. This part of the book discusses the concept of stochastic user equilibrium, which was mentioned earlier. It looks at the process of route choice in the context of random utility theory, applying appropriate choice models to this process.

The last part of the book (Part V) focuses on the data needed for equilibrium analysis of urban transportation systems. It explains how O–D matrices can be derived and presents the necessary background in traffic engineering needed for the derivation of link performance functions.

1.4 SUMMARY

The problem addressed in this book is that of describing the distribution of traffic and transit passenger flows through an urban transportation network. The characteristics of the network, in terms of physical dimensions, control strategies, and operations are assumed to be fixed during the analysis period. Similarly, the pattern of activities that generates the trips in the urban area is assumed to be constant. The analysis offered is therefore descriptive in nature. It should be used for studying various scenarios, each corresponding to a possible state of any one (or a combination of) the aforementioned constant characteristics.

The approach used to solve this problem is based on the notion of equilibrium. Unlike economic equilibrium between supply and demand, the equilibrium discussed here is between performance and demand. Instead of being determined in the quantity/price space, the equilibrium under consideration is being determined in the flow/level of service (or flow/impedance) space. The flow in this case is expressed in terms of the number of travelers (or vehicles) using each component of the transportation network per unit time, while the level of service is expressed in terms of the travel time borne by these travelers. The interdependencies between the components of the network necessitates a system-based analytical approach in which the equilibrium flows and travel times throughout the network have to be determined simultaneously.

The solution approaches are developed with regard to a network representation of the physical structure of streets and transit lines. The network representation includes nodes and directed links. Each link is associated with a single flow variable and a measure of travel time. Nodes are not associated with travel time or any other impedance measure.

The graph representation of the urban network is not unique in that there are many graphs that can represent the same network. The choice of representation depends on the level of detail at which the network is modeled, which, in turn, is a function of the available data and the analysis budget.

Given the origin–destination trip rates, the demand for urban travel can be formulated in terms of the rule by which motorists (and passengers) choose a route from their origin to their destination. Different assumptions regarding these choice mechanisms lead to different equilibrium flow patterns. All route-choice models, however, assume that motorists minimize their travel time through the network. The difference between the two types of equilibria discussed in this book is that user equilibrium assumes that motorists know all link travel times with certainty. Stochastic user equilibrium models are based on the assumption that each motorist may perceive a different travel time and act accordingly.

1.5 ANNOTATED REFERENCES

The notion of demand/performance equilibrium over urban networks is presented in several transportation planning textbooks, including Morlok (1978) and Manheim (1979). In particular, the early work of Beckman et al. (1956) contains many important conceptual developments. These authors trace a description of an equilibration process over a simple traffic network to Knight (1924). A discussion of transportation networks and their representation is given by Potts and Oliver (1972), Newell (1980), and others.

The operational definition of the user equilibrium given in Section 1.3 was suggested by Wardrop (1952). The concept of stochastic user equilibrium was developed and formalized by Daganzo and Sheffi (1977).

1.6 PROBLEMS

1.1. Consider a raffle in which the prize money is known and the ticket price is constant. The number of people buying a ticket is a function of the winning probability. This probability, in turn, is a function of the number of people buying tickets. Study this situation in the framework of a demand/performance equilibrium. Show, graphically, the equilibrium point and the forces that push toward this point.

1.2. Find the user equilibrium flow patterns and the equilibrium travel time for the network described in Figure P1.1. The performance functions are

$$t_1 = 2 + x_1^2$$

$$t_2 = 1 + 3x_2$$

$$t_3 = 3 + x_3$$

where t_a and x_a denote the travel time and flow, respectively, on link a (where $a = 1, 2, 3$). The origin–destination trip rate is 4 units of flow going from node 1 to node 3.

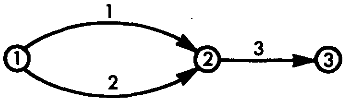


Figure P1.1

1.3. The network in Figure P1.2 includes many links connecting the origin node and the destination node. The performance curve on each link is linear, that is,

$$t_a = A_a + B_a x_a$$

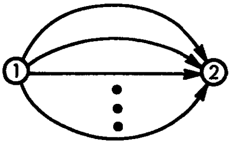


Figure P1.2

where A_a and B_a are constants and, as in Problem 1.2, t_a and x_a represent the travel time and flow, respectively, on link a. The origin–destination trip rate is q.

(a) Assuming that you know that at equilibrium, all links in this network carry flow, develop an expression for the flow on each link.

(b) How would you find the flow on each link in cases where it is not clear that all links are used at equilibrium (i.e., only a subset of the links may be used)?

II

User Equilibrium

Part II formulates and solves the standard user-equilibrium problem. The problem is to find the set of flows that corresponds to the user-equilibrium conditions set forth in Chapter 1. This part contains four chapters. Chapter 2 reviews, in short, some concepts in mathematical programming. These concepts are used in Chapter 3 to formulate the equilibrium problem as a mathematical program. Chapter 4 reviews algorithmic approaches to the solution of mathematical programs, and Chapter 5 concentrates on the solution of the user-equilibrium program.

2

Basic Concepts in Minimization Problems

This chapter reviews some concepts related to the formulation and solution of mathematical minimization programs. The focus of the discussion is on the conditions that characterize the solution of such programs and the conditions under which a solution may be unique.

In solving a mathematical program, the problem is to choose the values of a set of I variables, $x_1, x_2, \ldots, x_I$, that minimize a function (usually termed the *objective function*) $z(x_1, \ldots, x_I)$, yet comply with certain *constraints*. Each constraint can be expressed as an inequality condition on some function, $g(x_1, \ldots, x_I)$, of the variables; the set of all possible values of $x_1, \ldots, x_I$ that comply with all the constraints is known as the *feasible region*. A general minimization program with J constraints can be written (in standard form) as

$$\min z(x_1, \ldots, x_I)$$

subject to

$$\begin{gathered} g_1(x_1, \ldots, x_I) \geq b_1 \\ g_2(x_2, \ldots, x_I) \geq b_2 \\ \vdots \\ g_J(x_1, \ldots, x_I) \geq b_J \end{gathered}$$

where $g_j(x_1, \ldots, x_I) \geq b_j$ denotes the jth constraint on the values of the variables. These equations can be simplified by using vector notation. Accordingly, let $\mathbf{x}$ denote the array (vector) of decision variables, that is, $\mathbf{x} = (x_1, \ldots, x_I)$.

Using such notation, the program above can be written as

$$\min z(\mathbf{x}) \qquad [2.1a]$$

subject to

$$g_j(\mathbf{x}) \geq b_j; \qquad j = 1, \ldots, J \qquad [2.1b]$$

This is the standard form used in this text to write minimization programs. Note that all constraints are specified using a "greater than or equal to" sign. Constraint of the type "less than or equal to" can be converted to this standard form by multiplying them by -1. Equality constraints of the type $g_j(\mathbf{x}) = b_j$ can be expressed as the pair of constraints $g_j(\mathbf{x}) \geq b_j$ and $g_j(\mathbf{x}) \leq b_j$ (where the latter can be multiplied by -1 to conform to the standard form).

The solution to this program, $\mathbf{x}^*$, is a feasible value of $\mathbf{x}$ that minimizes $z(\mathbf{x})$, that is,

$$z(\mathbf{x}^*) \leq z(\mathbf{x}) \qquad \text{for any feasible } \mathbf{x} \qquad [2.2a]$$

and

$$g_j(\mathbf{x}^*) \geq b_j \qquad \forall\, j \in \mathcal{J} \qquad [2.2b]$$

where $\mathcal{J}$ is the set of constraint indices $(1, 2, \ldots, J)$†. Condition [2.2a] calls for the solution, $\mathbf{x}^*$, to have the lowest possible value of the objective function, while Eq. [2.2b] ensures that this solution satisfies the constraints. Note that the "less than or equal to" condition in Eq. [2.2a] means that there may be some value of $\mathbf{x}$ other than $\mathbf{x}^*$ which solves the program expressed in Eqs. [2.1] (hence the aforementioned concern with the conditions for uniqueness of the solution). Note also that there may be no value of $\mathbf{x}$ that satisfies all the constraints (i.e., Eqs. [2.2b] hold for no $\mathbf{x}$), in which case the program has no solution.

In the following chapter and in the remainder of this book, it is generally assumed that there is always at least one value of $\mathbf{x}$ that satisfies the constraints. Furthermore, this book deals only with programs that possess a finite minimum which occurs at a finite value of $\mathbf{x}$. To ensure the existence of such a minimum, these programs have to satisfy certain regularity conditions.‡ In addition, this book deals only with objective functions and constraints that are all continuously differentiable.

This chapter focuses on the characteristics of the optimal solutions to mathematical programs. Three topics are each discussed separately: single-variable minimization programs, multivariable programs, and some special

†The notation "$\forall$" means "for every." In this case, "$\forall\, j \in \mathcal{J}$" means "for every j in the set $\mathcal{J}$" (i.e., $j = 1, 2, \ldots, J$). When it is clear what the relevant set is, such notation is usually shortened in this book to "$\forall\, j$", meaning "for every j in the relevant set."

‡The objective function has to be continuous over the feasible region, which has to be closed and bounded.

cases of multivariable programs that are used frequently in the analysis of network equilibrium programs.

2.1 PROGRAMS IN ONE VARIABLE

The discussion of single-variable minimization is divided into sections on unconstrained and constrained problems. For each of these cases, included are discussion of the first-order conditions, which characterize the solution, and of the second-order conditions, which deal with its uniqueness. In both cases, it is assumed that the existence and smoothness conditions mentioned above hold (either for any value of the argument, in the case of an unconstrained program, or for the feasible values, in the case of constrained programs).

Unconstrained Minimization Programs

It is well known from elementary calculus that the necessary condition for a differentiable function in one variable, $z(x)$, to have a minimum at $x = x^*$ is that the derivative of $z(x)$ evaluated at x^* equals zero. In other words,

$$\frac{dz(x^*)}{dx} = 0 \qquad [2.3]$$

This is a *first-order* condition (involving only first derivatives) for a minimum.† If there is a minimum at x^*, this condition must hold. As demonstrated in Figure 2.1, however, the first-order condition is not sufficient to ensure that x^* is a minimum of $z(x)$. In this figure $dz(x)/dx = 0$ for four values of x, namely $x = a$, $x = b$, $x = c$, and $x = d$; all of which are *stationary points* of $z(x)$. Note that $x = a$ and $x = d$ are *local minima* (i.e., they minimize the function in their immediate vicinity), while $x = b$ is a point of inflection and $x = c$ is a (local) maximum.

To prove that a stationary point (such as $x = a$ in Figure 2.1) is a minimum, two characteristics have to be demonstrated: (1) that it is a local minimum and not a local maximum or an inflection point, and (2) that it is a global minimum. In other words, the value of $z(x)$ at x^* is lower than $z(x)$ at any other x (including, for example, all the other local minima, such as $x = d$ in Figure 2.1).

A function with a single minimum is called *ditonic* or *unimodal.*‡ A function is ditonic if it is strictly decreasing to the left of the minimum and strictly increasing to the right. Thus, if a function is ditonic, its stationary point is a global minimum.

†Here and later in the book, $dz(x^*)/dx$ denotes the derivative of $z(x)$ evaluated at x^*.

‡The term "unimodal" is sometimes reserved for functions with a single maximum and the term "ditonic" to functions with a single minimum (i.e., the negative of a unimodal function is a ditonic function). In this book these terms are used interchangeably to denote a function with a single minimum.

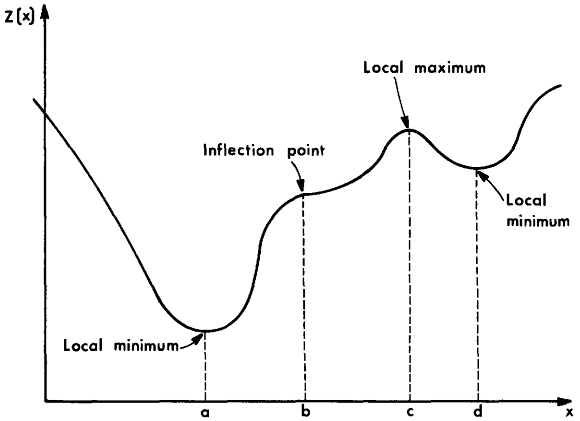


Figure 2.1 Stationary points of $z(x)$.

A sufficient condition for a stationary point to be a local minimum is for the function to be *strictly convex* in the vicinity of the stationary point. Intuitively, strict convexity means that the function is "heavy in the middle," as illustrated in the vicinity of $x = a$ and $x = d$ of Figure 2.1. Formally, strict convexity means that a line segment connecting any two points of the function lies entirely above the function. In other words, a function is strictly convex if

$$z[\theta x_1 + (1 - \theta)x_2] < \theta z(x_1) + (1 - \theta)z(x_2) \qquad [2.4a]$$

for any two distinct points x_1 and x_2 and for any value of θ where $0 < \theta < 1$. These relationships are shown in Figure 2.2a for a strictly convex function. (Note that this condition does not require that the function under consideration be differentiable.) Alternatively, strict convexity of differentiable functions can be determined by testing whether a linear approximation to the function always underestimates the function. Thus $z(x)$ is strictly convex at x_1 if and only if

$$z(x_1) + \frac{dz(x_1)}{dx}(x_2 - x_1) < z(x_2) \qquad [2.4b]$$

for any point x_2 that is not equal to x_1. This condition is demonstrated in Figure 2.2b. Finally, if a function is twice differentiable in the vicinity of a stationary point, the strict convexity condition is equivalent to requiring that the second derivative of $z(x)$ at x^* be positive, that is,

$$\frac{d^2z(x^*)}{dx^2} > 0 \qquad [2.4c]$$

Strict convexity can thus be defined by the line segment condition [2.4a], by the linear approximation condition [2.4b] (for continuously differentiable functions), or by the second derivative condition [2.4c] (for twice-continuously

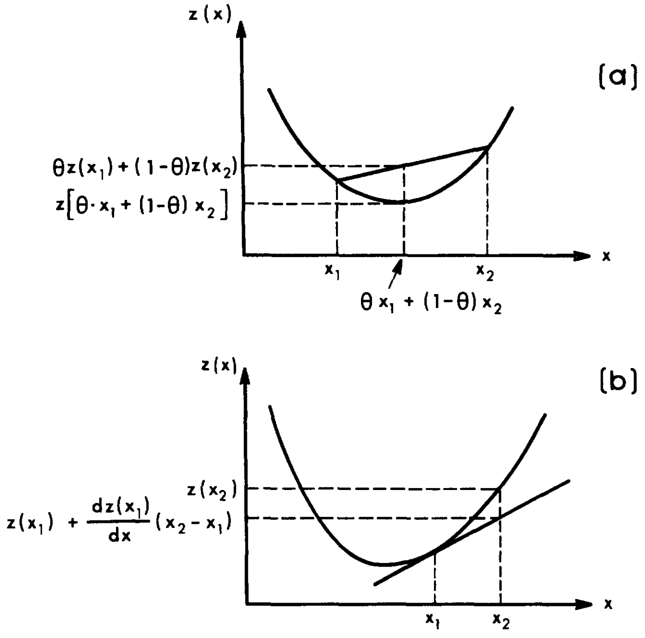


Figure 2.2 Strict convexity can be demonstrated (a) by showing that a line segment connecting any two points lies entirely above the function, or (b) by showing that a linear approximation always underestimates the function.

differentiable functions). If a function is strictly convex in the vicinity of a stationary point, this point is a local minimum.

If a function includes more than one local minimum, it is usually difficult to demonstrate that a particular local minimum is also a global one. Doing so may require identification of all the local minima (a process that may involve repeated solutions) and a comparison of the objective function values at these points. Such a task can be very difficult and is sometimes impossible. If, however, x^* is the only local minimum of $z(x)$, then, naturally, it is also a global minimum. To demonstrate that a local minimum is unique, it is sufficient to show that $z(x)$ is convex for all values of x (i.e. that it is globally convex).

Convexity means that the inequality in Eq. [2.4a] is replaced by a "less than or equal to" sign, meaning that a line segment never lies below the function. Similarly, convexity means that a linear approximation to the function never overestimates it and that the second derivative is nonnegative.

To understand second-order conditions better, note that a function $z(x)$ can have a unique minimum and not be globally convex, and that convexity alone does not guarantee the uniqueness of the minimum. Figure 2.3 illustrates three possible situations: In Figure 2.3a $z(x)$ is convex, but it is not strictly convex in the vicinity of x^*—this function does not have a unique minimum.

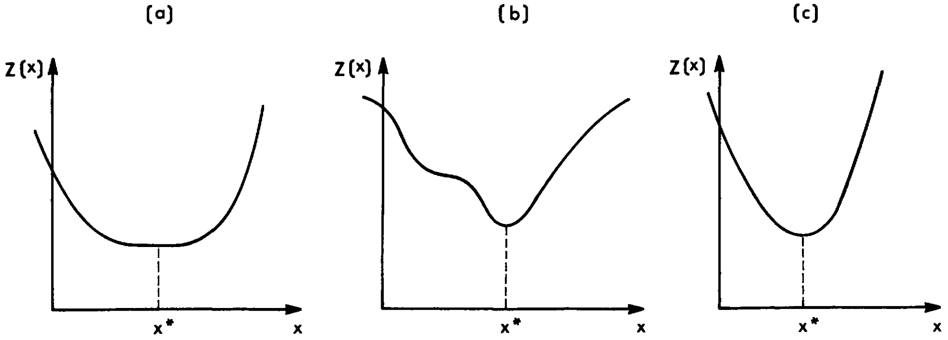


Figure 2.3 Convexity and unimodality. (a) $z(x)$ is convex but not strictly convex; its minimum is not necessarily unique. (b) $z(x)$ is not convex but it is strictly convex at the vicinity of x^*; it is also ditonic and thus x^* is a unique minimum. (c) $z(x)$ is convex everywhere and strictly convex at the vicinity of x^*, which is the unique minimum.

Figure 2.3b depicts a case in which $z(x)$ is not convex even though it is strictly convex in the vicinity of x^*. This function is ditonic and thus its only stationary point is also its unique minimum. In general, however, for a function of this shape it would be hard to demonstrate the uniqueness of the minimum because the function is not convex. Figure 2.3c illustrates a convex function that is strictly convex in the vicinity of its unique minimum, x^*.

In summary, then, it is sufficient to show that a function $z(x)$ is strictly convex at $x = x^*$ and convex elsewhere in order to establish the fact that x^* minimizes that function. A necessary condition, however, for x^* to be a minimum is for the first derivative to vanish at that point. (This is the first-order condition for a minimum.)

This review of programs in a single variable is extended in the next section to constrained minimizations. Again, the focus of the discussion is on the first- and second-order conditions for a minimum.

Constrained Minimization Programs

In constrained minimization it is possible to have a minimum where the first derivative does not vanish. As a result, the first-order conditions have to be generalized in order to include this possibility. Consider, for example, Figure 2.4, showing the function $z(x)$ defined over the feasible region $a' \leq x \leq a''$. In the first case (Figure 2.4a), the minimum point is internal (inside the feasible region) and the condition $dz(x^*)/dx = 0$ holds, while in Figure 2.4b the minimum point is on the boundary, at $x = a'$. Clearly, $dz(a')/dx \neq 0$.

When the minimum is on the boundary of the feasible region, however, the following observation can be made: If the constraint on which the solution lies (the binding constraint) is of the type $x \geq (\cdot)$, then $z(x)$ must be increasing

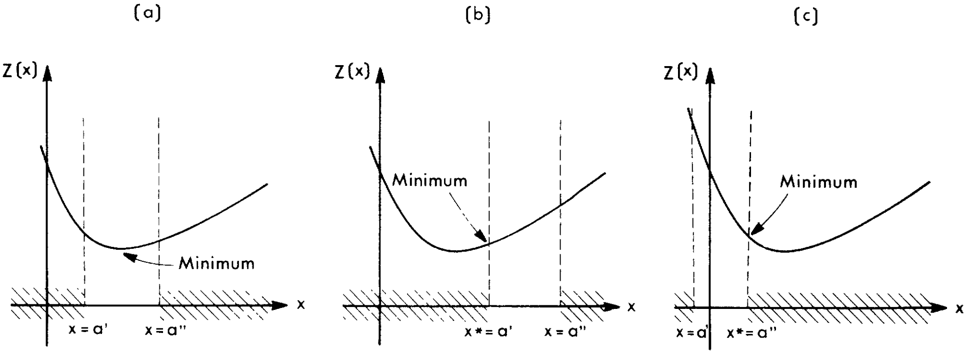


Figure 2.4 Constrained minimum: (a) internal minimum—$dz(x^*)/dx = 0$ inside the feasible region; (b) minimum on the boundary of the feasible region—the binding constraint is $x \geq a$ and $dz(x)/dx \geq 0$; (c) minimum on the boundary—the binding constraint is $x \leq b$ and $dz(x^*)/dx \leq 0$.

(or more precisely nondecreasing) at x^*, as is the case in Figure 2.4b. If, on the other hand, the binding constraint is of the type $x \leq (\cdot)$, then $z(x)$ must be decreasing (actually, nonincreasing) at x^*, as in the case shown in Figure 2.4c.

To characterize these situations mathematically in a statement of first-order conditions, the problem depicted in Figure 2.4 is first written in a standard form, as follows (see Eqs. [2.1]):

$$\min z(x) \qquad [2.5a]$$

subject to

$$x \geq a' \qquad [2.5b]$$

$$-x \geq -a'' \qquad [2.5c]$$

where constraints [2.5b] and [2.5c] are expressed as $g_j(x) \geq b_j$ [in this case $g_1(x) \equiv x$ and $b_1 \equiv a'$, while $g_2(x) \equiv -x$ and $b_2 \equiv -a''$].

When the constraints are expressed in standard form, the aforementioned observation about the slope of $z(x)$ at x^* means that the derivative of $z(x)$ at x^* and the derivative of the binding constraint at this point $[dg(x^*)/dx]$ will always have the same sign (or their product will be zero, that is, they will never have opposite signs). This observation can be used to extend the first-order conditions because it holds whenever a minimum occurs on the boundary of the feasible region as well as for an unconstrained solution. In the example given in Figure 2.4b, this condition can be verified since

$$\frac{dg_1(x^*)}{dx} = 1$$

and from the figure

$$\frac{dz(x^*)}{dx} > 0$$

whereas for Figure 2.4c,

$$\frac{dg_2(x^*)}{dx} = -1 \quad \text{and} \quad \frac{dz(x^*)}{dx} < 0$$

Since both derivatives are scalars, the condition of same signs can be written as

$$\frac{dz(x^*)}{dx} = u_j \frac{dg_j(x^*)}{dx} \qquad \text{for either } j = 1 \quad \text{or} \quad j = 2 \tag{2.6}$$

where u_j is a nonnegative scalar and $g_j(x)$ is the binding constraint (which can be either the first or the second one).

In order to develop a set of first-order conditions that would apply in all cases (i.e., whether the solution is internal or on the boundary of the feasible region), define a nonnegative variable, u_j, for each of the constraints (not only the binding ones). If the jth constraint is not binding [i.e., if $g_j(x^*) > b_j$], let $u_j = 0$. In other words, the condition

$$[b_j - g_j(x^*)]u_j = 0$$

holds for all the constraints $j \in \mathcal{J}$. (It means that either $u_j = 0$, in which case the jth constraint may or may not be binding, or $u_j \geq 0$ and the jth constraint is binding.) Using this convention, conditions [2.6] can be replaced by the condition

$$\frac{dz(x^*)}{dx} = \sum_j u_j \frac{dg_j(x^*)}{dx}$$

where u_j is a nonnegative scalar and the sum includes all the constraints.† Note that this equation generalizes the condition for an unconstrained minimum (see Eq. [2.3]). An unconstrained minimization problem can be looked upon as a constrained problem with an internal minimum point. In such a case, all $u_j = 0$ (since none of the constraints is binding) and the condition above simply states that $dz(x^*)/dx = 0$.

In summary, then, the first-order conditions for a constrained minimum are

$$\frac{dz(x^*)}{dx} = \sum_j u_j \frac{dg_j(x^*)}{dx} \tag{2.7a}$$

$$u_j \geq 0, \quad u_j[b_j - g_j(x^*)] = 0, \quad g_j(x^*) \geq b_j \qquad \text{for } j = 1, \ldots, J \tag{2.7b}$$

These conditions mean that the derivative of the function at the minimum and

†A notational convention used throughout this text is to omit the complete summation notation whenever the summation goes over all the indices (or subscripts) in the relevant set. In this expression "$\sum_j$" is equivalent to "$\sum_{j=1}^{J}$" or "$\sum_{\forall j \in \mathcal{J}}$."

the derivatives of the binding constraints at the minimum have the same signs and that the minimum point is (of course) feasible. Equations [2.7] thus depict the general form of the first-order conditions for the minimum of a single-variable function (see Problem 2.5).

The second-order conditions for a constrained minimum are similar to those for the constrained case, but with one condition added. As in the unconstrained case, strict convexity of $z(x)$ in the vicinity of x^* (where Eqs. [2.7] hold) would ensure that x^* is a local minimum; convexity, however, is not quite sufficient to guarantee the uniqueness of that minimum. Consider, for example, the situation depicted in Figure 2.5, where a strictly convex function $z(x)$ is constrained to the following region: either $0 \leq x \leq a$ or $b \leq x \leq c$. This function depicts two local minima at $x = a$ and at $x = b$ even though $z(x)$ is strictly convex everywhere. Again, proving that a local minimum is also a global one is difficult in the presence of other minima. A condition excluding situations such as the one depicted in Figure 2.5 is therefore added to the uniqueness requirements. This condition requires that the constraints define a convex set of feasible points, or in other words, that the feasible region be convex. The convexity of the feasible region means that a line segment connecting any two points of that feasible region must lie entirely within the region. This condition ensures the existence of only one minimum, which must necessarily be the global minimum. It obviously does not hold for the problem illustrated in Figure 2.5, which does not have a unique minimum.

In conclusion, then, the necessary conditions for a minimum are given by Eqs. [2.7]. The sufficient conditions include strict convexity in the vicinity of the minimum point (this guarantees that it is a local minimum) and convexity of both the objective function and the feasible region (this guarantees that there are no other local minima).

This concludes the discussion of single-variable programs. The concepts introduced in this section are expanded in the following section to the multivariable case.

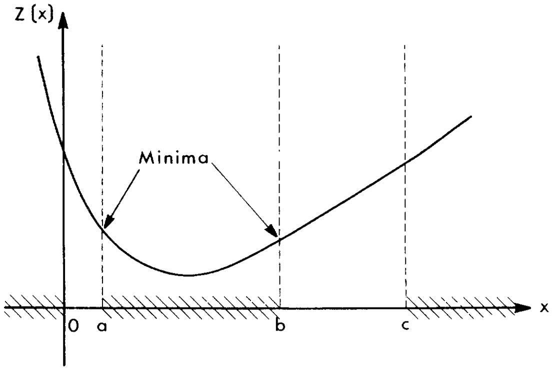


Figure 2.5 Two local minima for a strictly convex function bounded by a constraint set that is not convex.

2.2 MULTIDIMENSIONAL PROGRAMS

In multidimensional problems, the objective function is given by $z(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_I)$ is a vector of length I (i.e., it is an array of variables $x_1, x_2, \ldots$ up to x_I). The constraints are given by $g_j(\mathbf{x}) \geq b_j$, $\forall\, j \in \mathscr{J}$, where $\mathscr{J}$ is the constraints index set $(1, 2, \ldots, J)$ and b_j is a scalar. Again this discussion of multivariable minimization programs deals separately with unconstrained and constrained problems, including a review of both first- and second-order conditions in each case. As in the preceding section, the regularity conditions on $z(\mathbf{x})$ and $g_j(\mathbf{x})$ are assumed to hold, and a solution therefore always exists.

Unconstrained Minimization Programs

If the function to be minimized is unconstrained, the first-order condition for a minimum at $\mathbf{x} = \mathbf{x}^*$ is that the gradient of $z(\mathbf{x})$ vanish at $\mathbf{x}^*$. The gradient of $z(\mathbf{x})$ with respect to $\mathbf{x}$, $\nabla_{\mathbf{x}} z(\mathbf{x})$, is the vector of partial derivatives, that is,

$$\nabla_{\mathbf{x}} z(\mathbf{x}) = \left(\frac{\partial z(\mathbf{x})}{\partial x_1}, \frac{\partial z(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial z(\mathbf{x})}{\partial x_I} \right) \qquad [2.8]$$

The subscript $\mathbf{x}$ in the gradient notation is usually omitted if the variable with respect to which the partial derivatives are taken is obvious. At every point $\mathbf{x}$, the gradient aims in the direction of the steepest (local) increase in $z(\mathbf{x})$; thus the gradient is the directional derivative of $z(\mathbf{x})$ in the direction of the steepest ascent. As mentioned above, the first-order condition for a minimum is

$$\nabla z(\mathbf{x}^*) = \mathbf{0} \qquad [2.9a]$$

meaning that each component of the gradient (see Eq. [2.8]) has to be equal to zero. In other words,

$$\frac{\partial z(\mathbf{x})}{\partial x_i} = 0 \qquad \text{for } i = 1, 2, \ldots, I \qquad [2.9b]$$

This condition is entirely analogous to the condition $dz(x^*)/dx = 0$ used in the single-variable case. It is only a necessary condition for a minimum; that is, it establishes the fact that $z(\mathbf{x})$ has a stationary point at $\mathbf{x}^*$.

As an example of the application of the first-order conditions, consider the function $z(x_1, x_2)$ where

$$z(x_1, x_2) = x_1^2 + 2x_2^2 + 2x_1x_2 - 2x_1 - 4x_2 \qquad [2.10]$$

The gradient of this function at every point (x_2, x_2) is given by

$$\nabla z(x_1, x_2) = [(2x_1 + 2x_2 - 2), (4x_2 + 2x_1 - 4)] \qquad [2.11]$$

The stationary point is where $\nabla z(x_1, x_2) = 0$, which is the value of $\mathbf{x}$ that solves the equations

$$2x_1 + 2x_2 - 2 = 0 \qquad [2.12a]$$

$$2x_1 + 4x_2 - 4 = 0 \qquad [2.12b]$$

The solution of these equations is $x_1^* = 0$ and $x_2^* = 1$. Thus $\mathbf{x}^* = (0, 1)$ is a stationary point of $z(x_1, x_2)$ in Eq. [2.10].

To show that a stationary point, $\mathbf{x}^*$, is a local minimum, it is sufficient to demonstrate that $z(\mathbf{x})$ is strictly convex in the vicinity of $\mathbf{x} = \mathbf{x}^*$, as was the case with the single-dimensional function. The conditions for strict convexity in the multidimensional case parallel those given in the preceding section for single-variable functions (see Eqs. [2.4]). If a line segment lies entirely above the function or if it is underestimated by a linear approximation, the function is strictly convex. Thus the strict convexity condition is

$$z[\theta\mathbf{x}' + (1 - \theta)\mathbf{x}''] < \theta z(\mathbf{x}') + (1 - \theta)z(\mathbf{x}'') \qquad [2.13a]$$

where $\mathbf{x}'$ and $\mathbf{x}''$ are two different values of $\mathbf{x}$, and θ is a scalar between zero and 1, or†

$$z(\mathbf{x}') + \nabla z(\mathbf{x}') \cdot (\mathbf{x}'' - \mathbf{x}')^T < z(\mathbf{x}'') \qquad [2.13b]$$

where the superscript T denotes the vector (or matrix) transposition operation. Alternatively, a function $z(\mathbf{x})$ can be shown to be strictly convex if some conditions regarding the second derivatives of $z(\mathbf{x})$ hold. To express these conditions mathematically, the second derivatives are usually arranged in a (symmetric) matrix form. [This matrix is known as the *Hessian* of $z(\mathbf{x})$]. The Hessian, denoted by $\nabla^2 z(\mathbf{x})$, is given by

$$\nabla^2 z(\mathbf{x}) \equiv \begin{bmatrix} \dfrac{\partial^2 z(\mathbf{x})}{\partial x_1^2} & \dfrac{\partial^2 z(\mathbf{x})}{\partial x_1\, \partial x_2} & \cdots & \dfrac{\partial^2 z(\mathbf{x})}{\partial x_1\, \partial x_I} \\ \dfrac{\partial^2 z(\mathbf{x})}{\partial x_2\, \partial x_1} & \dfrac{\partial^2 z(\mathbf{x})}{\partial x_2^2} & & \\ \vdots & & & \vdots \\ \dfrac{\partial^2 z(\mathbf{x})}{\partial x_I\, \partial x_1} & & \cdots & \dfrac{\partial^2 z(\mathbf{x})}{\partial x_I^2} \end{bmatrix}$$

The mathematical condition which ensures that $z(\mathbf{x})$ is strictly convex is that the Hessian, $\nabla^2 z(\mathbf{x})$, is positive definite.‡ If the Hessian is positive definite at $\mathbf{x}^*$, this point is a local minimum of $z(\mathbf{x})$. As in the single-dimensional case, it is required that a function be convex everywhere in order to ensure that it has but one local minimum.§

†This condition can be written in expanded notation as follows:

$$z(x_1', \ldots, x_I') + \sum_i \frac{\partial z(x_1', \ldots, x_I')}{\partial x_i}(x_i'' - x_i') < z(x_1'', \ldots, x_I'')$$

‡A matrix, $\mathbf{H}$, is positive definite if the product $\mathbf{h} \cdot \mathbf{H} \cdot \mathbf{h}^T$ is positive for any nonzero vector, $\mathbf{h}$. A symmetric matrix (such as the Hessian) can also be shown to be positive definite if and only if its eigenvalues are positive or if its leading principal minor determinants are positive.

§Convexity is associated with a positive-semidefinite Hessian. A matrix $\mathbf{H}$ is positive semidefinite if and only if $\mathbf{h} \cdot \mathbf{H} \cdot \mathbf{h}^T \geq 0$ for any nonzero vector $\mathbf{h}$. Conditions [2.13a and 2.13b] indicate convexity if the "less than" sign is replaced by a "less than or equal to" sign.

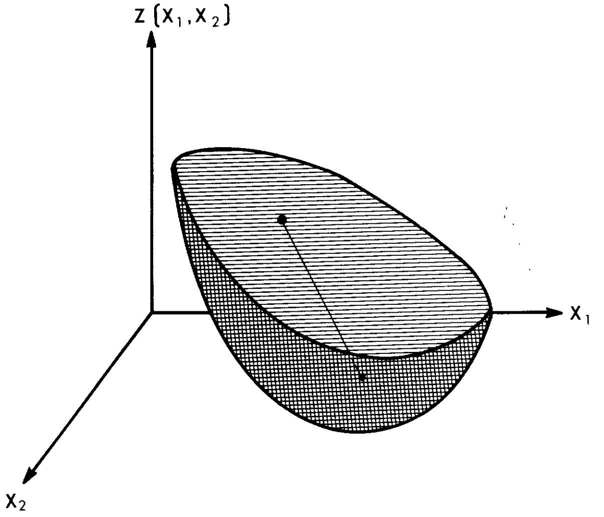


Figure 2.6 Strictly convex function in two dimensions.

It is important to note that the aforementioned conditions for a matrix to be positive definite means that any diagonal matrix (i.e., a matrix in which only the elements along the diagonal are nonzero) with positive elements is positive definite. Many of the objective functions discussed in this book have diagonal Hessians and it is therefore only necessary to check the signs of the diagonal elements to establish that such a Hessian is positive-definite, with the implication that the objective function is strictly convex.

Figure 2.6 shows the shape of a strictly convex function in two dimensions. The figure also shows a line segment connecting two points on this function. As required by the strict convexity condition, this line lies entirely above the function.

It should be noted here that if a line segment connecting any two points of $z(\mathbf{x})$ never lies above the function, $z(\mathbf{x})$ is concave. Concavity (which is a sufficient condition for a local maximum to be unique) can also be identified by a linear approximation of $z(\mathbf{x})$ overestimating (or rather, not underestimating) the function, or by an appropriate condition on the second derivatives.† Consequently, the negative of a concave function is a convex function, and vice versa. Note also that if a (strictly) convex function is multiplied by a positive constant, the product is still a (strictly) convex function, and that the sum of (strictly) convex functions is also a (strictly) convex function (see Problem 2.9). These properties are used later in the text to demonstrate convexity.

To see how the second-order conditions are applied, consider the example of minimizing $z(x_1, x_2)$ in Eq. [2.10]. The stationary point, $x^* = (0, 1)$, is a

†The Hessian of a concave function is negative semidefinite and the Hessian of a strictly concave function is negative definite.

local minimum if the function is strictly convex at that point. If the function is also convex everywhere, then (0, 1) is a global minimum.

The function $z(x_1, x_2)$ in Eq. [2.10] turns out to be strictly convex everywhere. To see this, condition [2.13b] can be used to check if

$$z(x_1, x_2) + \frac{\partial z(x_1, x_2)}{\partial x_1}(y_1 - x_1) + \frac{\partial z(x_1, x_2)}{\partial x_2}(y_2 - x_2) < z(y_1, y_2)$$

for any two points (x_1, x_2) and (y_1, y_2). Substitution of the appropriate functional forms (see Eqs. [2.10] and [2.11]) leads to the inequality

$$[(y_1 - x_1) + (y_2 - x_2)]^2 + (y_2 - x_2)^2 > 0$$

This inequality holds for any two distinct points (x_1, x_2) and (y_1, y_2), meaning that the strict convexity condition [2.13b] is satisfied by $z(x_1, x_2)$ in Eq. [2.10]. This function is, therefore, strictly convex everywhere.† Consequently, it can be concluded that $\mathbf{x}^* = (0, 1)$ is a local as well as a global minimum of $z(x_1, x_2)$. The minimum value of $z(x_1, x_2)$ is $z(0, 1) = -2$.

Constrained Minimization Programs

As in the single-variable case, the focus of the review in this section is on the first- and second-order conditions for a minimum. As in that case, the minimum of a constrained multidimensional program can occur on the boundary of the feasible region, where the condition $\nabla z(\mathbf{x}^*) = \mathbf{0}$ may not hold.

Consider the two-dimensional program depicted in Figure 2.7a, where the feasible region is defined by six constraints and the objective function, $z(\mathbf{x})$, is illustrated by the contour lines. The problem is

$$\min z(x_1, x_2)$$

subject to

$$g_j(x_1, x_2) \geq b_j; \qquad j = 1, 2, \ldots, 6$$

It is apparent from the figure that the solution of this problem, $\mathbf{x}^* = (x_1^*, x_2^*)$, lies at the intersection of constraints 2 and 3. These constraints are therefore binding at the minimum, or, in other words, $g_2(x_1^*, x_2^*) = b_2$ and $g_3(x_1^*, x_2^*) = b_3$.

The direction perpendicular to any curve of the form $g(x_1, x_2) =$ constant, at some point, $\mathbf{x}' = (x_1', x_2')$, is given by the gradient of the curve at

†The strict convexity can also be checked by using the condition on the Hessian, which is given by

$$\mathbf{H} = \begin{pmatrix} 2 & 2 \\ 2 & 4 \end{pmatrix}$$

This matrix is positive definite (its first leading minor equals 2 and its determinant, which is its second leading minor equals $2 \cdot 4 - 2 \cdot 2 = 4$). Both leading minors are positive and $\mathbf{H}$ is, therefore, positive definite, meaning that $z(x_1, x_2)$ is strictly convex everywhere.

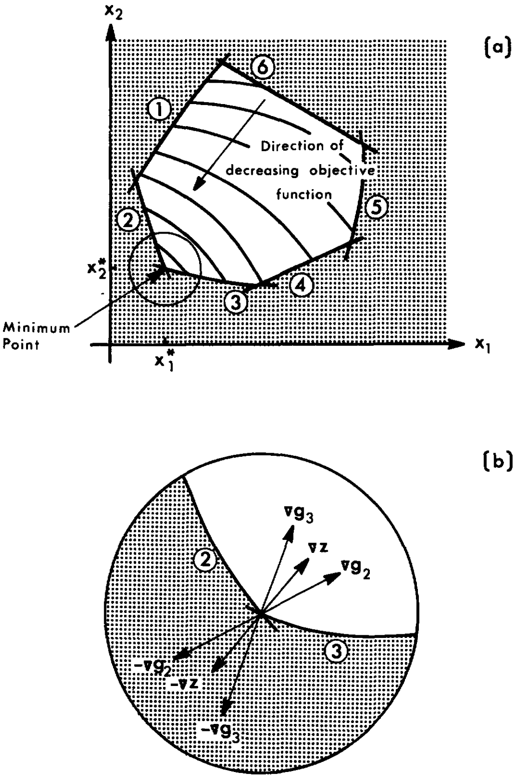


Figure 2.7 Kuhn–Tucker conditions: (a) contours of the objective function and the feasible region; (b) the gradient of the objective function lies within the cone generated by the gradients of the binding constraints.

that point, that is, by the vector

$$\nabla g(\mathbf{x}') = \left(\frac{\partial g(\mathbf{x}')}{\partial x_1}, \frac{\partial g(\mathbf{x}')}{\partial x_2}\right)$$

The enlarged portion shown in Figure 2.7b depicts the point (x_1^*, x_2^*) and the gradients of the two binding constraint functions at this point, $\nabla g_2(\mathbf{x}^*)$ and $\nabla g_3(\mathbf{x}^*)$. The definition of the first-order conditions for constrained problems rests on the observation that the gradient of the objective function at this point, $\nabla z(\mathbf{x}^*)$, must lie between the gradients of the binding constraints. The gradient vector, $\nabla z(\mathbf{x}^*)$, and its opposite vector, $-\nabla z(\mathbf{x}^*)$, as well as $-\nabla g_2(\mathbf{x}^*)$ and $-\nabla g_3(\mathbf{x}^*)$, are all shown in Figure 2.7b, where the aforementioned condition can be intuitively verified. If, for example, $-\nabla z(\mathbf{x}^*)$ were to point below $-\nabla g_3(\mathbf{x}^*)$, $z(\mathbf{x})$ could be further decreased by sliding to the right, along $g_3(\mathbf{x})$ (see Figure 2.7b). Similarly, if $-\nabla z(\mathbf{x}^*)$ were to point above $-g_2(\mathbf{x}^*)$, $z(\mathbf{x})$ could surely be decreased further by sliding upward along $g_2(\mathbf{x}^*)$. Only when $-\nabla z(x^*)$ is between $-\nabla g_2(\mathbf{x}^*)$ and $-\nabla g_3(\mathbf{x}^*)$ can the objective function not be decreased further. This condition is a generalization of the "same signs" condition which was applicable in the single-variable case (see Figure 2.4 and the related argument).

With multidimensional functions, the constraints define some surfaces and the gradient of each of these constraints at a given point is a vector that is perpendicular (normal) to the surface at that point. The observation on which the first-order conditions are based for this case in that the gradients of the binding constraints create an imaginary "cone" within which the gradient of the objective function must lie.

Thus the first-order conditions can be written as a specification of the direction of the gradient of $z(\mathbf{x})$ at $\mathbf{x}^*$ in relation to the directions of the binding constraints. In the multidimensional case, the requirement is that $\nabla z(\mathbf{x}^*)$ be expressed as a linear combination (with nonnegative coefficients) of the gradients of the binding constraints. For the example in Figure 2.7, this condition is

$$\nabla z(\mathbf{x}^*) = u_2 \nabla g_2(\mathbf{x}^*) + u_3 \nabla g_3(\mathbf{x}^*)$$

where u_2 and u_3 are nonnegative scalars associated with the second and third constraints, respectively. The same condition can be written in expanded notation as

$$\frac{\partial z(\mathbf{x}^*)}{\partial x_i} = u_2 \frac{\partial g_2(\mathbf{x}^*)}{\partial x_i} + u_3 \frac{\partial g_3(\mathbf{x}^*)}{\partial x_i} \qquad \text{for } i = 1, 2$$

In order to generalize this condition to problems of the type

$$\min z(\mathbf{x})$$

subject to

$$g_j(\mathbf{x}) \geq b_j \qquad \forall j \in \mathcal{J}$$

an auxiliary variable, u_j, can be defined for each constraint. In a fashion analogous to the single-dimensional case, let u_j be nonnegative if the jth constraint is binding at $\mathbf{x}^*$, and let $u_j = 0$ for any nonbinding constraint.

The first-order conditions can now be written as follows:†

$$\frac{\partial z(\mathbf{x}^*)}{\partial x_i} = \sum_j u_j \frac{\partial g_j(\mathbf{x}^*)}{\partial x_i} \qquad \text{for } i = 1, 2, \ldots, I \qquad [2.14a]$$

and

$$u_j \geq 0, \quad u_j[b_j - g_j(\mathbf{x}^*)] = 0, \quad g_j(\mathbf{x}^*) \geq b_j \qquad \forall j \in \mathcal{J} \qquad [2.14b]$$

Conditions [2.14] are known as the *Kuhn–Tucker* conditions, named after the two mathematicians who first proved that these are the conditions necessary

†Eq. [2.14a] can be written in vector notation as

$$\nabla z(\mathbf{x}^*) = \sum_j u_j \nabla g_j(\mathbf{x}^*)$$

for a constrained minimum. The auxiliary variables u_j are known as *dual variables* as well as *Lagrange multipliers* (since Eq. [2.14a] is identical to the Lagrangian condition in classical optimization theory, as shown in Section 2.3). The condition $u_j[b_j - g_j(\mathbf{x}^*)] = 0, \forall\, j \in \mathcal{J}$, is known as the *complementary slackness* condition.

The Kuhn–Tucker conditions include the observation (Eq. [2.14a]) that the gradient of the objective function, at the minimum, can be expressed as a linear combination, with nonnegative coefficient, of the gradients of the binding constraints. Equations [2.14b] include the condition that these coefficients (which are the dual variables) are, in fact, nonnegative. It also includes the complementary slackness condition, which states that the values of the dual variable associated with each nonbinding constraint is zero. These three conditions mean that if none of the constraints is binding (or if the program is unconstrained), the first-order condition is simply $\nabla z(\mathbf{x}^*) = 0$. The last set of conditions in Eq. [2.14b] ensures that the optimal solution is feasible.

As it turns out, there are some cases, which are seldom encountered in practice, in which the Kuhn–Tucker conditions do not hold at the minimum. If the set of constraints satisfies certain *constraint qualifications*, however, these situations will not arise. For the purposes of this text, it is sufficient to require that the constraint set define a convex feasible region. This condition is explained below in conjunction with the second-order conditions for a constrained minimum of a multidimensional function. The constraint qualification is always satisfied for a convex feasible region.

The values of the dual variables at the solution point provide valuable information about the sensitivity of the value of $z(\mathbf{x})$ at its minimum [i.e., $z(\mathbf{x}^*)$] to the constraints. If the right-hand side of the jth constraint, $g_j(\mathbf{x}) \geq b_j$, is relaxed by some small amount, Δb_j, the minimum value of $z(\mathbf{x})$ will decrease by $u_j\, \Delta b_j$. Thus u_j is a measure of the sensitivity of the minimum value of $z(\mathbf{x})$ to the restriction imposed by the jth constraint. If a constraint is not binding, one would expect that its relaxation would not affect $z(\mathbf{x}^*)$ and indeed, for these constraints $u_j = 0$.

The determination of the first-order conditions of a constrained minimum can be exemplified by using $z(x_1, x_2)$ in Eq. [2.10]. At this point, however, assume that the problem is to minimize the function

$$z(x_1, x_2) = x_1^2 + 2x_2^2 + 2x_1x_2 - 2x_1 - 4x_2 \qquad [2.15a]$$

subject to

$$x_1 + x_2 \geq 2 \qquad [2.15b]$$

Before this problem is attempted, the known unconstrained solution should be tested for feasibility since if it is feasible, it will also be optimal. (Why?) As calculated before, $\mathbf{x}^* = (0, 1)$; this point, unfortunately, is not feasible since constraint [2.15b] is not satisfied.

The Kuhn–Tucker conditions for this problem are as follows:†

$$2x_1^* + 2x_2^* - 2 = u \tag{2.16a}$$

$$4x_2^* + 2x_1^* - 4 = u \tag{2.16b}$$

$$u(2 - x_1^* - x_2^*) = 0 \tag{2.16c}$$

$$x_1^* + x_2^* \geq 2 \tag{2.16d}$$

$$u \geq 0 \tag{2.16e}$$

To solve these equations, note that if $u = 0$, Eqs. [2.16a] and [2.16b] reduce to [2.12a] and [2.12b]. The solution of these equations is $\mathbf{x} = (0, 1)$, a solution that does not satisfy Eq. [2.16d]. Consequently, $u > 0$, meaning that the solution satisfies the three equations

$$2x_1^* + 2x_2^* - 2 - u = 0 \tag{2.17a}$$

$$2x_1^* + 4x_2^* - 4 - u = 0 \tag{2.17b}$$

$$x_1^* + x_2^* - 2 = 0 \tag{2.17c}$$

The solution of this system of equations is

$$x_1^* = 1$$

$$x_2^* = 1$$

$$u^* = 2$$

The value of the constrained minimum is $z(1, 1) = -1$, which is, of course, higher than the value of the unconstrained minimum. (Check this.)

The second-order conditions for constrained multidimensional minimization programs include an additional requirement (in comparison with unconstrained problems), as in the single-dimensional case. For a program to have a unique minimum, the constraint set itself has to define a convex feasible region. Figure 2.8a illustrates two local minima that may occur under a nonconvex feasible region. Figures 2.8b and c illustrate convex feasible regions, where a line segment connecting any two points of each region lies entirely within the region. Hence, to ensure that $\mathbf{x}^*$ is a global minimum, convexity of the feasible region should be required in addition to requiring the convexity of $z(\mathbf{x})$ for all feasible $\mathbf{x}$ and strict convexity of $z(\mathbf{x})$ at $\mathbf{x}^*$.

As in the single-dimensional case, convexity is a relatively strong condition; it is sufficient to require only that a function be ditonic in order to ensure the uniqueness of a minimum. It is, however, easier to prove convexity

†The first two conditions correspond to Eq. [2.14a], stating that equality for each of the two components of the gradient vector. To relate this to Eq. [2.14a], note that the constraint in this problem is expressed in terms of the function $g(x_1, x_2) = x_1 + x_2$, for which $\partial g(\mathbf{x})/\partial x_1 = 1$ and $\partial g(\mathbf{x})/\partial x_2 = 1$. The last three equations above correspond to Eq. [2.14b]. Equation [2.16c] states the complementary slackness condition, Eq. [2.16d] requires the feasibility of the solution, and Eq. [2.16e] states the nonnegativity of the dual variable.

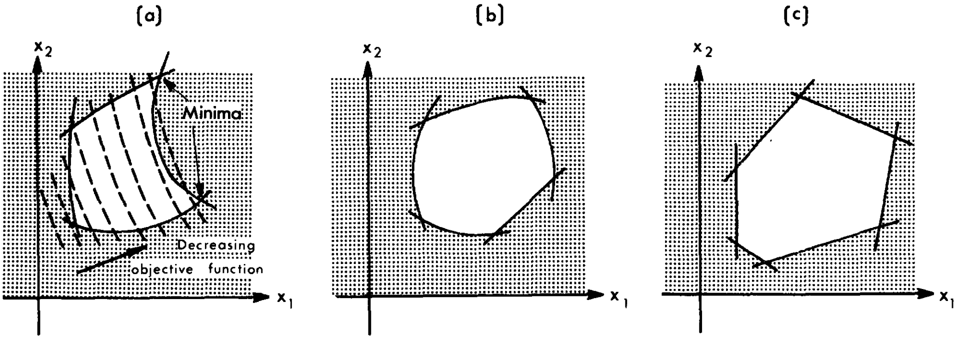


Figure 2.8 Shape of the feasible region: (a) nonconvex feasible region—two local minima are illustrated; (b) convex feasible region; (c) convex feasible region with linear constraints.

and this should always be tried first. Furthermore, in all cases dealt with in this text, the objective function is twice continuously differentiable, meaning that the second derivatives (Hessian) condition can be used to determine convexity. Also, this book deals exclusively with linear constraints and convex feasible regions.

2.3 SOME SPECIAL PROGRAMS

This section deals with several cases of multidimensional constrained minimization programs that are of special interest in the study of equilibrium assignment. Included in this discussion are programs with nonnegativity constraints, programs with equality constraints, programs with both nonnegativity and equality constraints, and linear programs. This section also suggests an alternative approach to the statement of the first-order conditions of any minimization program. This statement is based on the concept of Lagrangians, which is explained in relation to the programs discussed here.

Nonnegativity Constraints

The first-order conditions for the case in which the feasible region includes all nonnegative values of **x** can be posed without reference to the dual variables. In the one-dimensional case, the program "min $z(x)$ subject to $x \geq 0$" can result in the two situations shown in Figure 2.9. In Figure 2.9a the solution is at a point where $dz(x^*)/dx = 0$ (and $x^* \geq 0$), while in Figure 2.9b, $x^* = 0$, since the nonnegativity constraint is binding [and $dz(x^*)/dx \geq 0$]. The first-order condition for this problem can be written in a form encompassing both these possible situations, as follows:

$$x^* \frac{dz(x^*)}{dx} = 0 \quad \text{and} \quad \frac{dz(x^*)}{dx} \geq 0$$

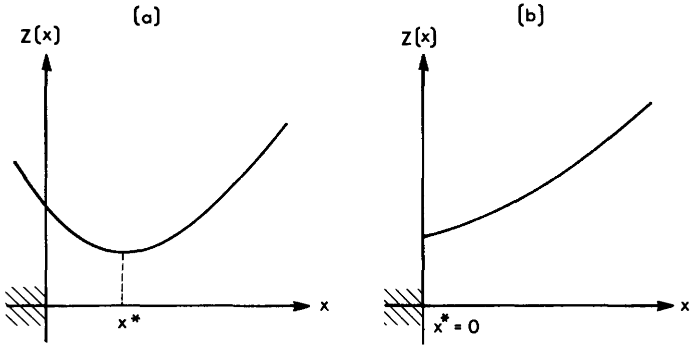


Figure 2.9 First-order conditions for a program with nonnegativity constraints: (a) internal minimum with $dz(x^*)/dx = 0$ and $x^* \geq 0$; (b) constrained minimum with $x^* = 0$ and $dz(x^*)/dx \geq 0$.

Both conditions hold at the minimum point of $z(x)$ (i.e., at x^*). In addition, the constraint $x^* \geq 0$ has to hold as well.

Similarly, in the multidimensional case, the solution of the program

$$\min z(\mathbf{x}) \qquad [2.18a]$$

subject to

$$x_i \geq 0; \qquad i = 1, 2, \ldots, I \qquad [2.18b]$$

can occur either for a positive $\mathbf{x}$ [in which case $\nabla z(\mathbf{x}^*) = \mathbf{0}$] or it can be on the boundary of the feasible region, where some $x_i^* = 0$. Accordingly, the first-order conditions for this problem can be stated as

$$x_i^* \frac{\partial z(\mathbf{x}^*)}{\partial x_i} = 0 \quad \text{and} \quad \frac{\partial z(\mathbf{x}^*)}{\partial x_i} \geq 0 \qquad \text{for } i = 1, 2, \ldots, I \qquad [2.19]$$

Obviously, the condition $x_i^* \geq 0$ has to hold as well for $i = 1, 2, \ldots, I$.

Linear Equality Constraints

One of the most widely used programming problem formulations is the minimization of a convex function subject to a set of equality constraints, that is,

$$\min z(\mathbf{x}) \qquad [2.20a]$$

subject to

$$\sum_i h_{ij} x_i = b_j; \qquad j = 1, 2, \ldots, J \qquad [2.20b]$$

where h_{ij} is a constant.†

†A set of linear equality constraints will always satisfy the aforementioned constraint qualification condition. Furthermore, such a constraint set will define a convex feasible region.

The Kuhn–Tucker conditions for a stationary point of this program are as follows (see Eqs. [2.14]):

$$\frac{\partial z(\mathbf{x}^*)}{\partial x_i} = \sum_j u_j h_{ij} \qquad \text{for } i = 1, \ldots, I \qquad [2.21a]$$

$$\sum_i h_{ij} x_i^* = b_j \qquad \text{for } j = 1, \ldots, J \qquad [2.21b]$$

At the solution point all the constraints are binding. The complementary slackness conditions are therefore automatically satisfied and the dual variables can have any sign (unlike the case of "greater than or equal to" constraints, in which the dual variables are restricted to be nonnegative).

These first-order conditions can also be derived by the method of Lagrange multipliers. This method involves the specifications of an auxiliary function known as the *Lagrangian.* It includes the original variables $\mathbf{x} = (x_1, \ldots, x_I)$ and the dual variables $\mathbf{u} = (u_1, \ldots, u_J)$, which are also known as Lagrange multipliers. The Lagrangian, $L(\mathbf{x}, \mathbf{u})$, for program [2.20] is given by

$$L(\mathbf{x}, \mathbf{u}) = z(\mathbf{x}) + \sum_j u_j \left[b_j - \sum_i h_{ij} x_i \right] \qquad [2.22]$$

The usefulness of this formulation is that the stationary point of the Lagrangian coincides with the minimum of the constrained optimization [2.20]. Since the Lagrangian is unconstrained, its stationary point can be found by solving for the root of the gradient of the Lagrangian, $\nabla_{\mathbf{x},\mathbf{u}} L(\mathbf{x}, \mathbf{u})$. Note that the gradient is taken with respect to both types of variables, **x** and **u**. This gradient can be naturally broken into its two types of components, and thus the stationary point of Eq. [2.22] is where

$$\nabla_{\mathbf{x}} L(\mathbf{x}^*, \mathbf{u}^*) = \mathbf{0} \qquad [2.23a]$$

and

$$\nabla_{\mathbf{u}} L(\mathbf{x}^*, \mathbf{u}^*) = \mathbf{0} \qquad [2.23b]$$

Condition [2.23a] means that

$$\frac{\partial z(\mathbf{x}^*)}{\partial x_i} - \sum_j u_j^* h_{ij} = 0 \qquad \text{for } i = 1, \ldots, I \qquad [2.24a]$$

and [2.23b] means that

$$b_j - h_j(\mathbf{x}^*) = 0 \qquad \text{for } j = 1, \ldots, J \qquad [2.24b]$$

Thus conditions [2.24] are identical to the Kuhn–Tucker conditions [2.21], with the Lagrangian multipliers $(\ldots, u_j, \ldots)$ playing the same role that the dual variables played in Eqs. [2.21]†. Note that at any feasible point (of the

†The asterisk added to u_j in Eq. [2.24] emphasizes that the u_j's are interpreted here as variables of the Lagrangian. The asterisk denotes the values of these variables at the optimal solution.

original program)

$$L(\mathbf{x}, \mathbf{u}) = z(\mathbf{x})$$

since the added term, $\sum_j u_j[b_j - \sum_i h_{ij} x_i]$, equals zero (each component in the sum is zero). In particular, note that the values of the Lagrangian and the objective function are the same at the minimum point, $\mathbf{x}^*$.

Nonnegativity and Linear Equality Constraints

Many of the problems dealt with in this book include minimization problems with both linear equality constraints and nonnegativity constraints. The general form of these problems is

$$\min z(\mathbf{x}) \qquad [2.25a]$$

subject to

$$\sum_i h_{ij} x_i = b_j; \qquad j = 1, \ldots, J \qquad [2.25b]$$

and

$$x_i \geq 0; \qquad i = 1, \ldots, I \qquad [2.25c]$$

To find the first-order conditions for a minimum for such a problem, the Lagrangian with respect to the equality constraints should be formed first as

$$L(\mathbf{x}, \mathbf{u}) = z(\mathbf{x}) + \sum_j u_j \left[b_j - \sum_i h_{ij} x_i \right] \qquad [2.26a]$$

Then the stationary point of this Lagrangian has to be determined, subject to the constraint

$$x_i \geq 0; \qquad i = 1, \ldots, I \qquad [2.26b]$$

Unlike the case discussed previously, this problem includes nonnegativity constraints. Consequently, the stationary point of program [2.26] has to be determined by the method used to define the first-order condition for such programs, shown in Eqs. [2.19]. For program [2.26] these conditions are the following:

$$x_i^* \frac{\partial L(\mathbf{x}^*, \mathbf{u}^*)}{\partial x_i} = 0 \quad \text{and} \quad \frac{\partial L(\mathbf{x}^*, \mathbf{u}^*)}{\partial x_i} \geq 0 \qquad \forall\, i \qquad [2.27a]$$

$$\frac{\partial L(\mathbf{x}^*, \mathbf{u}^*)}{\partial u_j} = 0 \qquad \forall\, j \qquad [2.27b]$$

Also, $x_i^* \geq 0$, $\forall\, i$, as required by Eq. [2.26b]. Note that in comparison with the first-order conditions for problems with only equality constraints, condition [2.27a] replaces [2.23a] (to account for the nonnegativity of all x_i). Equation [2.27b] requires, simply, that the derivatives of $L(\mathbf{x}, \mathbf{u})$ with respect to $\mathbf{u}$ vanish at the minimum. No other condition is necessary since the values of **u** are not constrained to be nonnegative (or anything else). This condition, then, is identical to [2.23b], specifying the original constraint.

The first-order conditions for the programs with linear equality and nonnegativity constraints can be written explicitly as follows:

$$x_i^* \left(\frac{\partial z(\mathbf{x}^*)}{\partial x_i} - \sum_j u_j^* h_{ij} \right) = 0 \qquad \forall\, i \qquad [2.28a]$$

$$\frac{\partial z(\mathbf{x}^*)}{\partial x_i} - \sum_j u_j h_{ij} \geq 0 \qquad \forall\, i \qquad [2.28b]$$

$$\sum_i h_{ij} x_i^* = b_j \qquad \forall\, j \qquad [2.28c]$$

$$x_i^* \geq 0 \qquad \forall\, i \qquad [2.28d]$$

These conditions are referred to repeatedly throughout the text. The same conditions can be derived also by applying the Kuhn–Tucker conditions [2.14] directly (see Problem 2.11).

More about Lagrangians†

Lagrangians can be used to derive the first-order conditions for general mathematical programs such as

$$\min z(\mathbf{x}) \qquad [2.29a]$$

subject to

$$g_j(\mathbf{x}) \geq b_j \qquad \forall\, j \in \mathcal{J} \qquad [2.29b]$$

This program can include any type of constraints. The Lagrangian for this program is given by‡

$$L(\mathbf{x}, \mathbf{u}) = z(\mathbf{x}) + \sum_j u_j[b_j - g_j(\mathbf{x})] \qquad [2.30]$$

The dual variables in this formulation are restricted to be nonnegative, due to the "greater than or equal to" type of constraints (as in the case discussed in Section 2.2). This distinguishes this formulation from the case of equality constraints in which the dual variables are unrestricted in sign. (See Problem 2.18.)

As it turns out, the stationary point of the Lagrangian of a convex function is not at a minimum or at a maximum of $L(\mathbf{x}, \mathbf{u})$ but rather at a *saddle point* of the Lagrangian. In fact, $L(\mathbf{x}^*, \mathbf{u}^*)$ minimizes $L(\mathbf{x}, \mathbf{u})$ with respect

†This section can be skipped without loss of continuity. The material in here is needed only as a background to the section on estimation of origin–destination matrices in Chapter 13.

‡Note that Eq. [2.22] is a particular case of this Lagrangian.

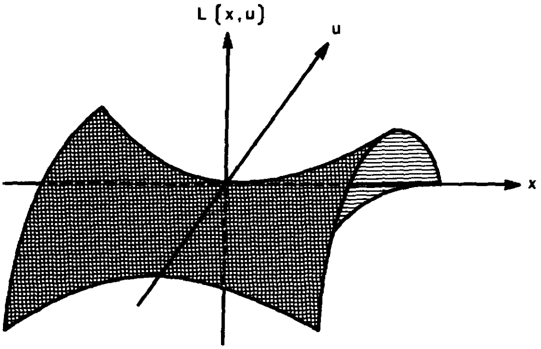


Figure 2.10 Saddle point of a two-argument function. The point (0, 0) maximizes $L(x, u)$ with respect to u and minimizes $L(x, u)$ with respect to x.

to **x** and *maximizes* it with respect to **u**. This condition can be stated as

$$L(\mathbf{x}^*, \mathbf{u}) \leq L(\mathbf{x}^*, \mathbf{u}^*) \leq L(\mathbf{x}, \mathbf{u}^*) \qquad [2.31]$$

Figure 2.10 depicts the shape of saddle point for a function of two variables.

In order to write the first-order conditions of Lagrangian [2.30], note that its minimization is unconstrained with respect to **x**. The maximization with respect to **u**, however, is subject to the nonnegativity constraints. The saddle point of $L(\mathbf{x}, \mathbf{u})$ satisfies, then, the following set of first-order conditions:

$$\frac{\partial L(\mathbf{x}^*, \mathbf{u}^*)}{\partial x_i} = 0 \qquad \forall\, i \qquad [2.32a]$$

$$u_j \frac{\partial L(\mathbf{x}^*, \mathbf{u}^*)}{\partial u_j} = 0 \quad \text{and} \quad \frac{\partial L(\mathbf{x}^*, \mathbf{u}^*)}{\partial u_j} \leq 0 \qquad \forall\, j \qquad [2.32b]$$

In addition, it is required that $u_j \geq 0$, $\forall\, j$. Condition [2.32a] states simply that the gradient vanishes at the stationary point. Conditions [2.32b] parallel condition [2.19] but for a maximum of a function (see also Eq. [2.27a]). Since $L(\mathbf{x}, \mathbf{u})$ has to be maximized with respect to $\mathbf{u} = (u_1, \ldots, u_j)$, the maximum of $L(\mathbf{x}, \mathbf{u})$ with respect to u_j can occur† either at a point where $\partial L(\mathbf{x}, \mathbf{u})/\partial u_j = 0$ or at a point where $u_j = 0$. In the latter case, it must be true that $\partial L(\mathbf{x}, \mathbf{u})/\partial u_j \leq 0$. This observation gives rise to conditions [2.32b]. Conditions [2.32] can be written explicitly as

$$\frac{\partial z(\mathbf{x}^*)}{\partial x_i} - \sum_j u_j^* \frac{\partial g_j(\mathbf{x}^*)}{\partial x_i} = 0 \qquad \forall\, i \qquad [2.33a]$$

$$b_j - g_j(\mathbf{x}^*) \leq 0 \qquad \forall\, j \qquad [2.33b]$$

$$u_j[b_j - g_j(\mathbf{x}^*)] = 0 \qquad \forall\, j \qquad [2.33c]$$

†Assuming, of course, that the maximum exists.

and, by definition,

$$u_j^* \geq 0 \qquad \forall\, j \tag{2.33d}$$

These conditions are identical to the Kuhn–Tucker conditions [2.14].

The Lagrangian approach means that constrained minimization programs can be solved as unconstrained problems of finding the saddle point of the Lagrangian. This point can be found by minimizing the Lagrangian with respect to **x** given **u**, and then maximizing over all values of **u**. This minimization approach is used in Chapter 13. In other parts of the book, the Lagrangian is used only as an aid in the formulation of first-order conditions.

Note that the functional form of the Lagrangian demonstrates why the dual variables can be interpreted as a measure of the sensitivity of the optimal solution to a constraint relaxation, as argued in Section 2.2. At the solution point,

$$L(\mathbf{x}^*, \mathbf{u}^*) = z(\mathbf{x}^*) + \sum_j u_j[b_j - g_j(\mathbf{x}^*)] \tag{2.34}$$

At this point $L(\mathbf{x}^*, \mathbf{u}^*) = z(\mathbf{x}^*)$. If, however, the kth constraint is relaxed by a small amount, Δb_k, and b_k in [2.34] is replaced by $b_k - \Delta b_k$, the new minimum value of $L(\mathbf{x}^*, \mathbf{u}^*)$ will approximately equal the old value (before the relaxation) minus $u_k \, \Delta b_k$. Thus a relaxation of the kth constraint by Δb_k improves the optimal value of the objective function by, approximately, $u_k \, \Delta b_k$.

Linear Programs

A special case of mathematical programming is linear programming (LP). In a linear minimization program, both the objective function and the constraints are linear functions of **x**. A linear program can be written as

$$\min \sum_i c_i x_i \tag{2.35a}$$

subject to

$$\sum_i h_{ij} x_i \geq b_j \qquad \forall\, j \tag{2.35b}$$

where c_i and h_{ij} are constants and the summations go from $i = 1$ to $i = I$.

The widespread use of linear programming problems stems from the relative ease of solving them. Large linear programming problems can be solved very efficiently by using modern computers. The ease of solving linear programs results from the fact that their solutions never lie at an internal point but always at the boundary of the feasible region. Furthermore, if a solution exists, it will always be at a "corner" of the feasible region. Thus, instead of searching for a minimum all over the feasible region, only the corner points of this region have to be checked. This property is intuitively apparent in Figure 2.11a, which illustrates a feasible region and the contour lines of an objective function for a linear program in two variables, x_1 and x_2. The simplex method

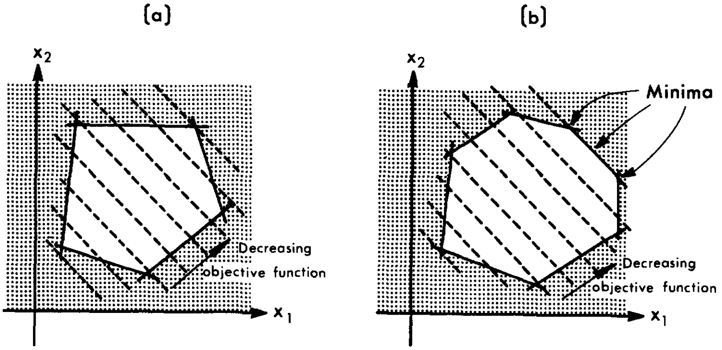


Figure 2.11 Linear programs in two variables: (a) the solution is at a corner of the feasible region; (b) multiple minima.

for solving linear programs exploits this property by progressing through adjacent corners of the feasible region.

In some cases, multiple minima [all with the same value of $z(x^*)$] may exist (since the "strict convexity" condition does not apply to linear programs), as illustrated in Figure 2.11b. Some of these minima, however, will always be at the intersection of several constraints (in other words, at the corners of the feasible region). The minimum of $z(\mathbf{x})$ thus, can still be determined even if only the corners of the feasible region are searched.

2.4 SUMMARY

Chapter 2 reviews the necessary and sufficient conditions for a minimum of a constrained minimization program. The necessary (first-order) conditions are formulated in terms of a set of auxiliary variables known as the dual variables. Each dual variable is associated with a constraint and its value at the solution point indicates the sensitivity of the solution to a small relaxation or tightening of that constraint. Consequently, it is zero for nonbinding constraints and nonnegative for the binding constraints. If the program is unconstrained, these first-order conditions (known as the Kuhn–Tucker conditions) reduce to the requirement that the gradient vanishes. These conditions alone, however, cannot be used to identify the minimum of any program because they hold at any stationary point of the objective function and not only at minima.

To make sure that a stationary point, $\mathbf{x}^*$, is the minimum of some function $z(\mathbf{x})$, two requirements have to be fulfilled: (1) $\mathbf{x}^*$ should be a local minimum, and (2) $\mathbf{x}^*$ should be either the lowest of all the local minima or the only one. In order to meet the first requirement, it is sufficient to show that $z(\mathbf{x})$ is strictly convex in the vicinity of $\mathbf{x}^*$. The second requirement is difficult to meet if there exist multiple minima. It is sufficient, however, to show that $z(\mathbf{x})$ is convex in order to ensure that a local minimum is unique (i.e., that no other local maxima exist). If a program is constrained, it is also required that the constraint set define a convex feasible region in order to ensure uniqueness. Convexity (or strict convexity) is a second-order condition that can be established with the help of several criteria. The criterion used most often in this book applies when the objective function is twice differentiable; in this case strict convexity can be established by determining that the Hessian of the objective function is positive definite. This condition can easily be checked in cases in which the Hessian is diagonal; if all the entries are positive, the Hessian is positive definite. This implies that the objective function is strictly convex. If the diagonal entries are nonnegative, the Hessian is positive semidefinite, meaning that the objective function is convex.

Two special types of mathematical programs are highlighted in Section 2.3. The first one is a program with only equality and nonnegativity constraints. This type of program is used throughout the book and thus its first-order conditions are derived explicitly. The constraint set of such a program is always convex, meaning that only the objective function has to be checked in order to establish the uniqueness of a minimum.

The first-order conditions for this type of programs, as well as for general maximization programs can be derived through the use of the Lagrangian, which is a function of the original variables and the dual variables of the program. The Lagrangian approach is used repeatedly throughout this book.

The second type of program highlighted is a linear program where both the objective function and the constraints are linear. This type of program is particularly easy to solve and, in fact, the solution techniques to many nonlinear programs consist of repeated solutions of a related linear program.

2.5 ANNOTATED REFERENCES

The material covered in this chapter can be found in any standard mathematical programming text. At the elementary level the reader can consult Chapters 14 and 15 of Wagner (1975), or the texts by Simmons (1975), Bradley et al. (1977), or Wismer and Chattergy (1978). A somewhat more advanced treatment is offered by Zangwill (1969) and Luenberger (1973).

2.6 PROBLEMS

2.1. Show that the first derivative of a continuously differentiable function, $z(x)$, is zero at a local minimum.

*2.2. Show that if a function is continuously differentiable, it is strictly convex if and only if a linear approximation to the function always underestimates it. (*Hint:* Show that if condition [2.4a] holds, then [2.4b] is true, and if [2.4b] holds, then [2.4a] is true.)

*Problems marked with an asterisk require a somewhat higher level of mathematics than the rest.

*2.3. Show that if a function is twice continuously differentiable, it is strictly convex if and only if the second derivative of the function is positive. (*Hint:* Show that if condition [2.4a] holds, then [2.4c] is true, and if [2.4c] holds, then [2.4a] is true.)

2.4. In which of the following cases does the program min $z(\mathbf{x})$ subject to $g_j(\mathbf{x}) \geq b_j$ with $j = 1, \ldots, J$ have a unique minimum? Explain your answers.

(a) $z(x)$ is convex and the constraint set is convex.

(b) $z(x)$ is ditonic and unconstrained.

(c) $z(x)$ is monotonically increasing and unconstrained.

(d) $z(x)$ is concave and the constraint set is convex.

(e) $z(x)$ is piecewise continuous over a convex constraint set.

2.5. Consider the program min $z(x) = (x - 2)^2$ subject to $0 \leq x \leq 1$.

(a) Write this program in standard form.

(b) Show that the first-order conditions [2.7] hold at the point that minimizes $z(x)$ in the feasible region.

2.6. Plot the contours (in the feasible region only) of a two-dimensional function $z(x_1, x_2)$ subject to $0 \leq x_1 \leq a$, $0 \leq x_2 \leq b$ under the assumption that the function is:

(a) Strictly convex.

(b) Convex with multiple minima.

(c) Linear.

(d) Constrained by $x_1 + x_2 = c$ (in addition to the previous constraints).

*2.7. Show that the gradient always points in the direction of the steepest (local) increase in $z(\mathbf{x})$.

2.8. Calculate the gradient of $z(x_1, x_2) = 2(x_1 - 3)^2 + 3(x_2 - 2)^2$ at $(x_1, x_2) = (2, 3)$ and at $(x_1, x_2) = (3, 2)$.

2.9. Show that the sum of convex functions is a convex function.

2.10. The value of the dual variable used in solving Eqs. [2.15] is $u^* = 2$. Relax the constraint by a small amount and solve the program again. Show that u^* measures the sensitivity of $z(\mathbf{x}^*)$ to the binding constraint.

2.11. **(a)** Using the Kuhn–Tucker conditions, show that Eq. [2.19] holds if the only constraints are nonnegativity constraints.

(b) Derive Eqs. [2.28] directly from the Kuhn–Tucker conditions (Eqs. [2.14]).

2.12. Using the Lagrangian of $z(\mathbf{x})$ subject to $g_j(\mathbf{x}) \geq b_j$ for $j = 1, \ldots, J$, demonstrate why a relaxation of the jth constraint by a small amount, Δb_j, would decrease $z(\mathbf{x}^*)$ by $u_j \, \Delta b_j$.

2.13. Using the Kuhn–Tucker conditions, solve the problem

$$\min z(x_1, x_2) = 4(x_1 - 2)^2 + 3(x_2 - 4)^2$$

subject to

$$x_1 + x_2 \geq 5$$
$$x_1 \geq 1$$
$$x_2 \geq 2$$

Show that the solution is unique.

2.14. Solve the program

$$\min z(x_1, x_2) = 5x_1^2 - 3x_1x_2 + 2x_2^2 + 7$$

subject to

$$2x_1 + x_2 = 5$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$

using Lagrangians. Plot the feasible region.

2.15. Solve the program

$$\min z(x_1, x_2) = 2x_1 + x_2$$

subject to

$$3x_1 + x_2 \geq 1$$

$$x_1 + 4x_2 \geq 2$$

$$-x + x_2 \geq 1$$

Plot the feasible region and the contours of the objective function.

2.16. Given the function

$$z(x_1, x_2, x_3) = 3x_1^2 + 2x_1x_2 + 6x_2^2 - x_2x_3 + 5x_3^2 - 4x_1 - 2x_2 - 6x_3 + 9$$

(a) Find the minimum of $z(x_1, x_2, x_3)$.
(b) Show that it is a local minimum.
(c) Show that this local minimum is a global minimum.

2.17. Given that the dual variables indicate the improvement in the objective function value, associated with the relaxation of any constraint, argue why $u_j \geq 0$, $\forall\, j$, for a minimization program with standard inequality constraints and why u_j can have any sign for equality constraints.

2.18. Derive the Kuhn–Tucker conditions for the program

$$\min z(\mathbf{x})$$

subject to

$$\sum_i h_{ij} x_i = b_j \qquad \forall\, j$$

without using Lagrangians. (*Hint:* Express the constraints in standard form.) Show that the dual variables associated with equality constraints are unrestricted in sign.

3

Formulating the Assignment Problem as a Mathematical Program

The traffic assignment or the transportation network equilibrium problem was defined in Chapter 1. As mentioned there, the basic problem is to find the link flows given the origin–destination trip rates, the network, and the link performance functions (see Section 1.3). The solution of the problem is based on the behavioral assumption that each motorist travels on the path that minimizes the travel time† from origin to destination. This choice rule implies that at equilibrium the link-flow pattern is such that the travel times on all used paths connecting any given O–D pair will be equal; the travel time on all of these used paths will also be less than or equal to the travel time on any of the unused paths. At this point, the network is in user equilibrium; no motorist can experience a lower travel time by unilaterally changing routes.

Section 1.3 demonstrated how the user-equilibrium flow pattern can be found for a small network by using graphical methods. Unfortunately, such methods cannot be used to solve for equilibrium over networks with a large number of nodes, links, and O–D pairs. The approach described in this text for solving large problems uses the equivalent minimization method. This approach involves the formulation of a mathematical program, the solution of which is the user-equilibrium flow pattern. This general approach is used often in operations research, in cases in which it is easier to minimize the equivalent program than to solve a set of conditions directly.

For the minimization formulation to be useful, the equivalent mathemat-

†As mentioned in Section 1.3, the term "travel time" can be understood to represent general travel impedance, combining elements of travel costs and other components of travel disutility.

ical program has to have a unique solution, which also satisfies the equilibrium conditions. Furthermore, the program has to be relatively easy to solve. The focus of this chapter is on the formulation of the equivalent minimization program corresponding to the equilibrium traffic assignment problem and on the properties of this program. Solution procedures are then described in Chapters 4 and 5. This chapter is organized as follows: Section 3.1 presents the equivalent minimization formulation, Section 3.2 shows that its solution satisfies the equilibrium conditions, and Section 3.3 proves that this solution is unique. Sections 3.4 and 3.5 explore the nature of the minimization program and the user-equilibrium flow pattern by comparing them to a related flow pattern over the network.

Before the basic formulation is discussed, the following paragraphs present the network notations used in this chapter and throughout this text. The network itself is represented by a directed graph that includes a set of consecutively numbered nodes, $\mathscr{N}$, and a set of consecutively numbered arcs (links), $\mathscr{A}$. In some cases it will be useful to refer to links by their end nodes (i.e., link $m \rightarrow n$ leading from node m to node n), especially in discussing certain algorithms. In these cases the notations will be clarified before the discussion. Let $\mathscr{R}$ denote the set of origin centroids (which are the nodes at which flows are generated) and let $\mathscr{S}$ denote the set of destination centroids (which are the nodes at which flows terminate). The origin node set and the destination node set are not mutually exclusive since nodes can serve as origins and destinations of different trips at the same time (i.e., $\mathscr{R} \cap \mathscr{S} \neq \varnothing$). Each O–D pair r–s is connected by a set of paths (routes) through the network. This set is denoted by $\mathscr{K}_{rs}$ where $r \in \mathscr{R}$ and $s \in \mathscr{S}$.

The origin–destination matrix is denoted by $\mathbf{q}$ with entries q_{rs}. In other words, q_{rs} is the trip rate between origin r and destination s during the period of analysis. Let x_a and t_a represent the flow and travel time, respectively, on link a (where $a \in \mathscr{A}$). Furthermore, $t_a = t_a(x_a)$, where $t_a(\cdot)$ represents the relationship between flow and travel time for link a. In other words, $t_a(x_a)$ is the link performance function (which is also known as the volume–delay curve or the link congestion function). Similarly, let f_k^{rs} and c_k^{rs} represent the flow and travel time, respectively, on path k connecting origin r and destination s ($k \in \mathscr{K}_{rs}$). The travel time on a particular path is the sum of the travel time on the links comprising this path. This relationship can be expressed mathematically as

$$c_k^{rs} = \sum_a t_a \delta_{a,k}^{rs} \qquad \forall\, k \in \mathscr{K}_{rs}, \quad \forall\, r \in \mathscr{R}, \quad \forall\, s \in \mathscr{S} \qquad [3.1a]$$

where $\delta_{a,k}^{rs} = 1$ if link a is a part of path k connecting O–D pair r–s, and $\delta_{a,k}^{rs} = 0$ otherwise. Using the same indicator variable, the link flow can be expressed as a function of the path flow, that is,

$$x_a = \sum_r \sum_s \sum_k f_k^{rs} \delta_{a,k}^{rs} \qquad \forall\, a \in \mathscr{A} \qquad [3.1b]$$

This equation means that the flow on each arc is the sum of the flows on all

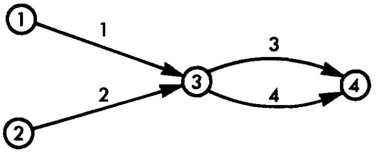


Figure 3.1 Network example with two O–D pairs and four links.

paths going through that arc. Equations [3.1] are known as the *path–arc incidence relationships.*

As an example of the use of the incidence relationships, consider the simple network shown in Figure 3.1. It includes two O–D pairs: 1–4 and 2–4 (node 3 is neither an origin nor a destination point). The link numbers are written on the links. Assume now that the first path from origin node 1 to destination node 4 uses links 1 and 3 and the second one uses links 1 and 4. Similarly, assume that the first path from origin node 2 to destination node 4 uses links 2 and 3 and the second one uses links 2 and 4. For example, $\delta^{14}_{1,1} = 1$ (since link 1 is on path 1 from node 1 to node 4), but $\delta^{24}_{3,2} = 0$ (since link 3 is not on the second path from node 2 to node 4). The incidence relationships for this network means that, for example,

$$\begin{aligned} c^{14}_1 &= t_1\delta^{14}_{1,1} + t_2\delta^{14}_{2,1} + t_3\delta^{14}_{3,1} + t_4\delta^{14}_{4,1} \\ &= t_1 + t_3 \end{aligned}$$

Thus Eq. [3.1a] reduces to the anticipated result that the travel time on path 1 between origin 1 and destination 4 is the sum of the travel times on the links comprising this path. Similarly,

$$\begin{aligned} x_3 &= f^{14}_1\delta^{14}_{3,1} + f^{14}_2\delta^{14}_{3,2} + f^{24}_1\delta^{24}_{3,1} + f^{24}_2\delta^{24}_{3,2} \\ &= f^{14}_1 + f^{24}_1 \end{aligned}$$

Again, Eq. [3.1b] gives the anticipated result here—that the flow on a particular link is the sum of the path flows traversing this link.

Many of the presentations appearing later in the text can be simplified by using vector notation. In most cases this notation is used only to shorten some mathematical expressions [as was the case in Chapter 2, where $z(x_1, \ldots, x_I)$ was denoted by $z(\mathbf{x})$]. Using vector notation, then, let $\mathbf{x} = (\ldots, x_a, \ldots)$, $\mathbf{t} = (\ldots, t_a, \ldots)$, $\mathbf{f}^{rs} = (\ldots, f^{rs}_k, \ldots)$, $\mathbf{f} = (\ldots, \mathbf{f}^{rs}, \ldots)$, $\mathbf{c}^{rs} = (\ldots, c^{rs}_k, \ldots)$, and $\mathbf{c} = (\ldots, \mathbf{c}^{rs}, \ldots)$. Furthermore, let $\boldsymbol{\Delta}$ be the link–path incidence matrix with elements $\delta^{rs}_{a,k}$. Typically, this matrix is arranged in block form by O–D pair. In other words, $\boldsymbol{\Delta} = (\ldots, \boldsymbol{\Delta}^{rs}, \ldots)$, where $\boldsymbol{\Delta}^{rs}$ is the link–path incidence matrix for O–D pair r–s. The number of rows in the incidence matrix, $\boldsymbol{\Delta}^{rs}$, equals the number of links in the network and the number of columns in this matrix is the number of paths between origin r and destination s. The element in the ath row and kth column of $\boldsymbol{\Delta}^{rs}$ is $\delta^{rs}_{a,k}$, in other words $(\boldsymbol{\Delta}^{rs})_{a,k} = \delta^{rs}_{a,k}$. Using vector operations, the incidence relationships can now be written in matrix notation as

$$\mathbf{c} = \mathbf{t} \cdot \boldsymbol{\Delta} \qquad \text{and} \qquad \mathbf{x} = \mathbf{f} \cdot \boldsymbol{\Delta}^T \qquad [3.2]$$

TABLE 3.1 **Basic Network Notation**

$\mathscr{N}$,	node (index) set
$\mathscr{A}$,	arc (index) set
$\mathscr{R}$,	set of origin nodes; $\mathscr{R} \subseteq \mathscr{N}$
$\mathscr{S}$,	set of destination nodes; $\mathscr{S} \subseteq \mathscr{N}$
$\mathscr{K}_{rs}$,	set of paths connecting O–D pair r–s; $r \in \mathscr{R}$, $s \in \mathscr{S}$
x_a,	flow on arc a; $\mathbf{x} = (\ldots, x_a, \ldots)$
t_a,	travel time on arc a; $\mathbf{t} = (\ldots, t_a, \ldots)$
f_k^{rs},	flow on path k connecting O–D pair r–s; $\mathbf{f}^{rs} = (\ldots, f_k^{rs}, \ldots)$; $\mathbf{f} = (\ldots, \mathbf{f}^{rs}, \ldots)$
c_k^{rs},	travel time on path k connecting O–D pair r–s; $\mathbf{c}^{rs} = (\ldots, c_k^{rs}, \ldots)$; $\mathbf{c} = (\ldots, \mathbf{c}^{rs}, \ldots)$
q_{rs},	trip rate between origin r and destination s; $(\mathbf{q})_{rs} = q_{rs}$
$\delta_{a,k}^{rs}$,	indicator variable: $\delta_{a,k}^{rs} = \begin{cases} 1 & \text{if link } a \text{ is on path } k \text{ between O–D pair } r\text{–}s \\ 0 & \text{otherwise} \end{cases}$ $(\boldsymbol{\Delta}^{rs})_{a,k} = \delta_{a,k}^{rs}$; $\boldsymbol{\Delta} = (\ldots, \boldsymbol{\Delta}^{rs}, \ldots)$

The incidence matrix for the network example depicted in Figure 3.1 can be written as follows:

$$
\begin{array}{cc}
 & \begin{array}{cc} \text{O–D} & \text{O–D} \\ 1\text{–}4 & 2\text{–}4 \end{array} \\
\begin{array}{r} \text{link} \diagdown \text{path} \end{array} & \overbrace{1 \quad 2} \quad \overbrace{1 \quad 2} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}
\end{array}
$$

As the reader can check, Eq. [3.2] is identical to Eqs. [3.1] in terms of specifying the relationships between flows and travel times for the network example shown in Figure 3.1.

The network notation introduced here is summarized in Table 3.1. Further notation is introduced as needed.

3.1 THE BASIC TRANSFORMATION

The equilibrium assignment problem is to find the link flows, $\mathbf{x}$, that satisfy the user-equilibrium criterion when all the origin–destination entries, $\mathbf{q}$, have been appropriately assigned. This link–flow pattern can be obtained by solving the following mathematical program:

$$\min z(\mathbf{x}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega \tag{3.3a}$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\, r, s \qquad [3.3b]$$

$$f_k^{rs} \geq 0 \qquad \forall\, k, r, s \qquad [3.3c]$$

The definitional constraints

$$x_a = \sum_r \sum_s \sum_k f_k^{rs} \delta_{a,k}^{rs} \qquad \forall\, a \qquad [3.3d]$$

are also part of this program.

In this formulation, the objective function is the sum of the integrals of the link performance functions. This function does not have any intuitive economic or behavioral interpretation. It should be viewed strictly as a mathematical construct that is utilized to solve equilibrium problems.

Equation [3.3b] represents a set of flow conservation constraints. These constraints state that the flow on all paths connecting each O–D pair has to equal the O–D trip rate. In other words, all O–D trip rates have to be assigned to the network. The nonnegativity conditions in Eq. [3.3c] are required to ensure that the solution of the program will be physically meaningful.

The objective function of program [3.3], $z(\mathbf{x})$, is formulated in terms of link flows, whereas the flow conservation constraints are formulated in terms of path flows. The network structure enters this formulation through the definitional incidence relationships [3.3d]. These incidence relationships express the link flows in terms of the path flows [i.e., $\mathbf{x} = \mathbf{x}(\mathbf{f})$]. The incidence relationships also mean that the partial derivative of link flow can be defined with respect to a particular path flow.† In other words,

$$\frac{\partial x_a(\mathbf{f})}{\partial f_l^{mn}} = \frac{\partial}{\partial f_l^{mn}} \sum_r \sum_s \sum_k f_k^{rs} \delta_{a,k}^{rs} = \delta_{a,l}^{mn} \qquad [3.4]$$

since $\partial f_k^{rs}/\partial f_l^{mn} = 0$ if r–$s \neq m$–n or $k \neq l$. Equation [3.4] implies that the derivative of the flow on link a with respect to the flow on path l between origin m and destination n equals 1 if the link is a part of that path and zero otherwise. These relationships are used later in the text to investigate the first- and second-order conditions of program [3.3].

It is important to note that this formulation assumes that the travel time on a given link is a function of the flow on that link only and not of the flow on any other link in the network. This somewhat restrictive assumption is discussed in detail and subsequently relaxed in Chapter 8. In addition, the link performance functions are assumed to be positive and increasing. These latter

†The function $x_a(\mathbf{f})$ includes flow summation using the subscripts r, s, and k. To avoid confusion in differentiation, the variable with respect to which the derivative is being taken is subscripted by m, n, and l. Thus f_l^{mn} is the flow on path l between origin m and destination n. Similarly, x_b is used below when the derivative of an expression including a sum over link flows is taken with respect to the flow on a particular link.

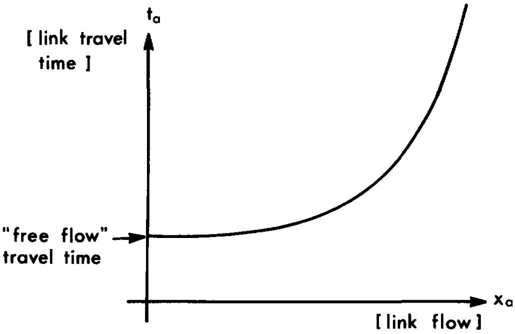


Figure 3.2 Typical link performance function, $t_a(x_a)$.

assumptions are not restrictive in the sense that the congestion effects described by these functions exhibit both characteristics, as mentioned in Section 1.2 (note that these curves are also convex). A typical performance curve is depicted in Figure 3.2 (see also Figure 1.8). The assumptions on the performance curves can be written mathematically as

$$\frac{\partial t_a(x_a)}{\partial x_b} = 0 \qquad \forall\, a \neq b \tag{3.5a}$$

$$\frac{\partial t_a(x_a)}{\partial x_a} > 0 \qquad \forall\, a \tag{3.5b}$$

The problem formulation represented by Eqs. [3.3] is known as Beckmann's transformation. It has been evident in the transportation literature since the mid-1950s, but its usefulness became apparent only when solution algorithms for this program were developed in the late 1960s and early 1970s. Some of these algorithms are discussed in Chapter 5.

The following section formally proves that the solution to Beckmann's transformation satisfies the user-equilibrium conditions. This equivalency is first illustrated, however, for a simple situation.

Consider the network depicted in Figure 3.3. This network includes two paths (which are also links), leading from the origin, O, to the destination, D. The volume–delay curves for the two links are given by

$$t_1 = 2 + x_1 \tag{3.6a}$$

$$t_2 = 1 + 2x_2 \tag{3.6b}$$

The O–D flow, q, is 5 units of flow, that is,

$$x_1 + x_2 = 5 \tag{3.6c}$$

The equilibrium condition for this example can be expressed as (see Section 1.3)

$$t_1 \leq t_2 \quad \text{if } x_1 > 0 \qquad \text{and} \qquad t_1 \geq t_2 \quad \text{if } x_2 > 0 \tag{3.6d}$$

For this example it can be verified by inspection that both paths will be used

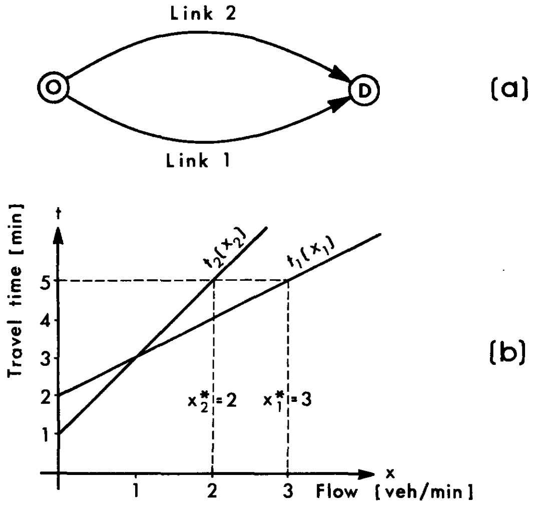


Figure 3.3 Equilibrium example: (a) a two-link network; (b) the performance functions and the equilibrium solution.

at equilibrium and the last equation can therefore be written simply (given that $x_1 > 0$ and $x_2 > 0$) as

$$t_1 = t_2 \qquad [3.6e]$$

The equilibrium problem then, is to solve four equations (the two volume-delay curves [3.6a] and [3.6b], the flow conservation condition [3.6c], and the user-equilibrium condition [3.6e]), in four unknowns: x_1, x_2, t_1, and t_2. The solution to this set of equations is

$$x_1 = 3 \quad \text{flow units}$$

$$x_2 = 2 \quad \text{flow units}$$

$$t_1 = t_2 = 5 \quad \text{time units}$$

When the problem is formulated as a minimization program, the result is the following:

$$\min z(\mathbf{x}) = \int_0^{x_1} (2 + \omega)\, d\omega + \int_0^{x_2} (1 + 2\omega)\, d\omega$$

subject to

$$x_1 + x_2 = 5$$

$$x_1, x_2 \geq 0$$

To set the problem up as a simple one-dimensional unconstrained minimization, $x_2 = 5 - x_1$ can be substituted into the objective function and into the remaining (nonnegativity) constraints to get the problem

$$\min z(x_1) = \int_0^{x_1} (2 + \omega)\, d\omega + \int_0^{5 - x_1} (1 + 2\omega)\, d\omega \qquad [3.7a]$$

subject to

$$x_1 \geq 0 \quad \text{and} \quad 5 - x_1 \geq 0 \qquad [3.7b]$$

To solve this program, the constraints can be relaxed and the objective function can be minimized as an unconstrained program. If the solution satisfies the constraints, it is valid for the constrained program as well. Carrying out the integration and collecting similar terms, the objective function becomes

$$z(x_1) = 1.5x_1^2 - 9x_1 + 30$$

This function attains its minimum at $x_1^* = 3$, where $dz(x_1)/dx_1 = 0$. This solution satisfies the two constraints in Eq. [3.7b] and is therefore a minimum of the constrained program [3.7] as well. The original flow conservation constraint guarantees that $x_2^* = 2$ and indeed, the solution of the mathematical program is identical to the solution of the equilibrium equations. This equivalency is demonstrated for the general case in the next section.

3.2 EQUIVALENCY CONDITIONS

To demonstrate the equivalence of the equilibrium assignment problem and program [3.3], it has to be shown that any flow pattern that solves [3.3] also satisfies the equilibrium conditions. This equivalency is demonstrated in this section by proving that the first-order conditions for the minimization program are identical to the equilibrium conditions. From the discussion in Chapter 2, recall that the solution of any mathematical program satisfies its first-order conditions at any local minimum or any stationary point of the program. If the first-order conditions are identical to the equilibrium conditions, the latter hold at any local minimum (or stationary point). Thus, by finding a minimum point of the program, an equilibrium flow pattern is obtained.

To derive the first-order conditions of the Beckmann transformation, observe that it is a minimization problem with linear equality and nonnegativity constraints. A general form of the first-order conditions for such problems was derived in Section 2.3 (see Eqs. [2.25] and the related discussion). Following that section, the Lagrangian of the equivalent minimization problem with respect to the equality constraints [3.3b] can be formulated as†

$$L(\mathbf{f}, \mathbf{u}) = z[\mathbf{x}(\mathbf{f})] + \sum_{rs} u_{rs}\left(q_{rs} - \sum_k f_k^{rs}\right) \qquad [3.8a]$$

where u_{rs} denotes the dual variable associated with the flow conservation constraint for O–D pair r–s in Eq. [3.3b]. The first-order conditions of program [3.3] are equivalent to the first-order conditions of Lagrangian [3.8a],

†The shorthand notation "$\sum_{rs}$" is used throughout the book to abbreviate "$\sum_r \sum_s$."

given that $L(\mathbf{f}, \mathbf{u})$ has to be minimized with respect to nonnegative path flows, that is,

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \tag{3.8b}$$

The formulation of this Lagrangian is given in terms of path flow, by using the incidence relationships, $x_a = x_a(\mathbf{f})$ in Eq. [3.3d], for every link a. At the stationary point of the Lagrangian, the following conditions have to hold with respect to the path-flow variables:

$$f_k^{rs} \frac{\partial L(\mathbf{f}, \mathbf{u})}{\partial f_k^{rs}} = 0 \quad \text{and} \quad \frac{\partial L(\mathbf{f}, \mathbf{u})}{\partial f_k^{rs}} \geq 0 \qquad \forall\ k, r, s \tag{3.9a}$$

and the following conditions have to hold with respect to the dual variables:

$$\frac{\partial L(\mathbf{f}, \mathbf{u})}{\partial u_{rs}} = 0 \qquad \forall\ r, s \tag{3.9b}$$

Also, the nonnegativity constraints have to hold (i.e., $f_k^{rs} \geq 0$, $\forall\ k, r, s$). The asterisks have been omitted from the last expressions (in comparison with Eqs. [2.27]) for clarity of notation. This practice is followed hereafter in this book.

Condition [3.9b] simply states the flow conservation constraints, which obviously, have to hold at equilibrium. The first-order conditions expressed in Eq. [3.9a] can be obtained explicitly by calculating the partial derivatives of $L(\mathbf{f}, \mathbf{u})$ with respect to the flow variables, f_l^{mn}, and substituting the result into [3.9a]. This derivative is given by

$$\frac{\partial}{\partial f_l^{mn}} L(\mathbf{f}, \mathbf{u}) = \frac{\partial}{\partial f_l^{mn}} z[\mathbf{x}(\mathbf{f})] + \frac{\partial}{\partial f_l^{mn}} \sum_{rs} u_{rs}\left(q_{rs} - \sum_k f_k^{rs}\right) \tag{3.10}$$

The right-hand side of Eq. [3.10] consists of the sum of two types of terms, the first of which is the derivative of the objective function and the second is the derivative of a term involving the constraints. These two types of terms are each calculated separately.

The first term on the right-hand side of Eq. [3.10] is the derivative of the objective function [3.3a] with respect to f_l^{mn}. This derivative can be evaluated by using the chain rule:

$$\frac{\partial z[\mathbf{x}(\mathbf{f})]}{\partial f_l^{mn}} = \sum_{b \in \mathcal{A}} \frac{\partial z(\mathbf{x})}{\partial x_b} \frac{\partial x_b}{\partial f_l^{mn}} \tag{3.11}$$

Each term in the sum on the right-hand side of this equation is the product of two quantities. The first quantity is $\partial z(\mathbf{x})/\partial x_b$, which can be easily calculated since the travel time on any link is a function of the flow on that link only. Hence

$$\frac{\partial z(\mathbf{x})}{\partial x_b} = \frac{\partial}{\partial x_b} \sum_a \int_0^{x_a} t_a(\omega)\, d\omega = t_b \tag{3.12a}$$

The second quantity in the product is the partial derivative of a link flow with respect to the flow on a particular path. As shown in Section 3.1 (see Eq. [3.4]),

$$\frac{\partial x_b}{\partial f_l^{mn}} = \delta_{b,l}^{mn} \qquad [3.12b]$$

Substituting the last two expressions into Eq. [3.11], the derivative of the objective function with respect to the flow on a particular path becomes

$$\frac{\partial z[\mathbf{x}(\mathbf{f})]}{\partial f_l^{mn}} = \sum_b t_b \delta_{b,l}^{mn} = c_l^{mn} \qquad [3.13]$$

In other words, it is the travel time on that particular path.

The second type of term in Eq. [3.10] is even simpler to calculate since

$$\frac{\partial f_k^{rs}}{\partial f_l^{mn}} = \begin{cases} 1 & \text{if } r = m,\ s = n \text{ and } k = l \\ 0 & \text{otherwise} \end{cases}$$

Thus (since q_{rs} is a constant and u_{rs} is not a function of f_l^{mn}) this term becomes

$$\frac{\partial}{\partial f_l^{mn}} \sum_{rs} u_{rs}\left(q_{rs} - \sum_k f_k^{rs}\right) = -u_{mn} \qquad [3.14]$$

Substituting both [3.13] and [3.14] into Eq. [3.10], the partial derivative of the Lagrangian becomes

$$\frac{\partial}{\partial f_l^{mn}} L(\mathbf{f}, \mathbf{u}) = c_l^{mn} - u_{mn} \qquad [3.15]$$

The general first-order conditions (Eqs. [3.9]) for the minimization program in Eqs. [3.3] can now be expressed explicitly as

$$f_k^{rs}(c_k^{rs} - u_{rs}) = 0 \qquad \forall\ k, r, s \qquad [3.16a]$$

$$c_k^{rs} - u_{rs} \geq 0 \qquad \forall\ k, r, s \qquad [3.16b]$$

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\ r, s \qquad [3.16c]$$

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [3.16d]$$

Conditions [3.16c] and [3.16d] are simply the flow conservation and nonnegativity constraints, respectively. These constraints, naturally, hold at the point that minimizes the objective function. The following discussion is focused on the nature of the first two conditions expressed in Eqs. [3.16a] and [3.16b]. These conditions hold for each path between any O–D pair in the network. For a given path, say path k connecting origin r and destination s, the conditions hold for two possible combinations of path flow and travel time. Either the flow on that path is zero (i.e., $f_k^{rs} = 0$ and Eq. [3.16a] holds), in which case the travel time on this path, c_k^{rs}, must be greater than or equal to the O–D specific Lagrange multiplier, u_{rs} (as required by Eq. [3.16b]); or the flow on the kth path is positive, in which case $c_k^{rs} = u_{rs}$ and both Eqs. [3.16a] and [3.16b] hold as equalities.† In any event, the Lagrange multiplier of a given O–D pair is less than or equal to the travel time on all paths connecting this pair. Thus u_{rs} equals the minimum path travel time between origin r and destination s.

With this interpretation, it is now clear that Eqs. [3.16], in fact, state the user-equilibrium principle. The paths connecting any O–D pair can be divided into two categories: those carrying flow, on which the travel time equals the minimum O–D travel time; and those not carrying flow, on which the travel time is greater than (or equal to) the minimum O–D travel time. If the flow pattern satisfies these equations, no motorist can be better off by unilaterally changing routes. All other routes have either equal or higher travel times. The user-equilibrium criteria are thus met for every O–D pair.

The equivalence between the UE conditions and the first-order conditions of program [3.3] means that the UE conditions are satisfied at any local minimum or, in fact, at any stationary point of this program. Accordingly, this program is usually referred to as the UE program or the UE equivalent minimization. The next section shows that the UE program has only one stationary point, which is a minimum.

3.3 UNIQUENESS CONDITIONS

In order to show that the UE equivalent minimization program has only one solution, it is sufficient to prove that objective function [3.3a] is strictly convex in the vicinity of $\mathbf{x}^*$ (and convex elsewhere) and that the feasible region (defined by constraints [3.3b] and [3.3c]) is convex. The convexity of the feasible region is assured for linear equality constraints, as mentioned in Section 2.3. The addition of the nonnegativity constraints does not alter this characteristic. The focus of this section, then, is on the properties of the objective function.

The convexity of the objective function is proved here with respect to link flows; path flows are treated later. This section demonstrates that the function

$$z(\mathbf{x}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega$$

is convex under the aforementioned assumptions on the link performance functions (namely, that $\partial t_a(\cdot)/\partial x_b = 0$ for $a \neq b$ and $dt_a(x_a)/dx_a > 0$, $\forall\ a$—see Eqs. [3.5]). This is done by proving that the Hessian [the matrix of the second derivatives of $z(\mathbf{x})$ with respect to $\mathbf{x}$] is positive definite, thus ensuring that $z(\mathbf{x})$ is, in fact, strictly convex everywhere.

The Hessian is calculated by using a representative term of the matrix.

†All this holds at the minimum point (the solution of the minimization program); recall that the asterisks are omitted from Eqs. [3.16] for presentation purposes only.

The derivative of $z(\mathbf{x})$ is therefore taken with respect to the flow on the mth and nth links. First,

$$\frac{\partial z(\mathbf{x})}{\partial x_m} = t_m(x_m)$$

as in Eq. [3.12a], and second,

$$\frac{\partial^2 z(\mathbf{x})}{\partial x_m \, \partial x_n} = \frac{\partial t_m(x_m)}{\partial x_n} = \begin{cases} \dfrac{dt_n(x_n)}{dx_n} & \text{for } m = n \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

because of condition [3.5a]. This means that all the off-diagonal elements of the Hessian, $\nabla^2 z(\mathbf{x})$, are zero and all the diagonal elements are given by $dt_a(x_a)/dx_a$. In other words,†

$$\nabla^2 z(\mathbf{x}) = \begin{bmatrix} \dfrac{dt_1(x_1)}{dx_1} & 0 & 0 & \cdots \\ 0 & \dfrac{dt_2(x_2)}{dx_2} & 0 & \cdots \\ 0 & 0 & \ddots & \\ \vdots & \vdots & & \dfrac{dt_A(x_A)}{dx_A} \end{bmatrix} \tag{3.18}$$

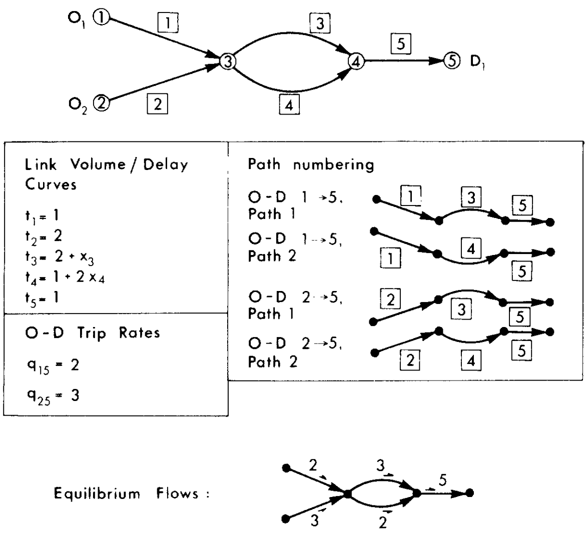
This matrix is positive definite since it is a diagonal matrix with strictly positive entries (all entries are positive because of condition [3.5b]). The objective function is thus strictly convex, and since the feasible region is convex as well, the UE program has a unique minimum.

This result means that there is only one flow pattern that minimizes program [3.3]. As shown in the last section, this minimum is a user-equilibrium flow pattern and consequently, the UE flow pattern can be found by minimizing this program.

The strict convexity of the UE program was established above with respect to the link flows. This program, however, is not convex with respect to path flows and, in fact, the equilibrium conditions themselves are not unique with respect to path flows. This point is demonstrated in the simple network example depicted in Figure 3.4. The network includes two origin–destination pairs connected by two paths each, as shown in the figure. The figure also depicts the link performance functions and the O–D trip rates ($q_{15} = 2$ and $q_{25} = 3$).

The equilibrium link flows for this example are given by $x_1 = 2$, $x_2 = 3$,

†Note that Eq. [3.18] is the Jacobian of the link-travel-time vector, $\mathbf{t}$, with respect to the link flows, $\mathbf{x}$. The Jacobian matrix of a given vector is the matrix of first derivatives of each of the vector components with respect to the arguments of these components.



Link Volume / Delay Curves
$t_1 = 1$
$t_2 = 2$
$t_3 = 2 + x_3$
$t_4 = 1 + 2x_4$
$t_5 = 1$
O-D Trip Rates
$q_{15} = 2$
$q_{25} = 3$

$$c_1^{15} = t_1(x_1) + t_3(x_3) + t_5(x_5) = 1 + (2+3) + 1 = 7$$

$$c_2^{15} = t_1(x_1) + t_4(x_4) + t_5(x_5) = 1 + (1+2\cdot 2) + 1 = 7$$

$$c_1^{25} = t_2(x_2) + t_3(x_3) + t_5(x_5) = 2 + (2+3) + 1 = 8$$

$$c_2^{25} = t_2(x_2) + t_4(x_4) + t_5(x_5) = 2 + (1+2\cdot 2) + 1 = 8$$

Figure 3.4 Equilibrium flows and path travel times in a network with two O–D pairs and five links.

$x_3 = 3$, $x_4 = 2$, and $x_5 = 5$, as shown in Figure 3.4. These equilibrium flows can be achieved by many combinations of path flows; for example:

$$f_1^{15} = 0, \quad f_2^{15} = 2, \quad f_1^{25} = 3, \quad \text{and} \quad f_2^{25} = 0$$

or

$$f_1^{15} = 2, \quad f_2^{15} = 0, \quad f_1^{25} = 1, \quad \text{and} \quad f_2^{25} = 2$$

Both these path-flow patterns generate the same link-flow pattern shown in Figure 3.4. In fact, any path-flow pattern that satisfies

$$f_1^{15} = 2\alpha, \quad f_2^{15} = 2(1-\alpha), \quad f_1^{25} = 3 - 2\alpha, \quad \text{and} \quad f_2^{25} = 2\alpha$$

for any value of α such that $0 \le \alpha \le 1$ will generate the equilibrium link-flow pattern in this example. Thus there are, in principle, an infinite number of equilibrium path flows.

The convexity of the UE equivalent minimization program with respect to link flows can be established without analyzing the Hessian of $z(\mathbf{x})$, by making use of the properties of convex functions. The objective function of the UE minimization consists of a sum each element of which is an integral of an increasing function. Such an integral is always strictly convex (prove it) and the sum of strictly convex functions is always strictly convex. Thus $z(\mathbf{x})$ is a strictly convex function which has only one minimum. This point is also the user-equilibrium solution for the network, as shown by the first-order condition.

The next two chapters focus on solution algorithms for convex minimization problems in general and the UE program in particular. Before discussing these algorithms, however, the next two sections of this chapter deal with some related minimization programs.

3.4 THE SYSTEM-OPTIMIZATION FORMULATION

As mentioned in Section 3.1, the UE minimization program is a mathematical construct that lacks an intuitive interpretation. It is merely an efficient method for solving the user equilibrium equations. These equations describe the flow pattern resulting from each motorist's choice of the shortest travel-time route from origin to destination.

This section examines a related minimization program including an objective function that has a straightforward interpretation and is subject to the same constraints as the UE equivalent program. The objective function of this program is the total travel time spent in the network. The flow pattern that solves this program minimizes this objective function while satisfying the flow conservation constraints (i.e., all the O–D trip rates are assigned to the network). This program can be expressed as follows:

$$\min \tilde{z}(\mathbf{x}) = \sum_a x_a t_a(x_a) \qquad [3.19a]$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\, r, s \qquad [3.19b]$$

$$f_k^{rs} \geq 0 \qquad \forall\, k, r, s \qquad [3.19c]$$

As in the UE program, the objective function is formulated in terms of link flows while the constraints are formulated in terms of path flows.

Program [3.19] is known as the system-optimization (SO) program. The flow pattern that minimizes this program does not generally represent an equilibrium situation. Except in special cases, it can result only from joint decisions by all motorists to act so as to minimize the total system travel time rather than their *own*. In other words, at the SO flow pattern, drivers may be able to decrease their travel time by unilaterally changing routes. Such a situation is unlikely to sustain itself and consequently the SO flow pattern is not stable and should not be used as a model of actual behavior and equilibrium. (The only cases in which the SO flow pattern can be used to represent equilibrium are those special cases in which the SO solution is identical to the UE solution.)

The significance of the SO formulation and the resulting flow pattern is that the value of the SO objective function may serve as a yardstick by which different flow patterns can be measured. Indeed, total (systemwide) travel time is a common measure of performance of a network under a given scenario (see Section 1.1). This measure can be computed in a straightforward manner given the equilibrium flows, and it does not require any data in addition to those required for the equilibrium analysis itself. Thus the flow pattern associated with any proposed project can be measured in terms of the total travel time associated with it relative to the minimum possible total travel time. By definition this minimum is obtained by solving the SO program.†

The necessary conditions for a minimum for the SO program are given by the first-order conditions for a stationary point of the following Lagrangian program:

$$\tilde{L}(\mathbf{f}, \tilde{\mathbf{u}}) = \tilde{z}[\mathbf{x}(\mathbf{f})] + \sum_{rs} \tilde{u}_{rs}\left(q_{rs} - \sum_{k} f_k^{rs}\right) \qquad [3.20a]$$

where $\tilde{L}(\mathbf{f}, \tilde{\mathbf{u}})$ has to be minimized with respect to **f**, subject to the set of nonnegativity conditions

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [3.20b]$$

The variable $\tilde{u}_{rs}$ is the Lagrange multiplier (or the dual variable) associated with the flow conservation constraint of O–D pair r–s (Eq. [3.19b]). The first-order conditions for a stationary point of Eqs. [3.20] are (see Eqs. [3.9])

$$f_k^{rs} \frac{\partial \tilde{L}(\mathbf{f}, \tilde{\mathbf{u}})}{\partial f_k^{rs}} = 0 \quad \text{and} \quad \frac{\partial \tilde{L}(\mathbf{f}, \tilde{\mathbf{u}})}{\partial f_k^{rs}} \geq 0 \qquad \forall\ k, r, s \qquad [3.21a]$$

as well as

$$\frac{\partial \tilde{L}(\mathbf{f}, \tilde{\mathbf{u}})}{\partial \tilde{\mathbf{u}}_{rs}} = 0 \qquad \forall\ r, s \qquad [3.21b]$$

and

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [3.21c]$$

Again, the asterisks denoting the optimal solution in terms of **f*** and $\tilde{\mathbf{u}}^*$ have been omitted from these equations, as in Eqs. [3.9].

As in the case of the equivalent UE program, conditions [3.21b] and [3.21c] simply restate the flow conservation and nonnegativity constraints, respectively, and are therefore not discussed further. Conditions [3.21a] can be expressed explicitly by deriving the partial derivatives of the Lagrangian with respect to the path flows. These derivatives are given by

$$\frac{\partial \tilde{L}(\mathbf{f}, \tilde{\mathbf{u}})}{\partial f_l^{mn}} = \frac{\partial}{\partial f_l^{mn}} \tilde{z}[\mathbf{x}(\mathbf{f})] + \frac{\partial}{\partial f_l^{mn}} \sum_{rs} \tilde{u}_{rs}\left(q_{rs} - \sum_{k} f_k^{rs}\right) \qquad \forall\ m, n, l \quad [3.22]$$

The partial derivative of Lagrangian [3.22] consists of two terms, which

†In addition, the SO program is used as a bound in many mathematical programs dealing with network design.

are calculated separately below. The second term is similar to the second term of Eq. [3.10] and therefore (see Eq. [3.14])

$$\frac{\partial}{\partial f_l^{mn}} \sum_{rs} \tilde{u}_{rs}\left(q_{rs} - \sum_k f_k^{rs}\right) = -\tilde{u}_{mn} \qquad \forall\ l, m, n \qquad [3.23]$$

The first term on the right-hand side of Eq. [3.22] includes the partial derivative of the SO objective functions with respect to the flow on path l between origin m and destination n. This derivative is given by

$$\begin{aligned}\frac{\partial}{\partial f_l^{mn}} \tilde{z}[\mathbf{x}(\mathbf{f})] &= \sum_b \frac{\partial \tilde{z}(\mathbf{x})}{\partial x_b}\frac{\partial x_b}{\partial f_l^{mn}} = \sum_b \frac{\partial \tilde{z}(\mathbf{x})}{\partial x_b}\delta_{b,l}^{mn} \\ &= \sum_b \delta_{b,l}^{mn} \frac{\partial}{\partial x_b} \sum_a x_a t_a(x_a) \\ &= \sum_b \delta_{b,l}^{mn}\left[t_b(x_b) + x_b \frac{dt_b(x_b)}{dx_b}\right] \qquad \forall\ l, m, n \qquad [3.24]\end{aligned}$$

This derivative can be interpreted, intuitively, by letting

$$\tilde{t}_a(x_a) = t_a(x_a) + x_a \frac{dt_a(x_a)}{dx_a} \qquad \forall\ a$$

The travel time $\tilde{t}_a$ can be interpreted as the marginal contribution of an additional traveler† on the ath link to the total travel time on this link. It is the sum of two components: $t_a(x_a)$ is the travel time experienced by that additional traveler when the total link flow is x_a, and $dt_a(x_a)/dx_a$ is the additional travel-time burden that this traveler inflicts on each one of the travelers already using link a (there are x_a of them). Using the new travel time variable, $\tilde{t}_a$, Eq. [3.24] can be written as

$$\begin{aligned}\frac{\partial}{\partial f_l^{mn}} \tilde{z}[\mathbf{x}(\mathbf{f})] &= \sum_b \delta_{b,l}^{mn}\tilde{t}_b \\ &= \tilde{c}_l^{mn} \qquad \forall\ l, m, n \qquad [3.25]\end{aligned}$$

where $\tilde{c}_l^{mn}$ is interpreted in a fashion analogous to $\tilde{t}_a$. It is the marginal total travel time on path l connecting O–D pair m–n.

The first-order conditions for the SO program can now be written as

$$f_l^{mn}(\tilde{c}_l^{mn} - \tilde{u}_{mn}) = 0 \qquad \forall\ l, m, n \qquad [3.26a]$$

$$\tilde{c}_l^{mn} - \tilde{u}_{mn} \geq 0 \qquad \forall\ l, m, n \qquad [3.26b]$$

$$\sum_l f_l^{mn} = q_{mn} \qquad \forall\ m, n \qquad [3.26c]$$

$$f_l^{mn} \geq 0 \qquad \forall\ l, m, n \qquad [3.26d]$$

†More accurately, an additional infinitesimal flow unit.

Equations [3.26a] and [3.26b] state that, at optimality, the marginal total travel times on all the used paths connecting a given O–D pair are equal. This result is reminiscent of classical optimization results in which the marginals of the functions minimized have to be equal at the solution point. The flow on a given route is zero only if the marginal total travel time on this route is greater than or equal to the marginal total travel time on the used routes. The value of the dual variable $\tilde{u}_{mn}$ at optimality is the marginal travel time on all the used paths between m and n.

For the solution of the program SO to be unique, it is sufficient to show that the Hessian of $\tilde{z}(\mathbf{x})$ is positive definite. A typical term of this Hessian can be obtained by taking the derivative of the objective function with respect to x_m and x_n, that is,

$$\frac{\partial \tilde{z}(\mathbf{x})}{\partial x_b} = \frac{\partial}{\partial x_b} \sum_a x_a t_a(x_a)$$

$$= t_b(x_b) + x_b \frac{dt_b(x_b)}{dx_b}$$

and

$$\frac{\partial^2 \tilde{z}(\mathbf{x})}{\partial x_b \, \partial x_a} = \begin{cases} 2 \dfrac{dt_a(x_a)}{dx_a} + x_a \dfrac{d^2 t_a(x_a)}{dx_a^2} & \text{for } b = a \\ 0 & \text{otherwise} \end{cases} \qquad \forall\, a, b \qquad [3.27]$$

As in the UE program, this result represents a diagonal Hessian with the nonzero terms $\partial^2 \tilde{z}(\mathbf{x})/\partial x_a^2$ given by Eq. [3.27]. This Hessian is positive definite if all those terms are positive, which is the case if all the link performance functions are convex. All performance functions dealt with in this text have the general shape shown in Figure 3.2, which is convex, and thus $dt_a^2(x_a)/dx_a^2 > 0$ and the diagonal terms are positive. Consequently, the SO program has a unique minimum in terms of link flows.

The next section compares some aspects of the SO problem to the UE program in order to illustrate the nature of the corresponding flow patterns.

3.5 USER EQUILIBRIUM AND SYSTEM OPTIMUM

It is interesting to note that when congestion effects are ignored, both UE and SO programs will produce identical results. Imagine a network where $t_a(x_a) = t'_a$. In other words, each link travel time is not a function of the flow on that (or any other) link. In this case, the SO objective function would be

$$\tilde{z}(\mathbf{x}) = \sum_a x_a t'_a \qquad [3.28a]$$

and the UE objective function

$$z(\mathbf{x}) = \sum_a \int_0^{x_a} t'_a \, d\omega$$

$$= \sum_a x_a t'_a \qquad [3.28b]$$

which is identical to the SO objective function, $\tilde{z}(\mathbf{x})$.

Minimizing the objective function shown in Eqs. [3.28] subject to the flow constraints [3.19b] and [3.19c] is an easier task than solving either the SO or UE problems. The reason is that in this case the link travel times are not a function of the link flows, whereas, in general, both the SO and UE problems assume that t_a does vary with x_a. The problem here is to find the flow pattern that minimizes the total travel time over the network, given the (fixed and known) values of the link travel times and the O–D matrix. The solution of this problem is conceptually straightforward—all the flow for a given O–D pair r–s, q_{rs}, is assigned to the minimum-travel-time path connecting this pair. All other paths connecting this O–D pair do not carry flow. Consequently, this traffic assignment procedure is known as the "all-or-nothing" assignment. The resulting flow pattern is both an equilibrium situation (since no user will be better off by switching paths) and an optimal assignment (since the total travel time in the system is obviously minimized).

The study of this special case is important in the development of solution algorithms for the more general problem of equilibrium assignment. This study is undertaken in Chapter 5, while the remainder of this section deals with both the UE and SO formulations.

The similarity in the structures of the SO and UE programs can be expressed in various ways. For example, if the travel times over the network are expressed in terms of $\tilde{t}_a(x_a)$, the solution of the UE program with these travel times will produce an SO flow pattern. Similarly, the SO formulation with link travel-time functions given by

$$\hat{t}_a(x_a) = \frac{1}{x_a} \int_0^{x_a} t_a(\omega) \, d\omega \qquad [3.29]$$

will result in a UE flow pattern (see Problem 3.10).

In cases in which the flow over the network is relatively low, the network links are not congested. The marginal travel time on each link, at this point, is very small due to the shape of the link performance functions (see Figure 3.2); the slope of this function is very small for low flow. In this case, the UE and SO flow patterns are similar since the travel times are almost insensitive to additional flows. The situation, then, is close to the fixed-travel-time case described in Eqs. [3.28].

As the flows between origins and destinations increase, the UE and SO patterns become increasingly dissimilar. Large flows mean that some links carry an amount of flow which is near their capacity. In that range the marginal travel time, $\tilde{t}_a(x_a)$, becomes very large though the travel time itself remains

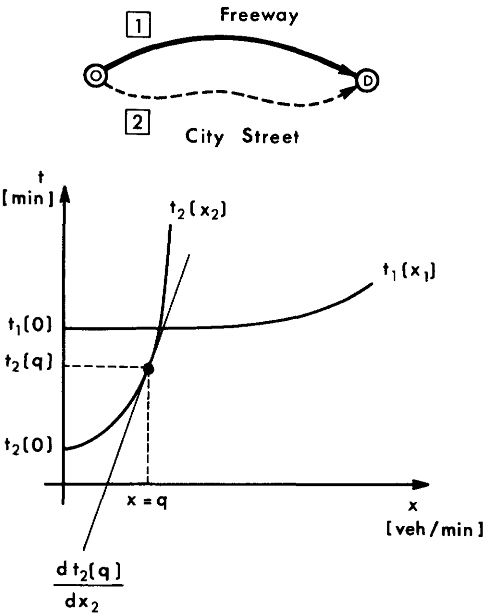


Figure 3.5 User equilibrium and system optimization. The UE solution to the two-link network equilibrium problem (with $x_1^* = 0$ and $x_2^* = q$) is not a system optimizing solution since $t_1(x_1^*) + x_1^* \, dt_1(x_1^*)/dx_1 \neq t_2(x_2^*) + x_2 \, dt_2(x_2^*)/dx_2$.

bounded. Thus the differences between solving the same problem with the set of marginal travel times $\{\tilde{t}_a\}$, versus solving it with the travel times themselves, $\{t_a\}$, increase as the network becomes more congested.

To understand this better, consider the network example shown in Figure 3.5. This network includes one O–D pair connected by two links (paths). Assume that the top link (number 1 in the figure) is a freeway, while the other link represents a city street. Hypothetical performance curves for such links are shown in the figure. If the total O–D flow is q, the UE solution will be the one shown in the figure with $x_1 = 0$ and $x_2 = q$. In this case, $t_2(q) < t_1(0)$, and no user will choose to travel on the freeway. Note, however, that the derivative of $t_2(x_2)$ at $x = q$, $dt_2(q)/dx_2$, is relatively large and it is therefore possible that $\tilde{t}_1(0) < \tilde{t}_2(q)$. In other words, this solution is not optimal from the system perspective. The SO solution to this problem may include some flow using the top route as well. If one driver shifts from route 2 to route 1 (at the UE flow pattern shown in the figure), his or her own travel time will increase—the driver will experience $t_1(0)$ instead of $t_2(q)$. The travel time experienced by each of the remaining drivers on route 2 will, however, decrease by $dt_2(q)/dx_2$. Thus, from the system perspective, an increase of $[t_1(0) - t_2(q)]$ may be more than offset by a decrease of $(q - 1) \, dt_2(q)/dx_2$. The SO flow pattern is achieved only when

$$t_1(x_1) + x_1 \frac{dt_1(x_1)}{dx_2} = t_2(x_2) + x_2 \frac{dt_2(x_2)}{dx_2} \qquad [3.30]$$

(Problem 3.12 includes a numerical example intended to demonstrate this point.) Note that flow should still be treated as a continuous quantity; the mention of an integer flow unit (one driver) above was made only to explain the concept.

A failure to realize the fundamental difference between the normative SO flow pattern and the descriptive UE flow pattern can lead to pseudo-paradoxical scenarios. The most famous of these is known as "Braess's paradox," which is described below.

Figure 3.6a depicts a simple network including one O–D pair connected by two paths and four links. The figure shows the two paths (numbered 1 and 2) and the congestion curves for each of the four links. Assume that 6 units of flow are to travel between O and D (i.e., $q = 6$). The user equilibrium flow pattern can be solved for this network by inspection (due to the travel time symmetry between the paths). Obviously, half the flow would use each path and the solution would be

$$f_1 = 3, \quad f_2 = 3 \qquad \text{flow units}$$

or, in terms of link flows,

$$x_1 = 3, \quad x_2 = 3, \quad x_3 = 3, \quad x_4 = 3 \qquad \text{flow units}$$

The associated link travel times are:

$$t_1 = 53, \quad t_2 = 53, \quad t_3 = 30, \quad t_4 = 30 \qquad \text{time units}$$

and the path times are

$$c_1 = 83, \quad c_2 = 83 \qquad \text{time units}$$

in accordance with the UE criterion. The total travel time on the network is 498 (flow · time) units.

Assume now that the local transportation authority decides to expand the transportation network in order to improve flow and reduce delay (as well as save energy and reduce pollution). This is to be accomplished by building a new highway connecting the two intermediate nodes as shown in Figure 3.6b. The figure shows the added (fifth) link, the performance function for this link, and the new path (number 3) generated as a result of the link addition.

The old UE flow pattern is no longer an equilibrium solution since under that flow pattern

$$x_1 = 3, \quad x_2 = 3, \quad x_3 = 3, \quad x_4 = 3, \quad x_5 = 0 \qquad \text{flow units}$$

with path travel times

$$c_1 = 83, \quad c_2 = 83, \quad c_3 = 70 \qquad \text{time units}$$

The travel time on the unused path (number 3) is lower than the travel time on the two used paths, meaning that this cannot be an equilibrium solution. An equilibrium flow pattern for the new network is given by the solution

$$x_1 = 2, \quad x_2 = 2, \quad x_3 = 4, \quad x_4 = 4, \quad x_5 = 2 \qquad \text{flow units}$$

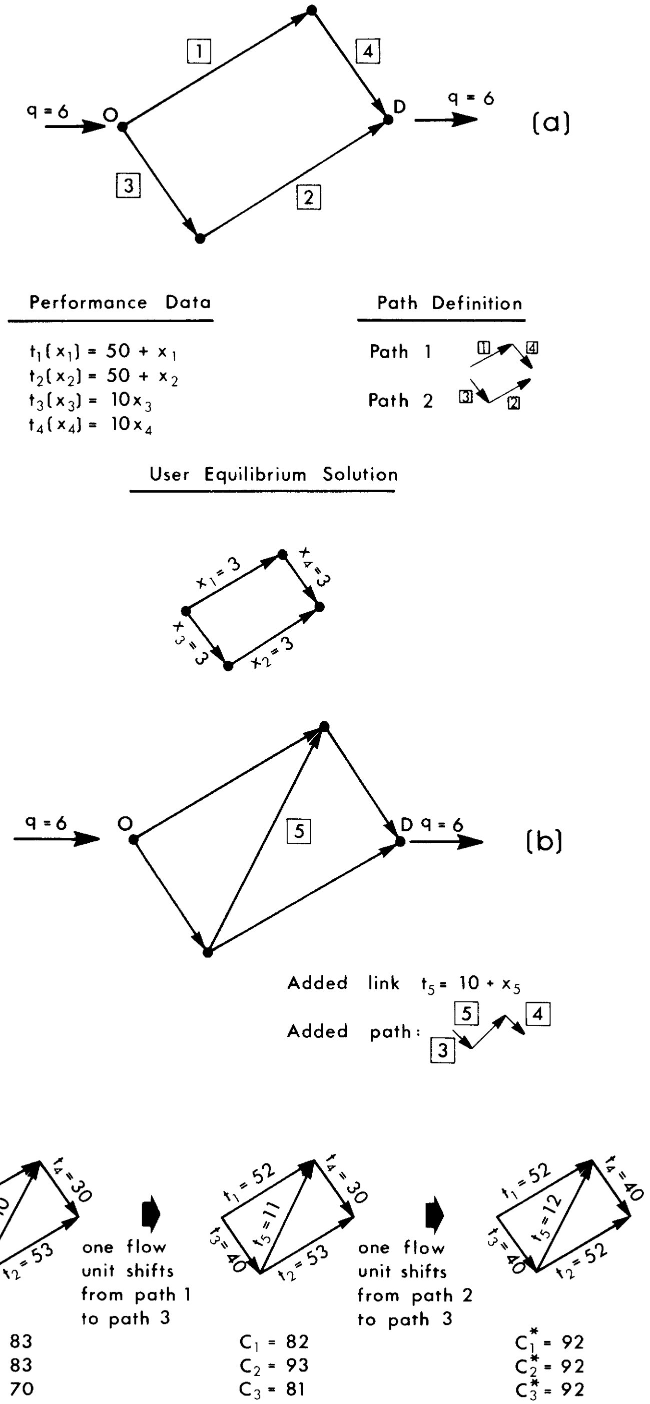


Figure 3.6 Braess's paradox example: (a) a user-equilibrium solution for a four-link network (link numbers are shown in the squares); (b) an additional link creates a UE solution with higher individual and total cost.

with path flows

$$f_1 = f_2 = f_3 = 2 \qquad \text{flow units}$$

and path travel times

$$c_1 = c_2 = c_3 = 92 \qquad \text{time units}$$

Figure 3.6b depicts a possible sequence of assignment of flow units that would generate this equilibrium from the old one.

The important point to note here is that the total travel time in the network is now 552 (flow · time) units as compared to 498 (flow · time) units before the link addition. Not only did the total travel time go up, but the travel time experienced by each traveler in the network increased from 83 time units to 92 time units. The additional link seems to have made the situation worse—congestion and delays increased instead of decreasing. This (seemingly) counter-intuitive result is known as Braess's paradox.

This "paradox" can, of course, be readily explained. The increase in travel time is rooted in the essence of the user equilibrium, where each motorist minimizes his or her own travel time. The individual choice of route is carried out with no consideration of the effect of this action on other network users. There is no reason, therefore, to expect the total travel time to decrease.

Looking at it from a mathematical programming point of view, the supply action (adding a link) was taken with the intention of reducing the SO objective function. The flow, however, is distributed according to the UE objective function, so the resulting pattern does not necessarily reduce the SO objective function. Had the flow before and after the link addition been assigned according to the SO objective function, the total travel time could not have increased with the addition of the new link. (Prove it.) Note that the value of the UE objective function did go down from a value of 399 before the link addition, to a value of 386, after the link addition.

From a more general perspective, Braess's paradox underscores the importance of a careful and systematic analysis of investments in urban networks. Not every addition of capacity can bring about all the anticipated benefits and in some cases, the situation may be worsened. In fact, traffic engineers have known for a long time that restrictions of travel choices and reductions in capacity may lead to better overall flow distribution patterns. This, for instance, is the underlying principle behind many traffic control schemes, such as ramp metering on freeway entrances.†

†Ramp metering is the process of restricting the entry of flow onto a freeway, usually by installing a traffic light at the entrance ramp. This light controls the number of cars allowed to enter the freeway during certain times of the day (typically, the peak traffic period). In other words, it effectively lowers the capacity of one of the network links (the ramp). The metering causes drivers to use alternative routes by raising the travel time associated with the freeway entrance. This reduction in capacity leads to better overall conditions and lower total delay compared with the situation associated with no controls.

3.6 SUMMARY

The focus of this chapter is on the formulation of the traffic assignment problem as a mathematical program. This program is given by Eqs. [3.3]. To demonstrate that the solution of this program is equivalent to the solution of the UE conditions, it is shown that the equilibrium equations are, in fact, the first-order conditions of the program. This guarantees that the equilibrium conditions hold at any stationary point of the program. Next, it is shown that this program is strictly convex, meaning that it has only one stationary point which is a minimum. All this proves that instead of solving the equilibrium equations directly, the equilibrium flows can be determined by minimizing the equivalent mathematical program. This approach to the solution of equilibrium problems is adopted throughout this book.

The nature of the user equilibrium and its equivalent minimization program are illustrated in this chapter by contrasting this formulation with the system-optimization formulation. The SO formulation calls for the flow pattern that minimizes the total travel time in the system. It is generally not an equilibrium flow pattern, since under it some travelers can be better off by unilaterally changing routes. It represents, however, the flow pattern for which the total systemwide travel time is the lowest. The difference between the UE equivalent minimization and the SO program is that the former is devised to describe motorists' behavior, whereas the latter is normative in nature. A failure to understand these differences may lead to "paradoxical" situations when travel choices are added to the network but all users are made worse off. Such situations highlight the noncooperative behavior represented by the user equilibrium.

3.7 ANNOTATED REFERENCES

The formulation of the user-equilibrium problem as a mathematical program was first developed by Beckmann et al. (1956), who proved the equivalency and the existence and uniqueness of the solution. Boyce (1981), in an editorial note, traces the historical development of the concept of equilibrium assignment, including many algorithms and problem formulations. The differences between the user-equilibrium and system-optimizing flow patterns over a transportation network are discussed by Potts and Oliver (1972) and Newell (1980), from whom the example shown in Figure 3.5 was taken. The paradox presented in Section 3.5 was discussed and explained by Murchland (1970) following Braess (1968). A case in which this phenomenon actually took place is reported by Knödel (1969). Newell (1980) pointed out the evident nature of this phenomenon in the context of standard traffic engineering practice.

3.8 PROBLEMS

3.1. Consider the network in Figure P3.1. It includes two O–D pairs (1 → 4 and 2 → 4) and five links. (The link numbers are shown on the respective links.) Devise a numbering system for the paths and identify the values of all the indicator variables. Write the link–path incidence matrix for this network. Use this matrix to get x_3 from **f**.

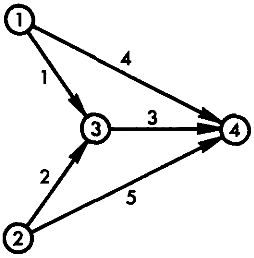


Figure P3.1

3.2. Solve the network example in Figure 3.3 by setting up the equivalent minimization and by using Lagrange multipliers. Comment on the value of the multipliers at the solution point.

3.3. Find the user-equilibrium flow and travel times for the network shown in Figure P3.2, where

$$t_1 = 2 + x_1^2$$

$$t_2 = 3 + x_2$$

$$t_3 = 1 + 2x_3^2$$

$$t_4 = 2 + 4x_4$$

(The link numbers are shown on the respective links.) The O–D trip rate is 4 units of flow between nodes 1 and 3.

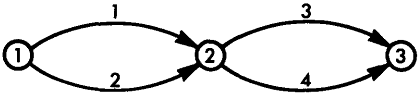


Figure P3.2

3.4. Solve for the equilibrium flows in the network example depicted in Figure 3.4. Set up the UE program and solve the first-order conditions.

*3.5. (a) Express the Hessian of the UE objective function in terms of path flow over the network depicted in Figure 3.4. Show that it is not a positive-definite matrix.

(b) Show that the path-flow Hessian of the UE objective function is usually not a positive-definite matrix. For what type of network will this Hessian be positive definite?

3.6. Show that the integral of an increasing function is always a convex function.

3.7. Find the system-optimizing flow pattern for the network example in Figure 3.3.

Compare this flow pattern to the UE flow pattern and comment on the differences.

3.8. Find the system-optimizing flow pattern for the network example discussed in Problem 3.3. Compare it to the UE flow pattern.

3.9. Does the SO program have a unique solution in terms of path flows? Explain your answer.

3.10. **(a)** Show that the solution of the SO program with travel times $\hat{t}_a(x_a)$ (see Eq. [3.29]) is an UE flow pattern.

(b) Show that the solution of the UE program with travel times $\tilde{t}_a(x_a)$ (see Eq. [3.24] and the definition that follows) is an SO flow pattern.

3.11. Show, analytically, that as the flows over the network become smaller (and congestion decreases), the SO flow pattern tends to grow in similarity to the UE flow pattern.

3.12. Consider the network of two routes (a freeway and a city street connecting one O–D pair) shown in Figure 3.5. Let

$$t_1 = 3 + 0.5x$$

$$t_2 = 1 + x_2$$

$$q = 1.5$$

(a) Find the UE solution and demonstrate that it is not an SO solution.

(b) Find the SO solution and demonstrate that it is not an UE solution.

3.13. Solve for the system-optimizing flow pattern over the network depicted in Figure 3.6 before and after the link addition. Show that the total travel time decreases. Explain why the total travel time would never increase if a link is added and the flow pattern follows the SO rule.

3.14. Assume that a transportation planner can set a toll on each one of the paths connecting the origins to the destination shown in Figure P3.3.

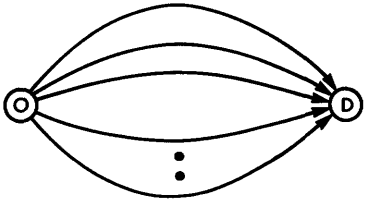


Figure P3.3

(a) Given that the O–D flow is q and the performance function on each link is $t_a(x_a)$, how should the tolls be set so that the total travel time in the network is minimized?

(b) Assuming that the link performance functions are linear, derive a closed-form expression for these tolls.

(c) Comment about the feasibility of this approach in real urban networks.

4

Review of Some Optimization Algorithms

This chapter describes some of the most common minimization algorithms. It is not a comprehensive ıeview, but rather a presentation of methods that are applicable to the solution of minimization programs of the type formulated in this text as well as methods that bring out some important principles. The discussion deals separately with the minimization of a function of a single variable and the minimization of multidimensional functions.

The focus of the discussion is on the mechanics of the algorithms and, therefore, proofs of convergence are not given explicitly. Such proofs can be found in most of the nonlinear programming books mentioned at the end of this chapter. The last section is devoted exclusively to one algorithm—the convex combinations method. This algorithm is the basis for solving many equilibrium problems.

4.1 ONE-DIMENSIONAL MINIMIZATION

This section deals with the minimization of a nonlinear function of a single variable, $z(x)$. The regularity conditions mentioned in Chapter 2 are still assumed to hold; that is, it is assumed that x lies within some finite interval $[a, b]$ and that $z(x)$ is continuous and uniquely defined everywhere in that interval. These requirements ensure the existence of a finite minimum of $z(x)$ for some x in the interval of interest. For the purposes of this discussion it is assumed that $z(x)$ is ditonic over the interval $[a, b]$, implying that it has only a single, unique minimum in that interval.

The study of one-dimensional optimization methods is important mainly because such an optimization (or *line search*) is in many cases a part of an algorithm designed to find the minimum of multivariate functions. Furthermore, some of the principles imbedded in the algorithms described below are used in minimizing multivariate functions as well.

This section includes two basic approaches to single-dimensional minimization. The first is known as interval reduction and includes the golden section and the bisection methods. The second utilizes quadratic curve fitting and includes Newton's search, the false position method, and the quadratic approximation method.

Interval Reduction Methods

Interval reduction methods involve iterative procedures in which each iteration is focused on a current interval. The current interval in the nth iteration is a portion of $[a, b]$, denoted $[a^n, b^n]$, which was determined to include the minimum point, x^*. At each iteration this interval is examined and divided into two parts: the part in which the minimum *cannot* lie and the current interval for the next iteration. The part in which the minimum cannot lie is discarded and the procedure is repeated for the new current interval. These procedures start by designating $[a, b]$ as the first current interval (i.e., $a^0 = a$ and $b^0 = b$). The interval is then reduced at each successive iteration until a good approximation (a small enough current interval) for x^* is obtained.

To understand this reduction process better, let the size of the current interval at iteration n be denoted by I_n and let $r_n = I_{n+1}/I_n$ denote the interval reduction ratio for the nth iteration. Since there is no reason to believe that any of these algorithms would work better (in terms of interval reduction) in any given iteration, the reduction ratio is usually a constant, (i.e., $r_n = r$ for every n).

Interval reduction algorithms are typically terminated when the size of the interval of interest is less than a predetermined constant. The estimate of x^* is then the midpoint, $\hat{x}$, of the interval remaining after N iterations, I_N [i.e., $\hat{x} = (a^N + b^N)/2$]. If the optimum has to be estimated with a tolerance of $\pm\epsilon$ (i.e., x^* must lie within $\hat{x} \pm \epsilon$), then the number of required iterations can be calculated as a function of the length, I_0, of the initial interval (see Problem 4.1). The number is

$$N = \text{INT}\left[\frac{\log 2\epsilon - \log I_0}{\log r} + 1\right] \qquad [4.1]$$

where INT $[\cdot]$ means the integer part of the argument.

The various interval reduction algorithms differ from each other only in the rules used to examine the current interval and to decide which portion of it can be discarded.

Golden section method. The interval reduction strategy of the golden section search is based on a comparison of the values of $z(x)$ at two points, x_L^n and x_R^n (where $x_L^n < x_R^n$). These points are within the interval of interest $[a^n, b^n]$ at the nth iteration. The choice rule for selecting the interior points is the unique feature of this method; it is explained below, following an explanation of the interval discarding process.

The discarding mechanism is demonstrated in Figure 4.1, depicting a ditonic function, $z(x)$, which has to be minimized in the interval $[a^n, b^n]$. The top drawing (denoted "iteration n") shows the two interior points at the nth iteration x_L^n and x_R^n. In this case, $z(x_L^n) > z(x_R^n)$. Since the function is ditonic, the optimum must lie "to the right" of x_L^n (i.e., $x^* \geq x_L^n$), and thus the interval $[a^n, x_L^n]$ can be discarded. This completes the nth iteration. The new current interval [for the $(n + 1)$st iteration] is $[a^{n+1}, b^{n+1}]$, where $a^{n+1} = x_L^n$ and $b^{n+1} = b^n$. The interval reduction process continues with two new interior points, x_L^{n+1} and x_R^{n+1}, as shown in the bottom drawing of Figure 4.1 (labeled "iteration $n + 1$"). Note that if the function was such that $z(x_R^n)$ was greater than $z(x_L^n)$, then the interval $[x_R^n, b^n]$ would have been discarded at the nth iteration. This is the case in iteration $n + 1$, where $z(x_L^{n+1}) < z(x_R^{n+1})$. In this case the interval $[x_R^{n+1}, b^{n+1}]$ is discarded and the new current interval is $[a^{n+1}, x_R^{n+1}]$ (since x^* cannot be "to the right" of x_R^{n+1}).

The essence of the golden section method is in the rule used for choosing x_L^n and x_R^n, given an interval $[a^n, b^n]$. This rule is designed to minimize the number of function evaluations. Given a new current interval the algorithm

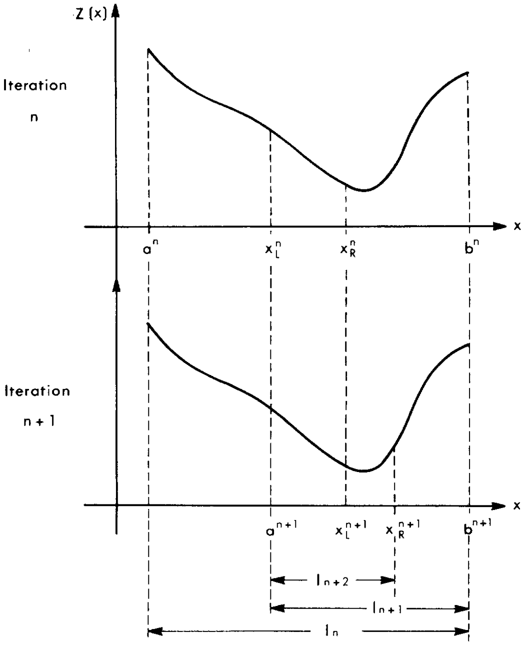


Figure 4.1 The interval reduction sequence followed by the golden section method.

needs two interior points to continue. The golden section procedure makes use of one of the interior points from the last interval (where the function value is already known). Only one new point where the function value needs to be evaluated is therefore added at each iteration.

This arrangement is illustrated in Figure 4.1, where, when going from the nth to the $(n + 1)$st iteration, only one interior point has to be added and evaluated (x_R^{n+1} in the bottom drawing); the other is available from the nth iteration since $x_L^{n+1} = x_R^n$ [so $z(x_L^{n+1})$ need not be evaluated because it equals $z(x_R^n)$]. Such a sequence of interior points can be obtained, while keeping a constant reduction ratio, by using $r = 0.618$ [more precisely, $r = \frac{1}{2}(\sqrt{5} - 1)$]. Accordingly, the interior points are selected so that x_R^n is 0.618 of the interval length to the right of a^n and x_L^n is 0.618 of the interval length to the left of b^n (or 0.382 of the interval length to the right of a^n). The quantity $\frac{1}{2}(\sqrt{5} - 1)$ is known as the "golden section," from which the method derives its name.

Note that such a sequence of intervals leads to a situation in which

$$I_n = I_{n+1} + I_{n+2} \qquad [4.2]$$

as shown in Figure 4.1 (see Problem 4.2).

A flowchart of the algorithm is presented in Figure 4.2.† This algorithm takes as input the function to be minimized, the interval end points, and the accuracy required. The output includes the estimate of the optimum, $\hat{x}$, the number of iterations performed, N, and the accuracy actually obtained, $\frac{1}{2}(b^N - a^N)$. Note that given r, the number of iterations needed to achieve a given degree of accuracy, can be determined a priori by using Eq. [4.1].

A well-known search method similar to the golden section is the Fibonacci algorithm mentioned in the references given in Section 4.5. It is slightly more efficient than the golden section procedure, due to better positioning of the first two interior points. From a practical point of view, however, the small advantage offered by this algorithm does not justify its detailed study (see Problem 4.5) herein.

Bisection method. In many cases, the derivative of the function to be minimized can be evaluated easily and the search for the minimum can therefore be expedited. The method of interval bisection exploits the fact that a ditonic function is monotonic on each side of the minimum. In other words, the derivative of the function to be minimized, $dz(x)/dx$, is negative for $x < x^*$ and positive for $x > x^*$.

The algorithm computes the derivative of $z(x)$ at the midpoint of the current interval, $[a^n, b^n]$. Denote this point by $\hat{x}^n$. If $dz(\hat{x}^n)/dx < 0$, then $x^* > \hat{x}^n$, meaning that the interval $[a^n, \hat{x}^n]$ can be discarded. The next current interval will thus be $[\hat{x}^n, b^n]$. If $dz(\hat{x}^n)/dx > 0$, then $x^* < \hat{x}^n$, and the search can

†The notation "$:=$" used in the figure indicates the equality sign in programming languages (i.e., the assignment of a variable's value). For example, $n := n + 1$ means an increment of the counter n by 1.

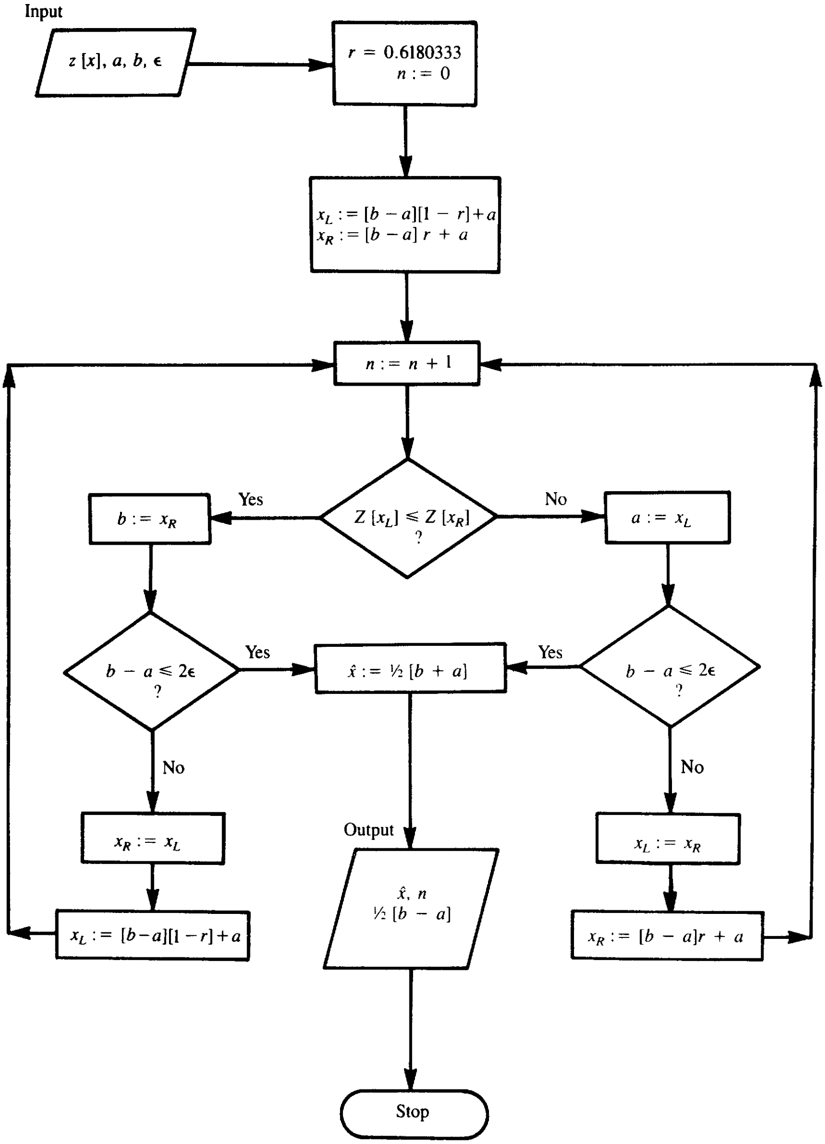


Figure 4.2 Flowchart of the golden section algorithm.

focus on the interval $[a^n, \hat{x}^n]$. Figure 4.3 depicts a flowchart of the bisection method (known also as Bolzano search).

As with the golden section method, a convergence criterion can be specified, causing the precedure to terminate (the estimate of x^* is taken at the middle of the remaining interval) when this criterion is met. The reduction ratio for this method is $r = 0.5$, and Eq. [4.1] can be used to determine the number of iterations required for a given accuracy. For example, only six

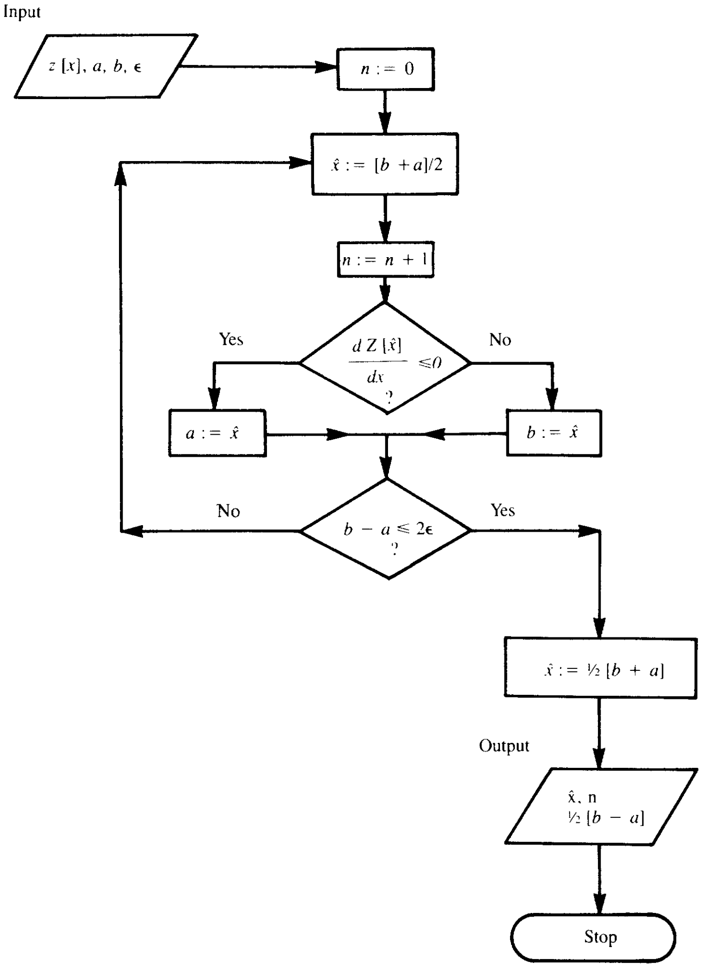


Figure 4.3 Flowchart of the bisection algorithm.

iterations of the bisection method are required to determine the minimum of $z(x)$ with an accuracy of $\pm 1\%$ of the length of the original interval. The golden section method, by way of comparison, would require nine iterations, involving 10 evaluations of $z(x)$ to reach this degree of accuracy. Table 4.1 shows the ratio of the length of the current interval and the original interval, after N function evaluations† for each of the two interval reduction methods discussed in this section. The table demonstrates that the convergence rate of the bisection method is almost twice that of the golden section method. The bisection method requires, however, that the derivative of $z(x)$ be evaluated in every iteration. This may not be easy in some cases, and thus this algorithm

†Note that the first iteration of the golden section method requires two evaluations of $z(x)$.

TABLE 4.1 Convergence Rate for Interval Reduction Methods (I_N/I_0 versus N for each algorithm)

Number of Function Evaluations, N	Size of Current Interval as a Fraction of the Initial Interval	
	Golden Section	Bisection
2	0.6180	0.2500
3	0.3819	0.1250
4	0.2361	0.0625
5	0.1459	0.0313
6	0.0902	0.0156
7	0.0557	0.0078
8	0.0344	0.0039
9	0.0213	0.0020
10	0.0132	0.0010

should be preferred to the golden section algorithm only in cases in which the calculation of the derivative is not much more difficult than calculating the function itself.

All the interval reduction methods produce, after a given number of iterations, a final interval $[a^N, b^N]$ which contains the minimum. If the minimized function is convex, the optimal value of $z(x)$ can be bounded. As mentioned in Section 2.1, a linear approximation to a convex function will always underestimate it. Thus

$$z(x^*) \geq z(a^N) + \frac{dz(a^N)}{dx}(x^* - a^N) \qquad [4.3a]$$

and

$$z(x^*) \geq z(b^N) + \frac{dz(b^N)}{dx}(x^* - b^N) \qquad [4.3b]$$

Since the value of x^* is not known, the value of b^N can be substituted for x^* in Eq. [4.3a], and a^N can be substituted for x^* in Eq. [4.3b]. The higher of the two lower bounds suggested by these linear approximations is a tighter lower bound for $z(x^*)$. Alternatively, the two approximations can be solved simultaneously for an even tighter bound.

Curve-Fitting Methods

When the function to be minimized, $z(x)$, is not only ditonic but also relatively smooth, such smoothness can be exploited by algorithms that are more efficient than the aforementioned interval reduction methods. Curve-fitting methods work by iteratively improving a current solution point. The characteristics of the function at the last point (or points) are utilized to generate a smooth approximation for $z(x)$. (All the methods described in this section use a parabola for this purpose.) The new point is taken to be at the minimum of this approximation as explained below in the discussion of specific algorithms. These algorithms differ from each other in the technique used to generate the approximation of the objective function.

Curve-fitting methods exhibit many of the characteristics of general minimization methods in that (unlike interval reduction methods) they generate a series of points $x^1, \ldots, x^N, \ldots$ that converge to the minimum, x^*. Each point is generated by applying some algorithmic procedure to the previous point, and the algorithm terminates when a convergence criterion is met.

The convergence criterion for curve-fitting methods can be based on various rules. These rules are typically based on the required accuracy of the solution or on "cost-effectiveness" considerations. For example, the algorithm can be terminated when the marginal contribution of an iteration to the improvement in the solution becomes small. In other words, if

$$z(x^{n-1}) - z(x^n) \leq \kappa$$

where κ is a predetermined constant (tolerance), the algorithm terminates. Alternatively, a dimensionless constant based on the relative change between successive solutions (i.e., $[z(x^{n-1}) - z(x^n)]/z(x^{n-1})$ can be used to test for convergence.† In many cases the algorithms can terminate on the basis of the change in the variable value (e.g., when $|x^n - x^{n-1}| \leq \kappa'$ where κ' is some other predetermined constant). The closeness of any particular solution to the minimum can also be tested by checking the derivative of $z(x^n)$ at x^n. If this derivative is close to zero, the algorithm terminates.

The choice of convergence criterion is based on the function to be minimized, the particular algorithms used, and the context in which the problem is solved. This point is discussed in later chapters in connection with some specific problems. The next paragraphs present some of the common curve-fitting algorithms. As mentioned above these algorithms differ from each other in the method used to approximate $z(x)$.

Newton's method. Newton's method approximates $z(x)$ at each iteration with a quadratic fit at the current point, x^n. The fitted curve, $\tilde{z}(x)$, is given by

$$\tilde{z}(x) = z(x^n) + \frac{dz(x^n)}{dx}(x - x^n) + \frac{1}{2}\frac{d^2z(x^n)}{dx^2}(x - x^n)^2 \qquad [4.4a]$$

This curve, $\tilde{z}(x)$, is a parabola that agrees with $z(x)$ at x^n up to second derivatives. The next solution, x^{n+1}, is located at the point that minimizes $\tilde{z}(x)$ [i.e.,

†These rules have been devised for descent method, that is, for algorithms in which $z(x^1) > z(x^2) > \cdots > z(x^n) > \cdots \geq z(x^*)$. While curve-fitting algorithms are not strictly descent methods (why?), they tend to behave as if they were after the first few iterations, and thus the convergence criteria above do apply.

where $d\hat{z}(x)/dx = 0$]. This point is given by

$$x^{n+1} = x^n - \frac{dz(x^n)/dx}{d^2z(x^n)/dx^2} \qquad [4.4b]$$

Equation [4.4b] specifies, then, the algorithmic step, that is, the rule by which x^{n+1} is obtained from x^n. (Equation [4.4a] is included here only to explain the basis of this rule; it need not be evaluated when the algorithm is executed.)

Newton's method is very efficient, but it requires, of course, that $z(x)$ be twice differentiable. It also requires that the first and second derivatives be evaluated at every iteration. In some cases this may be very costly in terms of computational effort.

False position method. The false position method is similar to Newton's search. The only difference is that, instead of using the second derivative at x^n, it uses the first derivative at x^{n-1} and at x^n to approximate the second derivative. The approximation is

$$\frac{d^2z(x^n)}{dx^2} \simeq \frac{dz(x^{n-1})/dx - dz(x^n)/dx}{x^{n-1} - x^n} \qquad [4.5]$$

This expression can be substituted for the second derivative in Eq. [4.4b] to obtain the algorithmic step of the false position method. This method can be used when the second derivatives are unavailable or difficult to evaluate.

Quadratic fit method. A third curve-fitting method uses no derivatives at all. The quadratic fit is based on three points, x_1, x_2, and x_3. At the nth iteration, these points are a subset of the series of solution points $x^1, \ldots,$ x^n. As with the other curve-fitting methods, the new solution point is chosen at the minimum of the fitted curve. This point is given by

$$x^{n+1} = \frac{1}{2} \frac{(x_2^2 - x_3^2)z(x_1) + (x_3^2 - x_1^2)z(x_2) + (x_1^2 - x_2^2)z(x_3)}{(x_2 - x_3)z(x_1) + (x_3 - x_1)z(x_2) + (x_1 - x_2)z(x_3)} \qquad [4.6]$$

The set of three points used in the next iteration includes x^{n+1}, and the two points out of x_1, x_2, and x_3 with the smallest value of $z(x)$. These points are labeled x_1, x_2, and x_3 and the procedure is repeated (see Problem 4.9).

For a given problem, the choice among the various curve-fitting methods should be guided by the difficulty of calculating the derivatives. If this calculation can be easily accomplished, Newton's method is the appropriate algorithm, whereas the quadratic fit method is preferred when even the first derivative is difficult to evaluate. In general, curve-fitting methods are faster than interval reduction methods when applied to functions that are relatively smooth. Curve-fitting methods should also be used in cases where the interval within which the minimum lies is not obvious (e.g., if the function to be minimized is unconstrained).

This concludes the discussion of algorithms for minimizing single-variable functions. The next section describes some approaches to the minimization of multivariable functions.

4.2 MULTIVARIATE MINIMIZATION†

This section deals with the minimization of a nonlinear convex function of several variables. The algorithms included here are all descent methods. In each case, the algorithm generates a point $\mathbf{x}^{n+1} = (x_1^{n+1}, \ldots, x_I^{n+1})$ from $\mathbf{x}^n = (x_1^n, \ldots, x_I^n)$, so that $z(\mathbf{x}^{n+1}) < z(\mathbf{x}^n)$. The focus of the discussion is on some principles of minimization. Particular procedures are mentioned only to illustrate these principles.‡

The core of any algorithmic procedure is the calculation of $\mathbf{x}^{n+1}$ from $\mathbf{x}^n$. This algorithmic step can be written in standard form as

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \alpha_n \mathbf{d}^n \qquad [4.7a]$$

where $\mathbf{d}^n$ is a descent direction vector $(d_1^n, \ldots, d_I^n)$ and α_n is a nonnegative scalar known as the *move size* (or "step size"). This formula means that at point $\mathbf{x}^n$ a direction in which the function is (locally) decreasing, $\mathbf{d}^n$, is identified. The move size, α_n, determines how far along $\mathbf{d}^n$ the next point, $\mathbf{x}^{n+1}$, will be.§ As Eq. [4.7a] is executed, each component of the variable vector is updated, that is,

$$x_i^{n+1} = x_i^n + \alpha_n d_i^n \qquad \text{for } i = 1, 2, \ldots, I \qquad [4.7b]$$

Consider, for example, the function $z(x_1, x_2) = (x_1 - 6)^2 + 3(x_2 - 4)^2$ depicted in Figure 4.4 and assume the current solution to be $\mathbf{x}^n = (3, 1)$. The dashed line in the figure is a tangent to the contour of $z(\mathbf{x})$ at $\mathbf{x}^n$. This line separates the directions of descent at $\mathbf{x}^n$ from the directions of ascent. The figure depicts several descent directions (denoted $\mathbf{d}^n$) emanating from $\mathbf{x}^n$, and also shows the direction of the gradient [denoted by $\nabla z(3, 1)$] at this point. The value of $z(\mathbf{x})$ at any point, $\mathbf{x}$, located a small distance away from $\mathbf{x}^n$ along one of the descent directions, is smaller than $z(\mathbf{x}^n)$, the value of the function at $\mathbf{x}^n$. Note that the gradient vector is perpendicular to the tangent of the contour line at $\mathbf{x}^n$. Consequently, any descent direction can be characterized by the inequality $\nabla \mathbf{z} \cdot \mathbf{d} < 0$. In other words, the cosine of the angle between the

†Readers who are not interested in the principles of minimization algorithms may skip this section and go directly to the description of the convex combinations algorithm in Section 4.3.

‡Proofs of convergence are not within the scope of this text. Several of the references mentioned in Section 4.4 contain such proofs. In particular, Zangwill's "global convergence theorem" specifies the necessary and sufficient conditions for algorithmic convergence.

§The term "move size" for α_n assumes that the direction vector is normalized so that $\sqrt{d_1^2 + d_2^2 + \cdots + d_I^2} = 1$. This normalization does not have to be performed in practice, meaning that the size of the move will actually be the product of α_n and the magnitude of the direction vector.

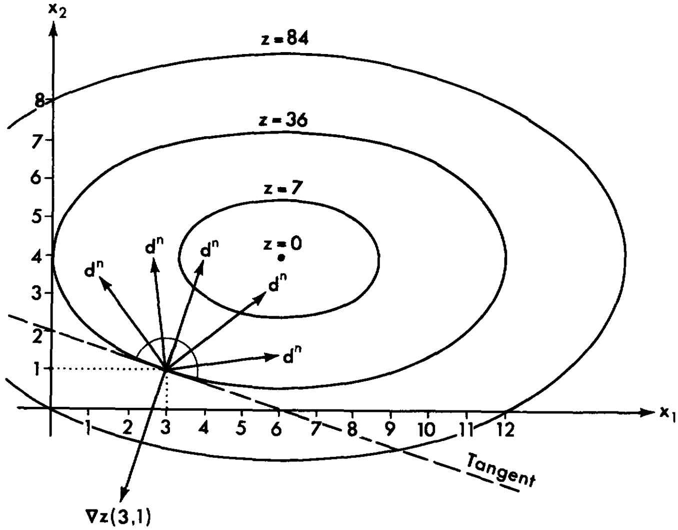


Figure 4.4 Contour lines of $z(x_1, x_2) = (x_1 - 6)^2 + 3(x_2 - 4)^2$ and the range of descent directions at $(x_1, x_2) = (3, 1)$.

gradient direction and the descent direction is always negative. Any direction vector, **d**, is a descent direction if and only if this inequality holds.

The magnitude of the step along the chosen direction is determined by the move size, α_n. If, for example, the chosen direction is $\mathbf{d}^n = (1/\sqrt{2}, 1/\sqrt{2})$, a move of $3\sqrt{2}$ would lead to the optimum in one iteration. (Check this.) If the descent direction is $\mathbf{d}^n = (0, 1)$ [pointing straight up from (3, 1)], any step size that is smaller than $\alpha_n = 6$ will cause a decrease in the objective function [as compared to $z(3, 1) = 36$]. The minimum value of $z(x_1, x_2)$ along this direction is at (3, 4). This point will be attained by using a move size, $\alpha_n = 3$ (verify this graphically).

The algorithmic iteration described in Eq. [4.7] can be used to summarize all descent methods for unconstrained minimization. These methods differ mostly in the rule used for finding the descent direction. The move size is typically chosen so that the objective function is minimized along the descent direction.

If the function to be minimized is constrained, the minimization algorithms have to be modified so that a desired but infeasible direction can be changed according to some appropriate rule. Similarly, if the move size leads outside the feasible region, it has to be truncated so that feasibility is maintained.†

The remainder of this section describes some issues related to mini-

†Note that conserving feasibility is not mandatory, and many problems can be solved without it. The algorithms discussed in this text, though, ensure that the final (optimal) solution is feasible by ensuring that the entire sequence of solution points is feasible.

mization techniques as well as specific descent algorithms for unconstrained and for constrained problems. These algorithms contain principles and ideas that are important to the analysis of network equilibium. The programs to which these algorithms are applied are assumed herein to be convex (that is, having a convex objective function defined over a convex feasible region).

Unconstrained Minimization Algorithms

The convergence criteria used to terminate the algorithmic iterations are an extension of the methods used in one-dimensional search algorithms. For instance, the convergence criterion can be based on the marginal contribution of successive iterations, that is, the closeness of $z(\mathbf{x}^n)$ and $z(\mathbf{x}^{n-1})$. In other words, terminate if

$$[z(\mathbf{x}^{n-1}) - z(\mathbf{x}^n)] \leq \kappa \qquad [4.8a]$$

Alternatively, the algorithm can be terminated if the elements of the gradient vector are close to zero, for example, if†

$$\max_i \left\{ \left| \frac{\partial z(\mathbf{x}^n)}{\partial x_i} \right| \right\} \leq \kappa \qquad [4.8b]$$

In some cases, criteria that are based on the change in the variables between successive iterations are used. These include, for example,

$$\max_i \left\{ \frac{| x_i^n - x_i^{n-1} |}{x_i^{n-1}} \right\} \leq \kappa \qquad [4.8c]$$

or

$$\sum_i (x_i^n - x_i^{n-1})^2 \leq \kappa \qquad [4.8d]$$

In these criteria (Eqs. [4.8]), κ is a predetermined tolerance (different for each criterion) selected especially for each problem based on the desired degree of accuracy.

The focus of the following discussion is on the commonly used steepest descent algorithm.

The method of steepest descent. The method of steepest descent is perhaps the most intuitively logical minimization procedure. The direction of search is opposite to the gradient direction, while the move size is chosen so as to minimize $z(\mathbf{x})$ in that direction. This means that each move is made in the direction in which $z(\mathbf{x})$ decreases (locally) the most (i.e., where the function is steepest). The length of each move is determined by the points (along this direction) where the value of $z(\mathbf{x})$ stops decreasing and starts increasing.

†The notation $\max_i \{\cdot\}$ stands for the maximum, over all possible values of i, of the arguments in the braces.

The basic iteration is given by

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \alpha_n[-\nabla z(\mathbf{x}^n)] \qquad [4.9]$$

where α_n is the optimal move size. The gradient can be found either analytically or, when the derivatives are difficult to evaluate, by using numerical methods. If $\nabla z(\mathbf{x})$ can be derived analytically, it has to be supplied as an input to the minimization routine. Alternatively, it can be computed numerically by approximating the partial derivative of $z(\mathbf{x})$ with respect to each x_i. In other words,

$$\frac{\partial z(\mathbf{x}^n)}{\partial x_i} \simeq \frac{z(\ldots, x_{i-1}^n, x_i^n + \Delta x_i, x_{i+1}^n, \ldots) - z(\ldots, x_i^n, \ldots)}{\Delta x_i} \qquad [4.10]$$

where Δx_i is a small interval associated with the ith component of $\mathbf{x}$. The evaluation of the gradient at $\mathbf{x}^n$ therefore involves I computations of $z(\mathbf{x})$ in addition to $z(\mathbf{x}^n)$.

In order to find the value of α_n, the function

$$z[\mathbf{x}^n + \alpha(-\nabla z(\mathbf{x}^n))] \qquad [4.11]$$

has to be minimized with respect to α subject to the constraint that $\alpha > 0$. The value of α that minimizes Eq. [4.11] is α_n. Finding α_n is a one-dimensional minimization problem that can be solved by using any of the techniques mentioned in Section 4.1. If the problem is simple (and convex), α_n can be determined analytically by solving†

$$\frac{d}{d\alpha} z[\mathbf{x}^n + \alpha(-\nabla z(\mathbf{x}^n))] = 0$$

rather than using a numerical method. Note that the interval reduction methods mentioned in the preceding section cannot be naturally applied here since the problem is not constrained and therefore no initial interval exists. Any one of the curve-fitting methods, however, would be appropriate for this minimization.

The steepest descent algorithm is a descent method, meaning that the objective function value decreases at every iteration. For objective functions satisfying certain regularity conditions this method converges to a local minimum, which would naturally be a global one for strictly convex functions.

Figure 4.5 depicts a typical sequence of solutions generated by the steepest descent algorithm. An interesting property of this method is that the successive search directions are perpendicular to each other, that is, $\mathbf{d}^n \cdot \mathbf{d}^{n+1} = 0$. This follows from the fact that $\mathbf{x}^{n+1}$ is located at a point where the rate of change in $z(\mathbf{x})$ with respect to movement in the direction $\mathbf{d}^n$ is zero, due to the optimization in the line search. Thus the gradient (and its opposite direction) at $\mathbf{x}^{n+1}$ is perpendicular to the direction of $\mathbf{d}^n$.

For a general quadratic form (the contour lines of which are ellipsoids),

†Note that the point $\alpha = 0$ has to be checked as well.

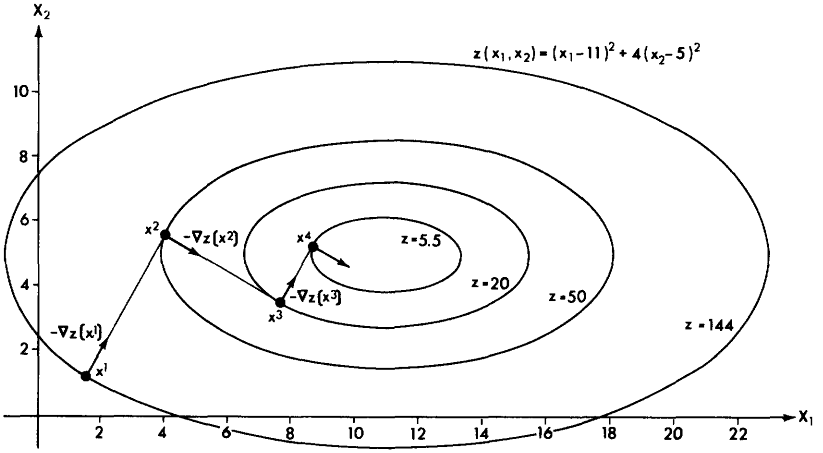


Figure 4.5 Convergence pattern of the steepest descent algorithm; note the zigzagging of the consecutive descent directions.

this "zigzagging" becomes more pronounced as the eccentricity (the ratio of the axes) increases. On the other hand, if the contours are circular, the steepest descent method converges in a single step. The zigzagging means that the steepest descent method requires a relatively large number of iterations. Several heuristic procedures to alleviate this problem are suggested in the references mentioned in Section 4.5. These methods generally choose a direction other than the opposite gradient every few iterations.

Other methods. The operations research literature includes many other minimization approaches including second order methods in which the search direction is determined by using information regarding the second derivatives of the objective function. Most of these methods are not particularly applicable to the solution of the problems arising in the course of studying urban networks. These problems are typically large (i.e., including many variables), a characteristic that often precludes certain calculations such as the determination of the Hessian (which is the matrix of second derivatives) or even approximations of this Hessian. The following paragraphs, then, are provided mostly for completeness.

One of the most efficient minimization algorithms is Newton's method. This method is a straightforward extension of the corresponding line search procedure (described in Section 4.1). The objective function $z(\mathbf{x})$ is approximated by a second-order Taylor series. This approximation is then minimized and the new solution is taken to be the point that minimizes the approximation. The resulting algorithmic step is given by

$$\mathbf{x}^{n+1} = \mathbf{x}^n - \nabla z(\mathbf{x}^n) \cdot [\mathbf{H}(\mathbf{x}^n)]^{-1} \qquad [4.12]$$

where $\mathbf{H}(\mathbf{x}^n)$ is the Hessian of $z(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}^n$, and $[\cdot]^{-1}$ denotes the matrix inversion operator. This method can be expressed in the standard form [4.7a] by defining $\mathbf{d}^n = -\nabla z(\mathbf{x}^n) \cdot [H(\mathbf{x}^n)]^{-1}$ and $\alpha_n = 1, \forall\, n$.†

Newton's method is more efficient than the steepest descent method because its search direction uses more information about the shape of $z(\mathbf{x})$ at $\mathbf{x}^n$. It requires, however, the calculation and inversion of the Hessian matrix at each iteration, a task that is computationally intensive even for medium-size problems, and totally impractical for large ones.

The approach underlying quasi-Newton methods‡ is to try to construct an approximation to the inverse Hessian by using information gathered in the minimization process. The current approximation is used at each step to modify the gradient direction and the process is usually coupled with a line search to determine the optimal move size. The particulars of these methods can be found in the references mentioned at the end of this chapter.

Constrained Minimization Algorithms

The study of transportation network equilibrium problems involves equivalent mathematical programs which include exclusively linear constraints. The discussion of constrained minimization is limited, therefore, to these types of constraints. The techniques reviewed in this section are applicable to both equality and inequality constraints, even though most network problems involve only equality constraints (apart from the nonnegativity constraints).

The basic framework outlined in the preceding section for unconstrained minimization (i.e., finding a descent direction and advancing by an optimal amount along this direction) can be used to describe algorithms for constrained optimization as well. The added difficulty in constrained optimization is to maintain feasibility. In other words, the search direction has to point toward a region where some feasible solutions lie, and the search for the optimal step size has to be constrained to feasible points only. The focus of this section is on these topics, which are common to all *feasible direction* methods. The emphasis on this minimization approach stems from its importance in equilibrium analysis. Most current approaches to minimizing equivalent equilibrium programs can be cast as feasible direction methods. In particular, the convex combinations algorithm, which is the basic tool for many equilibrium analyses, is such a method. This algorithm is described in detail in Section 4.3.

The discussion in this section does not mention explicitly convergence criteria since those can be similar to the ones used for unconstrained mini-

†Note that in order to ensure that Newton's is a descent method, especially at the initial iterations, the move size has to be optimized. In such cases α_n will not equal 1.

‡These methods are also called variable metric approaches. The name stems from a particular interpretation of the technique used to define the search direction—it can be seen as an effort to change the scale of the axes of the vector $\mathbf{x}$ so that $z(\mathbf{x})$ is close to a spherical shape and the gradient points in the right direction.

mization (see Eqs. [4.8]). Note only that the minimum of a constrained program is not necessarily a stationary point of the objective function, and thus a gradient-based convergence criterion (such as Eq. [4.8b]) is not applicable.

As mentioned above, the two issues associated with maintaining feasibility are that the search direction has to point in a direction in which at least some feasible points lie and that, given such a search direction, the step size would not cause infeasibility. These two issues are described below in reverse order. First the problem of maintaining feasibility given the search direction and then the problem of finding a feasible direction.

Maintaining feasibility in a given direction. Consider the standard form of a minimization program (see Chapter 2)

$$\min z(\mathbf{x}) \qquad [4.13a]$$

subject to

$$\sum_i h_{ij} x_i \geq b_j \qquad \forall\, j \in \mathscr{J} \qquad [4.13b]$$

where the constraints are all linear (the nonnegativity constraints are included in Eqs. [4.13]). Assume further that the current solution is $\mathbf{x}^n$, a point that may lie on the boundary of the feasible region. In this case some of the constraints are binding, that is,

$$\sum_i h_{ij} x_i^n = b_j \qquad \forall\, j \in \mathscr{J}^n \qquad [4.14a]$$

where $\mathscr{J}^n$ is the index set of the binding constraints at the nth iteration. The other constraints are not binding, that is,

$$\sum_i h_{ij} x_i^n > b_j \qquad \forall\, j \notin \mathscr{J}^n \qquad [4.14b]$$

Once a feasible descent direction, $\mathbf{d}^n$, is obtained, the maximum move size that can be taken without violating any constraints should be determined. This maximum can then be used to bracket the search along the descent direction, so the new solution will necessarily be feasible. Note that only the constraints which are not currently binding (i.e., the constraints indexed by $j \notin \mathscr{J}^n$) might pose a problem in this regard (why?—see Problem 4.17) and thus only the inequality constraints (all $j \notin \mathscr{J}^n$) need be considered. For $\mathbf{x}^{n+1} = \mathbf{x}^n + \alpha_n \mathbf{d}^n$ to be feasible, the (currently) nonbinding constraints must not be violated by the new solution, $\mathbf{x}^{n+1}$. In other words, the following must hold:

$$\sum_i h_{ij}(x_i^n + \alpha_n d_i^n) = \sum_i h_{ij} x_i^n + \alpha_n \sum_i h_{ij} d_i^n \geq b_j \qquad \forall\, j \notin \mathscr{J}^n$$

Obviously, if $\sum_i h_{il} d_i^n \geq 0$ for a given constraint l, there is no problem in satisfying the above mentioned requirement, since the current solution is feasible (i.e., $\sum_i h_{il} x_i^n \geq b_l$). (Intuitively, this means that the constraints for which $\sum_i h_{ij} d_i^n \geq 0$ will not be violated by the search direction.) Only in the cases where $\sum_i h_{ij} d_i^n < 0$ would there be a possible violation. Thus the maximum possible value of α_n, $\alpha_n^{\max}$, is given by

$$\alpha_n^{\max} = \min_{\forall j \in \mathcal{J}^n} \frac{b_j - \sum_i h_{ij} x_i^n}{\sum_i h_{ij} d_i^n} \qquad [4.15]$$

where the set $\mathcal{J}^n$ includes all the constraints that are nonbinding and for which $\sum_i h_{ij} d_i^n < 0$ at $\mathbf{x}^n$. In other words, the maximum move size is determined by the first constraint to be violated when moving along the descent direction. The optimal move size, α_n, can now be determined by solving

$$\min_{\alpha} z(\mathbf{x}^n + \alpha \mathbf{d}^n) \qquad [4.16a]$$

subject to

$$0 \le \alpha \le \alpha_n^{\max} \qquad [4.16b]$$

with any of the interval reduction methods of Section 4.1.

Figure 4.6 demonstrates the various types of constraints involved in this situation. It shows a feasible region bounded by five constraints (numbered 1 through 5). The current solution is on the boundary defined by constraint 1 and the descent direction points into the feasible region as shown in the figure. The nonbinding constraints are 2 through 5 but constraints 2 and 5 need not be considered since in the given search direction they will not be violated. (These are constraints for which $\sum_i h_{ij} d_i^n \ge 0$.) The move size is determined by checking which of the remaining constraints (3 or 4) will be violated first. In this case, it is constraint 4 that sets the limit, $\alpha_n^{\max}$, for the line search that follows.

Finding a feasible descent direction. Consider now the problem of determining a feasible descent direction. At some iterations this may not be a problem since a search direction identified by the algorithm used (which may

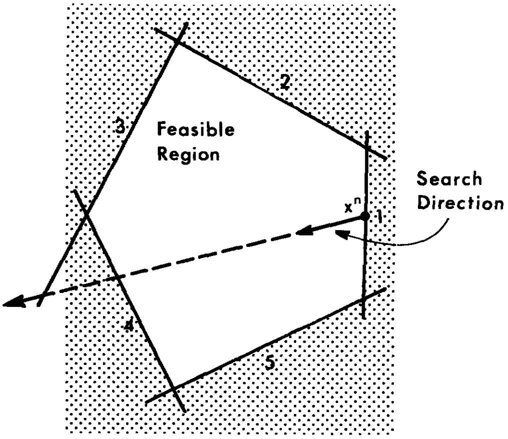


Figure 4.6 Maintaining feasibility along a search direction by bracketing the range of the line search.

be steepest descent, Newton, quasi-Newton, or any other method) may be feasible. In other cases, however, where the current solution is on the boundary of the feasible region, the best descent direction may be pointing outside that region. In such cases the direction has to be modified and the best (in some sense) *feasible* descent direction should be used to continue the search.

In looking for a feasible direction of descent from $\mathbf{x}^n$, only the binding constraints need be considered. Those constraints that are not currently binding would not be violated by a small (infinitesimal) move in any direction and should not, therefore, be considered. Given a candidate descent direction $\mathbf{d} = (\ldots, d_i, \ldots)$, the following must hold in order to ensure feasibility:

$$\sum_i h_{ij} x_i^n + \Delta\alpha \sum_i h_{ij} d_i \geq b_j \qquad \forall\, j \in \mathscr{J}^n \qquad [4.17a]$$

where $\Delta\alpha$ is a small move size. Since $\sum_i h_{ij} x_i^n = b_j$ for all $j \in \mathscr{J}^n$, expression [4.17a] reduces to

$$\sum_i h_{ij} d_i \geq 0 \qquad \forall\, j \in \mathscr{J}^n \qquad [4.17b]$$

Condition [4.17b] should be satisfied by any feasible direction.

The direction offering the steepest feasible descent can be found by solving the program

$$\min \nabla z(\mathbf{x}^n) \cdot \mathbf{d}^T \qquad [4.18a]$$

subject to

$$\sum_i h_{ij} d_i \geq 0 \qquad \forall\, j \in \mathscr{J}^n \qquad [4.18b]$$

$$\sum_i d_i^2 = 1 \qquad [4.18c]$$

The objective function [4.18a] is the cosine of the angle between the gradient and the descent direction.† This program then finds the descent direction that is closest to the opposite gradient direction (the cosine is minimized for an angle of 180°) yet satisfies the feasibility requirements. The last constraint normalizes the descent direction vector in order to ensure a unique solution. It is, unfortunately, a nonlinear constraint, meaning that program [4.18] may be difficult to solve.‡

The solution of this program is the steepest feasible descent vector. This vector defines the direction in which the search for the next solution, $\mathbf{x}^{n+1}$, should be conducted. (As explained above, this search may have to be bracketed first to ensure the feasibility of $\mathbf{x}^{n+1}$.)

The literature includes many other algorithms for determining a "good"

†Since $\mathbf{d}$ is constrained to be a unit vector, the objective function can be interpreted as the slope of $z(\mathbf{x})$ at $\mathbf{x}^n$, in the direction $\mathbf{d}$.

‡The references mentioned in Section 4.5 include some descriptions of approximate methods to determine the best feasible direction. For example, one method approximates constraint [4.18c] by the constraints $-1 \leq d_i \leq 1, \forall\, i$, which is linear and therefore easier to deal with.

feasible descent method. As an example of such methods, consider the gradient projection method, which obtains the next solution by "sliding" along the binding constraints at the current point. This strategy is, of course, followed only if the current descent direction (which is the direction opposite the gradient) points out of the feasible region (otherwise, there is no problem and the algorithm proceeds along the opposite gradient direction). The sliding direction is identified by projecting the opposite gradient direction on the binding constraints. The projection procedure itself includes some matrix manipulations involving the binding constraints. The move size is optimized along the descent direction, subject to the requirement that none of the (currently nonbinding) constraints is violated.

4.3 CONVEX COMBINATIONS METHOD

The convex combination algorithm was orginally suggested by Frank and Wolfe in 1956 as a procedure for solving quadratic programming problems with linear constraints and is known also as the Frank–Wolfe (FW) method. The method is especially useful for determining the equilibrium flows for transportation networks, which is why it is emphasized in this chapter.

Algorithm

The convex combinations algorithm is a feasible direction method. Unlike the general procedure for feasible direction methods described in the preceding section, the bounding of the move size does not require a separate step (such as Eq. [4.15]) with this algorithm. The bounding is accomplished as an integral part of the choice of descent direction. The direction-finding step of the convex combinations algorithm is explained in this section from two angles. First, this step is explained by using the logic of the general procedure for feasible direction methods outlined in the preceding section, and second, this step is presented as a linear approximation method.

To look at the direction-finding step of the convex combination algorithm as that of a feasible direction method, consider the convex program

$$\min z(\mathbf{x}) \tag{4.19a}$$

subject to

$$\sum_i h_{ij} x_i \geq b_j \qquad \forall j \in \mathscr{J} \tag{4.19b}$$

Assume now that at the nth iteration, the current solution is $\mathbf{x}^n$. Most feasible direction methods use information about the shape of the objective function in the vicinity of $\mathbf{x}^n$ to determine the descent direction. Consequently, the descent direction is based on the opposite gradient direction or the direction of the (locally) steepest feasible descent.

The convex combinations method selects the (feasible) descent direction not only on the basis of how steep each candidate direction is in the vicinity of $\mathbf{x}^n$, but also according to how far it is possible to move along this direction. If, for example, only a small move is feasible in a certain direction, effort might be wasted in actually performing an iteration based on such a move. Even though the direction may be associated with a steep *local* decrease in $z(\mathbf{x})$, the overall reduction in the objective function from $z(\mathbf{x}^n)$ to $z(\mathbf{x}^{n+1})$ may not be significant. On the other hand, a direction of change in which the local rate of improvement is modest but where the feasible region allows a considerable movement may achieve a larger reduction. The criterion for choosing directions in the convex combinations method is therefore based on the product of the rate of descent in the vicinity of $\mathbf{x}^n$ in a given direction and the length of the feasible region in that direction. This product, known as the "drop," is an upper bound to the possible reduction in the objective function value which can be achieved by moving in this direction. The algorithm uses the direction that maximizes the drop.

To find a descent direction, the algorithm looks at the entire feasible region for an auxiliary feasible solution, $\mathbf{y}^n = (y_1^n, \ldots, \ldots y_1^n)$, such that the direction from $\mathbf{x}^n$ to $\mathbf{y}^n$ provides the maximum drop. The direction from $\mathbf{x}^n$ to any feasible solution, $\mathbf{y}$, is the vector $(\mathbf{y} - \mathbf{x}^n)$ [or the unit vector $(\mathbf{y} - \mathbf{x}^n)/\|\mathbf{y} - \mathbf{x}^n\|$].† The slope of $z(\mathbf{x}^n)$ in the direction of $(\mathbf{y} - \mathbf{x}^n)$ is given by the projection of the opposite gradient $[-\nabla z(\mathbf{x}^n)]$ in this direction, that is,

$$-\nabla z(\mathbf{x}^n) \cdot \frac{(\mathbf{y} - \mathbf{x}^n)^T}{\|\mathbf{y} - \mathbf{x}^n\|}$$

The drop in the objective function in the direction $(\mathbf{y} - \mathbf{x}^n)$ is obtained by multiplying this slope by the distance from $\mathbf{x}^n$ to $\mathbf{y}$, $\|\mathbf{y} - \mathbf{x}^n\|$. The result is

$$-\nabla z(\mathbf{x}^n) \cdot (\mathbf{y} - \mathbf{x}^n)^T$$

This expression has to be maximized (in $\mathbf{y}$) subject to the feasibility of $\mathbf{y}$. Alternatively, the expression can be multiplied by (-1) and minimized, resulting in the program

$$\min \nabla z(\mathbf{x}^n) \cdot (\mathbf{y} - \mathbf{x}^n)^T = \sum_i \frac{\partial z(\mathbf{x}^n)}{\partial x_i} (y_i - x_i^n) \qquad [4.20a]$$

subject to

$$\sum_i h_{ij} y_i \geq b_j \qquad \forall j \in \mathcal{J} \qquad [4.20b]$$

where the constraints [4.20b] are equivalent to the original constraint set [4.19b] expressed in $\mathbf{y}$. The solution of program [4.20] is $\mathbf{y}^n$ and the descent direction is the vector pointing from $\mathbf{x}^n$ to $\mathbf{y}^n$, that is, $\mathbf{d}^n = (\mathbf{y}^n - \mathbf{x}^n)$, or in expanded form, $d_i^n = y_i^n - x_i^n$, $\forall\ i$.

†The notation $\|\mathbf{v}\|$ means the norm (or length) of the vector $\mathbf{v}$, that is, $\sqrt{\mathbf{v} \cdot \mathbf{v}} = \sqrt{\sum_i v_i^2}$.

Finding the descent direction, then, involves a minimization of a linear program, which is given in Eqs. [4.20]. As mentioned in the beginning of this section, this algorithmic step can also be motivated by looking at the convex combination method as a linear approximation method. This approach is based on finding a descent direction by minimizing a linear approximation to the function (instead of the function itself) at the current solution point. Minimizing this linearized function subject to a linear constraint set is a linear programming problem which has its solution at a corner of the feasible region (see Section 2.3). The line connecting the current solution points, $\mathbf{x}^n$, with the solution of the linearized problem (denoted $\mathbf{y}^n$) is the direction of search. This approach results in the same program as Eqs. [4.20]. To see this, let $z_L^n(\mathbf{y})$ denote the linear approximation of the value of the objective function at some point $\mathbf{y}$, based on its value at $\mathbf{x}^n$. This approximation is given by

$$z_L^n(\mathbf{y}) = z(\mathbf{x}^n) + \nabla z(\mathbf{x}^n) \cdot (\mathbf{y} - \mathbf{x}^n)^T \qquad [4.21]$$

This linear function of $\mathbf{y}$ has to be minimized subject to the constraints of the original problem, that is,

$$\min z_L^n(\mathbf{y}) = z(\mathbf{x}^n) + \nabla z(\mathbf{x}^n) \cdot (\mathbf{y} - \mathbf{x}^n)^T \qquad [4.22a]$$

subject to

$$\sum_i h_{ij} y_i \geq b_j \qquad \forall j \in \mathscr{J} \qquad [4.22b]$$

Note that, at the point $\mathbf{x} = \mathbf{x}^n$, the value of the objective function is constant and thus $z(\mathbf{x}^n)$ can be dropped from Eq. [4.22a]. The problem that remains is that of minimizing $\nabla z(\mathbf{x}^n) \cdot (\mathbf{y} - \mathbf{x}^n)^T$ subject to constraints [4.22b]. This program is identical to program [4.20], and its solution (again) can be used to construct a direction for the convex combinations algorithm. Thus both points of view can be used to motivate the algorithmic step of the convex combinations method.

The objective function of the linearized problem, can, however, be simplified even further by noting that $\nabla z(\mathbf{x}^n)$ is constant at $\mathbf{x}^n$ and the term $\nabla z(\mathbf{x}^n) \cdot (\mathbf{x}^n)^T$ can therefore be dropped from the linearized program (see Eq. [4.20a]), which can then be written as

$$\min z^n(\mathbf{y}) = \nabla z(\mathbf{x}^n) \cdot \mathbf{y}^T = \sum_i \left(\frac{\partial z(\mathbf{x}^n)}{\partial x_i}\right) y_i \qquad [4.23a]$$

subject to

$$\sum_i h_{ij} y_i \geq b_j \qquad \forall j \in \mathscr{J} \qquad [4.23b]$$

The variables of this linear program are $y_1, y_2, \ldots, y_I$ and the objective function coefficients are $\partial z(\mathbf{x}^n)/\partial x_1, \partial z(\mathbf{x}^n)/\partial x_2, \ldots, \partial z(\mathbf{x}^n)/\partial x_I$. These coefficients are the derivatives of the original objective function at $\mathbf{x}^n$, which are known at this point. The solution of program [4.23], $\mathbf{y}^n = (y_1^n, y_2^n, \ldots, y_I^n)$, is used to define the descent direction (i.e., $\mathbf{d}^n = \mathbf{y}^n - \mathbf{x}^n$), as explained above.

Once the descent direction is known, the other algorithmic steps involve the determination of the move size and a convergence test.

As in many other descent methods, the move size in the direction of $\mathbf{d}^n$ equals the distance to the point along $\mathbf{d}^n$ which minimizes $z(\mathbf{x})$. As mentioned before, the convex combinations methods does not require a special step to bracket the search for an optimal move size (such as Eq. [4.15]) in order to maintain feasibility. The new solution, $\mathbf{x}^{n+1}$, must lie between $\mathbf{x}^n$ and $\mathbf{y}^n$ (since $\mathbf{y}^n$, being a solution of a linear program, naturally lies at the boundary of the feasible region). In other words, the search for a descent direction automatically generates a bound for the line search by accounting for all the constraints (not only the binding ones) when the descent direction is determined. Since the search interval is bracketed, then, any one of the interval reduction methods would be suitable for the minimization of $z(\mathbf{x})$ along $\mathbf{d}^n = (\mathbf{y}^n - \mathbf{x}^n)$, that is, for solving

$$\min z[\mathbf{x}^n + \alpha(\mathbf{y}^n - \mathbf{x}^n)] \qquad [4.24a]$$

subject to

$$0 \leq \alpha \leq 1 \qquad [4.24b]$$

Once the optimal solution of this line search, α_n, is found, the next point can be generated with the usual step,

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \alpha_n(\mathbf{y}^n - \mathbf{x}^n) \qquad [4.25]$$

Note that Eq. [4.25] can be written as $\mathbf{x}^n = (1 - \alpha_n)\mathbf{x}^n + \alpha_n \mathbf{y}^n$. The new solution is thus a *convex combination* (or weighted average) of $\mathbf{x}^n$ and $\mathbf{y}^n$.

The convergence criterion used in conjunction with the convex combinations algorithm can be any one of the criteria commonly used for feasible descent methods. Thus the convergence criterion can be based on the similarity of two successive solutions or the reduction of the objective function values between successive iterations.

Two general comments are in order here. First, as explained above, the convex combinations algorithm involves a minimization of a linear program as part of the direction-finding step. The convex combinations algorithm, then, is useful only in cases in which this linear program can be solved relatively easily. It is also useful when algorithms which are generally more efficient than the convex combination method (e.g., Newton and quasi-Newton methods) cannot be utilized due to the size of the problem. Many of the minimization problems described in this book possess both properties: they include a large number of variables, yet the linear program associated with the direction finding step can be solved with particular ease. The reason is that this program, in the cases dealt with in this text, has a special structure to it—that of a network—which can be exploited to facilitate the solution of the linear program.

The second comment concerns an interesting characteristic of the convex

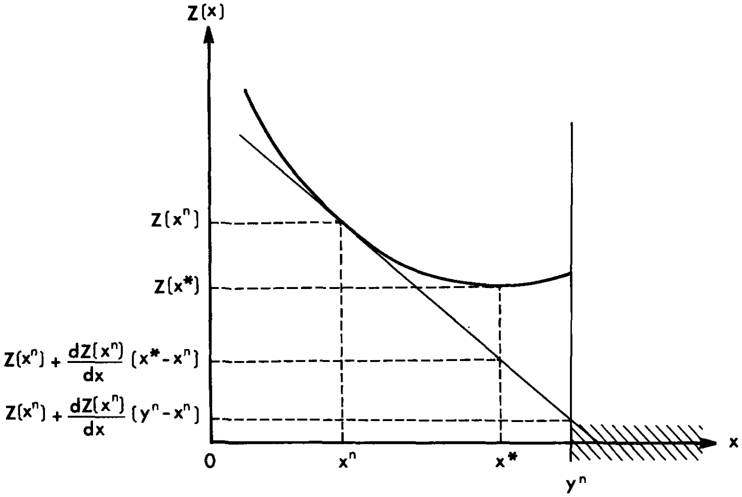


Figure 4.7 The bounds generated by a linear approximation to a function $z(x)$ at a point x^n.

combinations algorithm. In this method, the value of the linearized objective function at its solution, $z_L^n(\mathbf{y}^n)$, is a lower bound to the optimal value of the objective function itself, $z(\mathbf{x}^*)$. To see this, recall that by convexity,

$$z(\mathbf{x}^*) \geq z(\mathbf{x}^n) + \nabla z(\mathbf{x}^n) \cdot (\mathbf{x}^* - \mathbf{x}^n)^T = z_L^n(\mathbf{x}^*) \qquad [4.26a]$$

However, $\mathbf{y}^n$ minimizes $z_L^n(\mathbf{y})$ for any feasible $\mathbf{y}$. Consequently,

$$z_L^n(\mathbf{x}^*) \geq z_L^n(\mathbf{y}^n) = z(\mathbf{x}^n) + \nabla z(\mathbf{x}^n) \cdot (\mathbf{y}^n - \mathbf{x}^n)^T \qquad [4.26b]$$

Thus $z_L^n(\mathbf{y}^n)$ is a lower bound for $z(\mathbf{x}^*)$ at every iteration. Note that this lower bound generated by the convex combinations method is a straightforward extension of the lower bound mentioned in Section 4.1 for interval reduction methods. Figure 4.7 illustrates the relationships in Eqs. [4.26] for a one-dimensional convex function, $z(x)$. Note also that this lower bound is not monotonically increasing from iteration to iteration and therefore cannot be used in a straightforward fashion to create a convergence criterion (see Problem 4.21).

Given a current feasible solution, $\mathbf{x}^n$, the nth iteration of the convex combinations algorithm can be summarized as follows:

Step 1: *Direction finding.* Find $\mathbf{y}^n$ that solves the linear program [4.23].

Step 2: *Step-size determination.* Find α_n that solves

$$\min_{0 \leq \alpha \leq 1} z[\mathbf{x}^n + \alpha(\mathbf{y}^n - \mathbf{x}^n)]$$

Step 3: *Move.* Set $\mathbf{x}^{n+1} = \mathbf{x}^n + \alpha_n(\mathbf{y}^n - \mathbf{x}^n)$.

Step 4: *Convergence test.* If $z(\mathbf{x}^n) - z(\mathbf{x}^{n+1}) \leq \kappa$, stop.† Otherwise, let $n := n + 1$ and go to step 1.

Starting with a feasible solution, $\mathbf{x}^0$, the algorithm will converge after a finite number of iterations.

Example

Consider the following mathematical program:

$$\min z(\mathbf{x}) = x_1^2 + 2x_2^2 - 2x_1x_2 - 10x_2$$

subject to

$$0 \leq x_1 \leq 4$$

$$0 \leq x_2 \leq 6$$

Application of the convex combinations method to this program can be facilitated by some preliminary calculations. The gradient of $z(\mathbf{x})$ at $\mathbf{x}^n$ is given by

$$\nabla z(x_1, x_2) = [(2x_1^n - 2x_2^n), (4x_2^n - 2x_1^n - 10)]$$

The linear program at the nth iteration is thus

$$\min z^n(\mathbf{y}) = (2x_1^n - 2x_2^n)y_1 + (4x_2^n - 2x_1^n - 10)y_2$$

subject to

$$0 \leq y_1 \leq 4$$

$$0 \leq y_2 \leq 6$$

Since this problem is small, the optimal move size can be determined analytically by setting

$$\frac{dz[\mathbf{x}^n + \alpha(\mathbf{y}^n - \mathbf{x}^n)]}{d\alpha} = 0$$

Given $\mathbf{x}^n$ and $\mathbf{y}^n$, the optimal move size is, as the reader can verify,

$$\alpha_n = \frac{(y_1^n - x_1^n)(x_1^n - x_2^n) + (y_2^n - x_2^n)(2x_2^n - x_1^n - 5)}{2(y_1^n - x_1^n)(y_2^n - x_2^n) - 2(y_2^n - x_2^n)^2 - (y_1^n - x_1^n)^2}$$

This concludes the preliminary calculations and the algorithm can now be executed. The convergence criterion is set, for example, at 0.1, terminating the procedure when $z(\mathbf{x}^n) - z(\mathbf{x}^{n+1}) \leq 0.1$. The algorithmic steps are depicted in Figure 4.8 in relation to the feasible region and the objective function. These steps are explained below.

†Other convergence criteria could also be used.

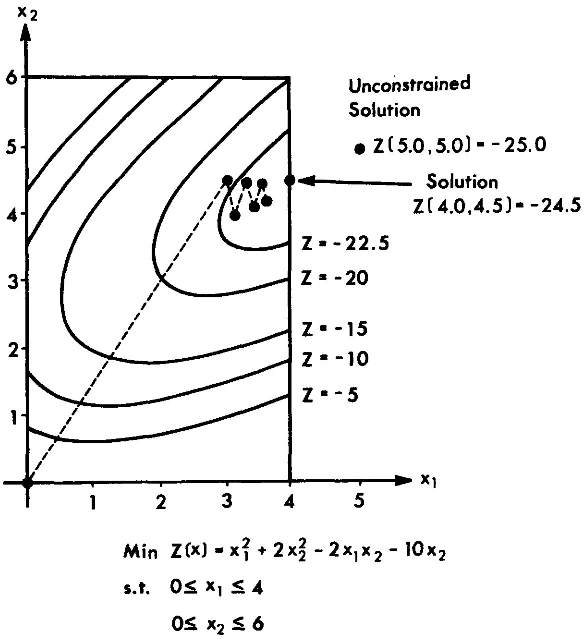


Figure 4.8 Convergence pattern of the convex combinations algorithm for a quadratic function with linear constraints.

Starting with the feasible solution $\mathbf{x}^1 = (0, 0)$ and $z(0, 0) = 0$, the first iteration goes as follows:

First iteration:

Step 1: The gradient is $\nabla z(0, 0) = (0, -10)$. The linear program is

$$\min \nabla z(\mathbf{x}^1)\cdot \mathbf{y} = -10y_2$$

subject to

$$0 \le y_1 \le 4$$

$$0 \le y_2 \le 6$$

Solution (by inspection of Figure 4.8): $\mathbf{y}^1 = (0, 6)$ or $\mathbf{y}^1 = (4, 6)$. Choose $\mathbf{y}^1 = (4, 6)$.

Step 2:

$$\alpha_1 = \frac{(4-0)(0-0) + (6-0)(2\cdot 0 - 0 - 5)}{2(4-0)(6-0) - 2(6-0)^2 - (4-0)^2} = 0.750$$

Step 3:

$$x_1^2 = 0 + 0.750(4 - 0) = 3$$

$$x_2^2 = 0 + 0.750(6 - 0) = 4.5$$

Step 4:

$$z(3, 4.5) = -22.5$$

$$z(\mathbf{x}^1) - z(\mathbf{x}^2) = 0 - (-22.5) = 22.5$$

Second iteration:

Step 1: $\nabla z(3, 4.5) = (-3.0, 2.0)$. The linear program is

$$\min \nabla z(\mathbf{x}^2) \cdot \mathbf{y} = -3y_1 + 2y_2$$

subject to

$$0 \leq y_1 \leq 4$$

$$0 \leq y_2 \leq 6$$

Solution (by inspection): $\mathbf{y}^2 = (4, 0)$.

Step 2: $\alpha_2 = 0.119$.
Step 3:

$$x_1^3 = 3.0 + 0.119(4.0 - 3.0) = 3.119$$

$$x_2^3 = 4.5 + 0.119(0.0 - 4.5) = 3.966$$

Step 4:

$$z(3.119, 3.966) = -23.213$$

$$z(\mathbf{x}^2) - z(\mathbf{x}^3) = 0.713$$

Obviously, the convergence criterion is not met and the algorithm continues. Table 4.2 shows the sequence of iterations of the convex combinations algorithm for this example. As evident in the table, the convergence criterion is

TABLE 4.2 Iterations of the Convex Combinations Algorithm (see Figure 4.8)

n	$\nabla z(\mathbf{x}^n)$	$\mathbf{y}^n$	$z_L^n(\mathbf{y}^n)$	α_n	$\mathbf{x}^{n+1}$	$z(\mathbf{x}^{n+1})$	$z(\mathbf{x}^n) - z(\mathbf{x}^{n+1})$
0					(0.000, 0.000)		
1	(0.000, −10.000)	(4, 6)	−60.000	0.750	(3.000, 4.500)	−22.500	22.500
2	(−3.000, 2.000)	(4, 0)	−35.213	0.119	(3.119, 3.969)	−23.213	0.713
3	(−1.693, −0.376)	(4, 6)	−24.211	0.206	(3.301, 4.385)	−23.446	0.233
4	(−2.169, 0.939)	(4, 0)	−29.257	0.063	(3.344, 4.111)	−23.622	0.176
5	(−1.833, −0.295)	(4, 6)	−25.196	0.144	(3.439, 4.383)	−23.728	0.106
6	(−1.889, 0.656)	(4, 0)	−27.752	0.045	(3.464, 4.186)	−23.816	0.089

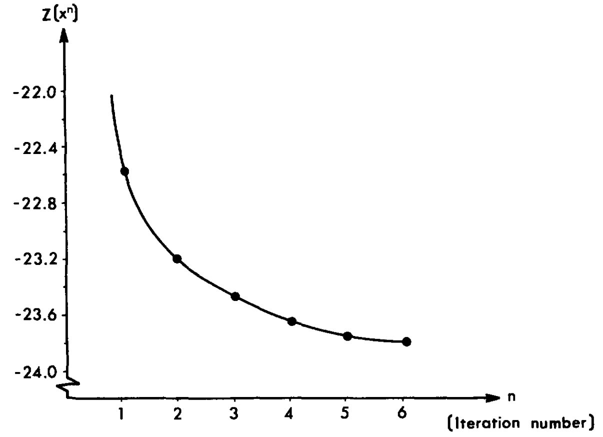


Figure 4.9 The asymptotic reduction in the objective function value as the iterations of the convex combinations algorithm proceed.

met after six iterations and the algorithm terminates. Note that, as mentioned before, the lower bound generated at each step, $z_L^n(\mathbf{y}^n)$, is not monotonically increasing.

Figure 4.8 depicts the pattern of convergence toward the minimum. This convergence pattern is demonstrated in Figure 4.9 in terms of the reduction in the value of the objective function from iteration to iteration. The asymptotic pattern shown in this figure is typical of the convex combinations algorithm. The marginal contribution of each successive iteration becomes smaller and smaller as the algorithm proceeds (this property was the basis for the convergence criterion used in the example).

4.4 SUMMARY

This chapter reviews numerical minimization methods. It begins with a study of line search algorithms, that is, methods for minimizing a function of a single argument. The presentation covers two groups of methods: interval reduction algorithms and quadratic fit approaches. Quadratic fit methods are usually faster if the objective function is relatively smooth. Interval reduction methods are naturally useful in minimizing a function over a given interval.

The review of minimization algorithms for multivariable programs deals separately with unconstrained and constrained minimizations. Most of the methods available for minimizing an unconstrained function of several variables are based on choosing a descent direction at a current solution and moving along this direction to the next solution point. The steepest descent method uses the opposite gradient as the descent direction and then chooses the next solution at the point that minimizes the objective function along this direction. This method often requires a large number of iterations to converge due to its "zigzagging" tendency. Many of the algorithms that require smaller number of iterations cannot be used for the problems covered in this book since they require computation of the Hessian (or an approximation of it). Such requirements are computationally prohibitive for large problems of the type encountered in traffic assignment analyses.

When minimizing constrained functions, the choice of descent direction and move size cannot be governed by considering only the objective function, since feasibility has to be maintained. The descent vector has to point in a direction that includes at least some feasible points. Similarly, the move size has to be constrained so that the next point is within the feasible region.

The last section of this chapter describes the convex combinations algorithm. This algorithm is a feasible descent method that is based on a linear approximation of the objective function at every iteration. Consequently, a linear program has to be solved at every iteration in order to determine the descent direction. The move size is determined by minimizing the objective function along the descent direction. This algorithm is efficient when the linear program can be solved easily.

Convergence of any of the algorithms described in this chapter can be measured by the similarity of successive solutions or by the rate of reduction in the objective function from one iteration to the next.

4.5 ANNOTATED REFERENCES

Much of the material covered in this chapter can be found in the references mentioned at the end of Chapter 2. In particular, the reader who is interested in convergence proofs should consult the texts of Zangwill (1969) and Luenberger (1973). The former includes many original contributions, and the latter summarizes the global convergence conditions for the various algorithms mentioned in this text. Good descriptions of feasible direction methods can be found in Simmons (1975) (including several methods for expediting convergence of steepest descent algorithms by modifying the search direction every few steps) and Wagner (1975). More detailed descriptions of the Fibonacci method can be found in the texts of Wilde (1964) and Wismer and Chattergy (1978). The convex combinations method first appeared in the literature in the paper by Frank and Wolfe (1956). A general proof of convergence for this method is offered by Zangwill in the aforementioned text.

4.6 PROBLEMS

4.1. Derive formula [4.1] for interval reduction methods.

4.2. Show that if an interval reduction method with a constant reduction ratio has to satisfy Eq. [4.2], then the reduction ratio is the golden section. Show also that $r = \frac{1}{2}(\sqrt{5} - 1)$ leads to Eq. [4.2].

4.3. Use the golden section method to find the minimum of the function $z(x) = 3x^2 - 4x + 1$ within $x^* \pm 0.1$ in the interval [0, 2]. How many function evaluations are needed? How many function evaluations would be needed to find the solution within $x^* \pm 0.001$?

4.4. Use the golden section to find, within ± 0.15 of the optimum, the value of x that minimizes the function

$$z(x) = \max\left\{\left(2 - \frac{x}{3}\right), (x-3)^2, \frac{x}{5}\right\}$$

such that $0 \le x \le 8$.

4.5. Using a mathematical programming text, describe the Fibonacci search algorithm. Compare its mechanism and its efficiency to that of the golden section method and the bisection method.

4.6. Use the bisection method to find the minimum of $z(x) = 3x^2 - 4x + 1$ over the interval [0, 2]. Determine the optimal value of x within 5% of the initial interval. How many function evaluations are needed to get within $x^* \pm 0.001$?

4.7. Find the minimum of $z(x) = 2x^3 + 3x^2 - 12x + 5$ in the interval [0, 4] using:

(a) Newton's method (use $x^0 = 0$).

(b) The false position method (use $x^0 = 0$, $x^1 = 0.5$).

(c) The quadratic fit method (use $x_1 = 0$, $x_2 = 0.5$, $x_3 = 1$).

***4.8.** **(a)** Show why the curve-fitting algorithms described in the text are not truly descent methods. How can the quadratic fit method be modified to be a descent method.

(b) Show why Newton's method for minimizing multivariate functions (Eq. [4.12]) is not a descent method and how should it be modified to be a descent method.

4.9. Develop the algorithmic iteration for the quadratic fit method (i.e., derive formula [4.6]).

4.10. Write a computer subroutine that accepts a function, an interval, and a convergence criterion and performs a golden section search.

4.11. Write a computer subroutine that executes a line search using the bisection method.

4.12. Suggest a procedure for amending an interval reduction method so that it can be used for the line search phase when minimizing an unconstrained function.

***4.13.** Prove that the steepest descent algorithm is a descent method.

4.14. Using the steepest descent method, find the minimum of the quadratic function

$$z(x_1, x_2) = (x_1 - 2)^2 + 4(x_2 - 3)^2$$

Starting from $\mathbf{x}^0 = (0, 0)$ plot the algorithmic moves and verify the zigzag property of the algorithm.

4.15. Perform four iterations of the steepest descent method to solve

$$\min z(\mathbf{x}) = x_1^2 + 2x_2^2 + x_3^2 - 2x_1 - 3x_2 - 2x_3$$

Start at the point $x = (0, 0, 0)$.

***4.16.** Find an approximate analytical expression for the optimal step size along the negative gradient direction for convex functions. [*Hint:* Use a second-order expansion of $z(\mathbf{x})$ about $z(\mathbf{x}^n)$.]

4.17. Consider the mechanics of feasible descent algorithms.

(a) Why do only the nonbinding constraints need be considered in maintaining feasibility along the (feasible) descent direction?

(b) Why do only the binding constraints need be considered in finding the best feasible descent direction?

4.18. Show that if $z(\mathbf{x})$ is a convex function of $\mathbf{x}$, $z(\mathbf{x}^n + \alpha\mathbf{d}^n)$ is a convex function of α.

4.19. Show that the direction used by the convex combinations method is always a descent direction.

4.20. Use the convex combinations algorithm to solve the program:

$$\min f(x_1, x_2) = 4(x_1 - 10)^2 + (x_2 - 4)^2$$

subject to

$$x_1 - x_2 \leq 10$$

$$\tfrac{1}{5}x_1 - x_2 \geq 3$$

$$x_1 \geq 0$$

Use a plot of the feasible region to solve the linearized problem by inspection. Plot the algorithmic steps in this space.

4.21. The convex combinations algorithm generates a lower bound to the objective function at every iteration. Use this bound to develop a convergence criterion for this algorithm.

5

Solving for User Equilibrium

The problem of finding the user equilibrium over a transportation network was introduced in Chapter 1. It involves the assignment of O–D flows to the network links so that travel time on all used paths for any O–D pair equals the minimum travel time between the origin and destination. This problem was formulated as a mathematical program (the equivalent UE minimization) in Chapter 3. Chapter 4 reviewed a general class of algorithms for solving mathematical programs and focused on the convex combinations method. This chapter describes the application of this method to the solution of the user-equilibrium minimization program.

Before this application of the algorithm is explained, however, two of the most common heuristic methods for finding the user-equilibrium flow pattern are outlined in Section 5.1. These approaches had been extensively used before solution algorithms for the UE program were developed and are still widely applied today. Section 5.2 then demonstrates how the convex combinations algorithm can be applied to the equivalent user-equilibrium program. The last section of this chapter concentrates on an algorithm for finding the minimum-travel-time path between two network nodes. The determination of minimum paths is a major component of the algorithmic solution of the UE program (and of the heuristic algorithms mentioned in Section 5.1).

5.1 HEURISTIC EQUILIBRATION TECHNIQUES

The heuristic approaches to the user-equilibrium problem reviewed in this section include *capacity restraint* methods and *incremental assignment* techniques. At the core of these methods lies the *network loading* mechanism.

Network loading is the process of assigning the O–D entries to the network for specific (constant) link travel times. The process follows the route-choice criterion, which underlies any traffic assignment model. As argued in Chapter 1, the UE flow pattern is the result of each motorist using the minimum travel time path between his origin and his destination. Accordingly, the network loading mechanism used in all the algorithms designed to solve the UE problem assigns each O–D flow to the shortest travel time path connecting this O–D pair. As mentioned in Chapter 3 (see discussion following Eqs. [3.28]), this procedure is known as the "all-or-nothing" assignment.

In the all-or-nothing procedure, each O–D pair r–s is examined in turn and the O–D flow, q_{rs}, is assigned to every link that is on the minimum-travel-time path connecting origin r to destination s. All other paths connecting this O–D pair are not assigned any flow. During this process, the link travel times are assumed to be fixed (i.e., not flow dependent) at some value. Accordingly, if there is a performance curve, $t_a(x_a)$, corresponding to each link in the network, any mention of all-or-nothing assignment (or any other network loading mechanism) should be made in conjunction with a specific travel time. For example, the all-or-nothing assignment can be applied to an empty network, that is, using the travel times $t_a = t_a(0)$ for every link, a. The only computational difficulty with the all-or-nothing procedure involves the identification of the minimum-travel-time paths connecting each O–D pair. This, however, is a well-researched problem in graph theory and several algorithms for its solution are readily available. Section 5.3 describes one of the more efficient of these procedures, one that is particularly applicable to transportation networks.

Many of the early urban transportation studies have used the all-or-nothing procedure (based on empty network times) as the traffic assignment procedure. This assignment method does not recognize, of course, the dependence between flows and travel time, thus, in effect, it ignores the equilibrium problem altogether.

Capacity Restraint

In an attempt to capture the equilibrium nature of the traffic assignment problem, transportation planners have devised an iterative scheme known as capacity restraint. This method involves a repetitive all-or-nothing assignment in which the travel times resulting from the previous assignment are used in the current iteration. The algorithm can be summarized as follows:

Step 0: *Initialization.* Perform all-or-nothing assignment based on $t_a^0 = t_a(0), \forall\, a$. Obtain a set of link flows $\{x_a^0\}$. Set iteration counter $n := 1$.

Step 1: *Update.* Set $t_a^n = t_a(x_a^{n-1}), \forall\, a$.

Step 2: *Network loading.* Assign all trips to the network using all-or-nothing based on travel times $\{t_a^n\}$. This yields a set of link flows $\{x_a^n\}$.

TABLE 5.1 Capacity Restraint Algorithm Applied to the Network in Figure 5.1

Iteration Number	Algorithmic Step	Link		
		1	2	3
0	Initialization	$t_1^0 = 10$	$t_2^0 = 20$	$t_3^0 = 25$
		$x_1^0 = 10$	$x_2^0 = 0$	$x_3^0 = 0$
1	Update	$t_1^1 = 947$	$t_2^1 = 20$	$t_3^1 = 25$
	Loading	$x_1^1 = 0$	$x_2^1 = 10$	$x_3^1 = 0$
2	Update	$t_1^2 = 10$	$t_2^2 = 137$	$t_3^2 = 25$
	Loading	$x_1^2 = 10$	$x_2^2 = 0$	$x_3^2 = 0$
3	Update	$t_1^3 = 947$	$t_2^3 = 20$	$t_3^3 = 25$
	Loading	$x_1^3 = 0$	$x_2^3 = 10$	$x_3^3 = 0$
		⋮	⋮	⋮

Step 3: *Convergence test.* If $\max_a \{|x_a^n - x_a^{n-1}|\} \le \kappa$,[†] stop (the current set of link flows is the solution). Otherwise, set $n := n + 1$ and go to step 1.

Table 5.1 demonstrates an application of this procedure to the example network depicted in Figure 5.1, shown on page 114. Starting with travel times corresponding to an empty network, the initial solution is determined and the iterations follow the above-mentioned algorithm. Note that the algorithm does not converge, as the flow "flip-flops" between links 1 and 2, whereas link 3 does not get loaded at all.

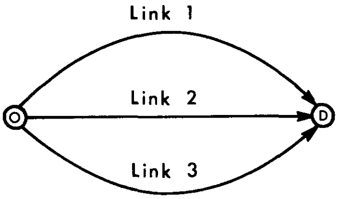
To remedy this situation, the algorithm can be modified as follows. First, instead of using the travel time obtained in the previous iteration for the new loading, a combination of the last two travel times obtained is used. This introduces a "smoothing" effect. Second, the failure to converge is recognized explicitly and the algorithm is terminated after a given number of iterations, N. The equilibrium flow pattern is then taken to be the average flow for each link over the last four iterations (obviously, N should never be less than 4). This form of the algorithm was adopted by the U.S. Federal Highway Administration (FHWA) as part of its transportation planning package. The steps of the modified capacity restraint algorithm (using weights of 0.75 and 0.25 for the averaging process) are as follows:

Step 0: *Initialization.* Perform an all-or-nothing assignment based on $t_a^0 = t_a(0)$. Obtain $\{x_a^0\}$. Set $n := 1$.

Step 1: *Update.* Set $\tau_a^n = t_a(x_a^{n-1})$, $\forall\, a$.

Step 2: *Smoothing.* Set $t_a^n = 0.75 t_a^{n-1} + 0.25 \tau_a^n$, $\forall\, a$.

[†] This convergence test is based on the maximum change in link flow between successive iterations. Other criteria can also be used.



$$t_1 = 10[1+0.15[\frac{x_1}{2}]^4] \text{ time units}$$

$$t_2 = 20[1+0.15[\frac{x_2}{4}]^4] \text{ time units}$$

$$t_3 = 25[1+0.15[\frac{x_3}{3}]^4] \text{ time units}$$

$$x_1 + x_2 + x_3 = 10 \text{ flow units}$$

Figure 5.1 Network example, with three links and one O–D pair.

Step 3: *Network loading.* Perform all-or-nothing assignment based on travel times $\{t_a^n\}$. This yields $\{x_a^n\}$.

Step 4: *Stopping rule.* If $n = N$, go to step 5. Otherwise, set $n := n + 1$ and go to step 1.

Step 5: *Averaging.* Set $x_a^* = \frac{1}{4} \sum_{l=0}^{3} x_a^{n-l}\ \forall\ a$ and stop. ($\{x_a^*\}$ are the link flows at equilibrium.)

The smoothing is accomplished by creating a temporary link-travel-time variable, τ_a^n, which is not used as the travel time for the next iteration (see step 1). Instead, it is averaged together with the travel time used in the last iteration, t_a^{n-1}, to obtain the link travel time for the current iteration, t_a^n. This is done in step 2. This algorithm differs from the original capacity restraint algorithm in the addition of the smoothing step and the averaging step. Note that step 4 is called a stopping rule rather than a convergence test, since there is no reason to expect this algorithm to converge to the equilibrium solution, in spite of these changes.

An application of this algorithm to the network example of Figure 5.1 is demonstrated in Table 5.2. Note that it produces a solution that is not an equilibrium flow pattern since, even though all paths are used, t_1^* is substantially different from t_2^* and t_3^*. The table shows three iterations in addition to the initialization, meaning that four minimum-path computations are represented. For large networks, this is the main computational burden, and thus traffic assignment algorithms should be compared in terms of efficiency for a given number of minimum-path computations (i.e., a given number of all-or-nothing assignments).

TABLE 5.2 Modified Restraint Algorithm Applied to the Network in Figure 5.1

Iteration Number	Algorithmic Step	Link		
		1	2	3
0	Initialization	$t_1^0 = 10$	$t_2^0 = 20$	$t_3^0 = 25$
		$x_1^0 = 10$	$x_2^0 = 0$	$x_3^0 = 0$
1	Update	$\tau_1^1 = 947$	$\tau_2^1 = 20$	$\tau_3^1 = 25$
	Smoothing	$t_1^1 = 244$	$t_2^1 = 20$	$t_3^1 = 25$
	Loading	$x_1^1 = 0$	$x_1^1 = 10$	$x_3^1 = 0$
2	Update	$\tau_1^2 = 10$	$\tau_2^2 = 137$	$\tau_3^2 = 25$
	Smoothing	$t_2^2 = 186$	$t_2^2 = 49$	$t_3^2 = 25$
	Loading	$x_1^2 = 0$	$x_2^2 = 0$	$x_3^2 = 10$
3	Update	$\tau_1^3 = 10$	$\tau_2^3 = 20$	$\tau_3^3 = 488$
	Smoothing	$t_1^3 = 142$	$t_2^3 = 42$	$t_3^3 = 141$
	Loading	$x_1^3 = 0$	$x_2^3 = 10$	$x_3^3 = 0$
Average		$x_1^* = 2.5$	$x_2^* = 5.0$	$x_3^* = 2.5$
		$t_1^* = 13.7$	$t_2^* = 27.3$	$t_3^* = 26.8$

Incremental Assignment

Another heuristic method for attaining the user-equilibrium solution assigns a portion of the origin–destination matrix at each iteration. The travel times are then updated and an additional portion of the O–D matrix is loaded onto the network. In this manner, the general shape of the link performance functions can be "traced" with the successive assignments. This procedure, which is known as incremental assignment, is outlined below. (In this description, w_a^n denotes the flow on link a resulting from the assignment of the nth increment of the O–D matrix onto the network.)

Step 0: *Preliminaries.* Divide each origin–destination entry into N equal portions (i.e. set $q_{rs}^n = q_{rs}/N$). Set $n := 1$ and $x_a^0 = 0$, $\forall$ a.

Step 1: *Update.* Set $t_a^n = t_a(x_a^{n-1})$, $\forall$ a.

Step 2: *Incremental loading.* Perform all-or-nothing assignment based on $\{t_a^n\}$, but using only the trip rates q_{rs}^n for each O–D pair. This yields a flow pattern $\{w_a^n\}$.

Step 3: *Flow summation.* Set $x_a^n = x_a^{n-1} + w_a^n$, $\forall$ a.

Step 4: *Stopping rule.* If $n = N$, stop (the current set of link flows is the solution); otherwise, set $n := n + 1$ and go to step 1.

In some versions of this algorithm, the incremental all-or-nothing procedure in step 2 is modified and origin–destination pairs are selected in random order,

TABLE 5.3 Incremental Assignment Algorithm Applied to the Network in Figure 5.1

Iteration (Increment)	Algorithmic Step	Link 1	Link 2	Link 3
1	Update	$t_1^1 = 10$	$t_2^1 = 20$	$t_3^1 = 25$
	Incremental loading	$w_1^1 = 2.5$	$w_2^1 = 0$	$w_3^1 = 0$
	Summation	$x_1^1 = 2.5$	$x_2^1 = 0$	$x_3^1 = 0$
2	Update	$t_1^2 = 14$	$t_2^2 = 20$	$t_3^2 = 25$
	Incremental loading	$w_1^2 = 2.5$	$w_2^2 = 0$	$w_3^2 = 0$
	Summation	$x_1^2 = 50$	$x_2^2 = 0$	$x_3^2 = 0$
3	Update	$t_1^3 = 69$	$t_2^3 = 20$	$t_3^3 = 25$
	Incremental loading	$w_1^3 = 0$	$w_2^3 = 2.5$	$w_3^3 = 0$
	Summation	$x_1^3 = 5.0$	$x_2^3 = 2.5$	$x_3^3 = 0$
4	Update	$t_1^4 = 69$	$t_2^4 = 20.5$	$t_3^4 = 25$
	Incremental loading	$w_1^4 = 0$	$w_2^4 = 2.5$	$w_3^4 = 0$
	Summation	$x_1^4 = 5.0$	$x_2^4 = 5.0$	$x_3^4 = 0$
Travel time at convergence		$t_1^* = 69$	$t_2^* = 27.3$	$t_3^* = 25$

with a flow summation phase (as in step 3) and travel-time update (as in step 4) following each partial assignment (i.e., after each O–D entry is loaded).

Table 5.3 demonstrates the application of this algorithm to the network example of Figure 5.1. The flow pattern resulting from an incremental assignment with four increments is $\mathbf{x} = (5, 5, 0)$. The last row in the table $[t_a^* = t_a(x_a^4)$ for $a = 1, 2, 3]$ is provided so that the reader can judge the closeness of the solution to an equilibrium flow pattern. As evident from this table the two used paths (links 1 and 2) do not exhibit equal travel times. Furthermore, these travel times are higher than that of the unused path (link 3).

In conclusion, it is clear that the heuristic methods reviewed in this section either do not converge or produce a set of flows that is not in agreement with the user-equilibrium criterion. It may be reasonable to believe, though, that in general, as the number of increments grows, the incremental assignment algorithm may generate a flow pattern closer to the user-equilibrium condition. A very large number of increments (associated with a considerable computational effort) may be required, however, and even then the method will not always produce the user-equilibrium flow pattern.

5.2 APPLYING THE CONVEX COMBINATIONS METHOD

As the examples of the preceding section demonstrate, the heuristic methods discussed thus far may not converge to the equilibrium solution. It is this failure to converge which motivates the overall approach taken in this book—

formulating the user-equilibrium problem as a mathematical program and solving this program. Chapter 3 dealt with the program formulation and showed that the flow pattern which minimizes the UE equivalent program is, in fact, a user-equilibrium solution. This program includes a convex (nonlinear) objective function and a linear constraint set. Chapter 4 reviewed various feasible direction methods for solving such programs, emphasizing the convex combinations method. This method is especially suitable for solving the equivalent UE program since the direction-finding step can be executed relatively efficiently. This step involves the solution of a linear program, which, in the case of the UE program, has a special structure that simplifies its solution.

Using the notation introduced in Chapter 3, the UE objective function is given by

$$\min z(\mathbf{x}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega \qquad [5.1a]$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\, r, s \qquad [5.1b]$$

$$f_k^{rs} \geq 0 \qquad \forall\, k, r, s \qquad [5.1c]$$

where x_a is the flow on link a, f_k^{rs} is the flow on path k connecting origin r with destination s, and the incidence relationships $x_a = \sum_{rs} \sum_k f_k^{rs} \delta_{a,k}^{rs}$, $\forall$ a, hold.

Applying the convex combinations algorithm to the minimization of the UE program requires, at every iteration, a solution of the linear program (LP)

$$\min z^n(\mathbf{y}) = \nabla z(\mathbf{x}^n) \cdot \mathbf{y}^T = \sum_a \frac{\partial z(\mathbf{x}^n)}{\partial x_a}\, y_a \qquad [5.2]$$

over all feasible values of $\mathbf{y} = (\ldots, y_a, \ldots)$ (see Eqs. [4.23]). The gradient of $z(\mathbf{x})$ with respect to the link flows at the nth iteration is the link travel-time vector, since $\partial z(\mathbf{x}^n)/\partial x_a = t_a^n$ (see Eq. [3.12a]). The LP objective function at the nth iteration (given by Eq. [5.2]) thus becomes

$$\min z^n(\mathbf{y}) = \sum_a t_a^n y_a \qquad [5.3a]$$

subject to

$$\sum_k g_k^{rs} = q_{rs} \qquad \forall\, r, s \qquad [5.3b]$$

$$g_k^{rs} \geq 0 \qquad \forall\, k, r, s \qquad [5.3c]$$

where $y_a = \sum_{rs} \sum_k g_k^{rs} \delta_{a,k}^{rs}$, $\forall$ a, and $t_a^n = t_a(x_a^n)$. In this linear program, y_a is the auxiliary variable representing the flow on link a, while g_k^{rs} is the auxiliary flow variable for path k connecting O–D pair r–s. This program calls for minimizing the total travel time over a network with fixed (not flow-dependent) travel times, t_a^n. The total travel time spent in the network will be minimized by assigning all motorists to the shortest travel-time path connecting their origin to their destination (see Section 3.5). Such an assignment is performed by the all-or-nothing network loading procedure mentioned earlier. Consequently, the program in Eqs. [5.3] is known as the all-or-nothing program. The core of the all-or-nothing procedure is the determination of the shortest paths between all origins and all destinations. Section 5.3 outlines an efficient method for finding these paths between all the network nodes. At this point, it is sufficient to mention that program [5.3] can be solved efficiently.

To see that the solution of program [5.3] does not involve more than an all-or-nothing assignment, the linearization step of the convex combinations method can be derived by taking the gradient of objective function [5.1a] with respect to path flows (instead of link flow, as in [5.2]). The linearized program then becomes

$$\min z^n(\mathbf{g}) = \nabla_f z[\mathbf{x}(\mathbf{f}^n)] \cdot \mathbf{g}^T = \sum_{rs} \sum_k c_k^{rs^n} g_k^{rs} \qquad [5.4a]$$

subject to

$$\sum_{rs} \sum_k g_k^{rs} = q_{rs} \qquad \forall\ r, s \qquad [5.4b]$$

$$g_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [5.4c]$$

where $c_k^{rs^n}$ is the travel time on path k connecting r and s, at the nth iteration of the algorithm. This program can be decomposed by O–D pair since the path travel times are fixed. The resulting subproblem for pair r–s is given by

$$\min z^n(\mathbf{g}^{rs}) = \sum_k c_k^{rs} g_k^{rs} \qquad [5.5a]$$

subject to

$$\sum_k g_k^{rs} = q_{rs} \qquad [5.5b]$$

$$g_k^{rs} \geq 0 \qquad \forall\ k \qquad [5.5c]$$

This program is obviously minimized by finding the path, m, with the smallest travel time among all paths connecting r and s, and assigning all the flow to it. In other words,

$$g_m^{rs} = q_{rs} \qquad \text{if } c_m^{rs} \leq c_k^{rs} \quad \forall\ k \qquad [5.6a]$$

and

$$g_k^{rs} = 0 \qquad \text{for all other paths} \qquad [5.6b]$$

In case two or more paths are tied for the minimum, any one of them can be chosen for flow assignment.

Once the path flows $\{g_k^{rs^n}\}$ are found, the link flows can be calculated by using the incidence relationships, that is,

$$y_a^n = \sum_{rs} \sum_k g_k^{rs^n} \delta_{a,k}^{rs} \qquad \forall\ a$$

This solution defines the descent direction $\mathbf{d}^n = \mathbf{y}^n - \mathbf{x}^n$.

The remaining algorithmic steps of the convex combinations method are similar to those outlined in Section 4.3. The initial solution can be usually determined by applying an all-or-nothing network loading procedure to an empty network, in a way similar to the initialization of the capacity restraint methods discussed in Section 5.1.

The line search for the optimal move size can be performed with any of the interval reduction methods, but the bisection method (Bolzano search) may be particularly applicable. The reason is that the derivative of the objective function $z[\mathbf{x}^n + \alpha(\mathbf{y}^n - \mathbf{x}^n)]$ with respect to α is given by

$$\frac{\partial}{\partial \alpha} z[\mathbf{x}^n + \alpha(\mathbf{y}^n - \mathbf{x}^n)] = \sum_a (y_a^n - x_a^n) t_a[x_a^n + \alpha(y_a^n - x_a^n)] \qquad [5.7]$$

which can be easily calculated for any value of α.

The stopping criterion for solving the UE program could be based on the values of the objective function. As mentioned in Section 3.1, however, this function is merely a mathematical construct that lacks behavioral or economic meaning. Consequently, the convergence criterion should be based on the relevant figures of merit, which in this case consist of the flows and the travel times. A possible measure of the closeness of a particular solution to equilibrium is the similarity of successive O–D travel times. Letting u_{rs}^n denote the minimum path travel time between O–D pair r–s at the nth iteration, the algorithm can terminate if, for example,

$$\sum_{rs} \frac{|u_{rs}^n - u_{rs}^{n-1}|}{u_{rs}^n} \leq \kappa \qquad [5.8a]$$

Alternatively, a criterion that is based on the change in flows can be used. For example, the algorithm can terminate if

$$\frac{\sqrt{\sum_a (x_a^{n+1} - x_a^n)^2}}{\sum_a x_a^n} \leq \kappa' \qquad [5.8b]$$

Other convergence criteria can be used as well. The algorithm itself, when applied to the solution of the UE problem, can be summarized as follows:

Step 0: *Initialization.* Perform all-or-nothing assignment based on $t_a = t_a(0)$, $\forall$ a. This yields $\{x_a^1\}$. Set counter $n := 1$.

Step 1: *Update.* Set $t_a^n = t_a(x_a^n)$, $\forall$ a.

Step 2: *Direction finding.* Perform all-or-nothing assignment based on $\{t_a^n\}$. This yields a set of (auxiliary) flows $\{y_a^n\}$.

Step 3: *Line search.* Find α_n that solves

$$\min_{0 \leqslant \alpha \leqslant 1} \sum_a \int_0^{x_a^n + \alpha(y_a^n - x_a^n)} t_a(\omega)\, d\omega$$

Step 4: *Move.* Set $x_a^{n+1} = x_a^n + \alpha_n(y_a^n - x_a^n)$, $\forall\ a$.

Step 5: *Convergence test.* If a convergence criterion is met, stop (the current solution, $\{x_a^{n+1}\}$, is the set of equilibrium link flows); otherwise, set $n := n + 1$ and go to step 1.

Note the similarity between this algorithm and the capacity restraint method reviewed in Section 5.1. In fact, if the move size, α_n, is fixed at $\alpha_n = 1$ for all n, the resulting algorithm is identical to the capacity restraint method. This observation is important because it means that existing computer programs that use the capacity restraint method can be easily modified to give a convergent algorithmic solution to the UE problem (by inserting step 3).

To demonstrate the convergence of this algorithm to the equilibrium solution, it is applied to the three-link network example depicted in Figure 5.1. The iterations of this algorithm are shown in Table 5.4. The results in this table should be compared to the convergence patterns of the heuristic methods shown in Tables 5.1 to 5.3. From Table 5.4 it is evident that after five iterations (six minimum-path calculations, including the initialization process) the flows are close to equilibrium; the travel times on all three routes are very similar.

A comparison of this algorithm with the previous (heuristic) methods, in

TABLE 5.4 Convex Combinations Algorithm Applied to the Network in Figure 5.1

Iteration Number	Algorithmic Step	Link			Objective Function	Step Size
		1	2	3		
0	Initialization	$t_1^0 = 10.0$	$t_2^0 = 20.0$	$t_3^0 = 25.0$		
		$x_1^1 = 10.00$	$x_2^1 = 0.00$	$x_3^1 = 0.00$		
1	Update	$t_1^1 = 947.0$	$t_2^1 = 20.0$	$t_3^1 = 25.0$	$z(\mathbf{x}) = 1975.00$	
	Direction	$y_1^1 = 0$	$y_2^1 = 10$	$y_3^1 = 0$		$\alpha_1 = 0.596$
	Move	$x_1^2 = 4.04$	$x_2^2 = 5.96$	$x_3^2 = 0.00$		
2	Update	$t_1^2 = 35.0$	$t_2^2 = 35.0$	$t_3^2 = 25.0$	$z(\mathbf{x}) = 197.00$	
	Direction	$y_1^2 = 10$	$y_2^2 = 0$	$y_3^2 = 0$		$\alpha_1 = 0.161$
	Move	$x_1^3 = 3.39$	$x_2^3 = 5.00$	$x_3^3 = 1.61$		
3	Update	$t_1^3 = 22.3$	$t_2^3 = 27.3$	$t_3^3 = 35.3$	$z(\mathbf{x}) = 189.98$	
	Direction	$y_1^3 = 10$	$y_2^3 = 0$	$y_3^3 = 0$		$\alpha = 0.035$
	Move	$x_1^4 = 3.62$	$x_2^4 = 4.83$	$x_3^4 = 1.55$		
4	Update	$t_1^4 = 26.1$	$t_2^4 = 26.3$	$t_3^4 = 25.3$	$z(\mathbf{x}) = 189.44$	
	Direction	$y_1^4 = 0$	$y_2^4 = 0$	$y_3^4 = 10$		$\alpha = 0.020$
	Move	$x_1^5 = 3.54$	$x_2^5 = 4.73$	$x_3^5 = 1.72$		
5	Update	$t_1^5 = 24.8$	$t_2^5 = 25.8$	$t_3^5 = 25.4$	$z(\mathbf{x}) = 189.33$	
	Direction	$y_1^5 = 10$	$y_2^5 = 0$	$y_3^5 = 0$		$\alpha = 0.007$
	Move	$x_1^6 = 3.59$	$x_2^6 = 4.70$	$x_3^6 = 1.71$		
	Update	$t_1^6 = 25.6$	$t_2^6 = 25.7$	$t_3^6 = 25.4$	$z(\mathbf{x}) = 189.33$	

terms of computational efficiency, can be made on the basis of the flow pattern at the end of the third iteration (this is equivalent to four all-or-nothing assignments in terms of computational effort). At this point, the travel times generated by the convex combinations algorithm on the three paths are $\mathbf{t}^4 = (26.1, 26.3, 25.3)$. This situation is much closer to equilibrium than the path travel times generated by the modified capacity restraint method, $\mathbf{t}^* = (13.7, 27.3, 26.8)$, or the ones produced by the incremental assignment method, $\mathbf{t}^* = (69.0, 27.3, 25.0)$.

As Table 5.4 shows, flow is taken away from congested paths and assigned to less congested paths, at each iteration of the convex combinations method. This process equalizes the travel times among all paths and brings the system toward equilibrium. The reduction in the objective function value illustrated in Table 5.4 follows the pattern demonstrated in Figure 4.9. Again, the marginal contribution of each successive iteration to the reduction in the objective function value is decreasing.

In solving the UE program over a large network, each iteration involves a significant computational cost, due primarily to the effort required to calculate the shortest paths in the direction-finding step. It is important, then, that a good answer is achieved after a relatively small number of iterations.

In practice, this is not a major problem because of two reasons. First, the convergence pattern of the convex combinations algorithm is such that the first few iterations are the most "cost effective." In other words, the flow pattern after only a few iterations is not very far from equilibrium. Second, the convergence criteria used in practice are not very stringent and thus convergence can be achieved after only a small number of iterations. This is because the accuracy of the input data does not warrant the effort needed to obtain an extremely accurate equilibrium flow pattern.

The number of iterations required for convergence is significantly affected by the congestion level on the network. In relatively uncongested networks, a single iteration may suffice since the link flows may be in the range where the performance functions are almost flat. This means that the updated travel times are very close to the initial ones, generating a set of link flows that is quite similar to the initial solution. As congestion builds up, more iterations are required to equilibrate the network. This effect is demonstrated in Figure 5.2, which depicts the convergence pattern of the convex combinations method for a medium-sized network, for three congestion levels. The convergence measure depicted in this figure is based on the objective function values. The three curves in the figure correspond to low, medium, and high levels of congestion over the network. As the figure shows, the congestion effect on the convergence rate can be quite pronounced.

In actual applications, only four to six iterations are usually sufficient to find the equilibrium flow pattern over large urban networks. This number reflects common practice in terms of trade-offs among analytical accuracy, data limitations, and budget, given typical congestion levels.

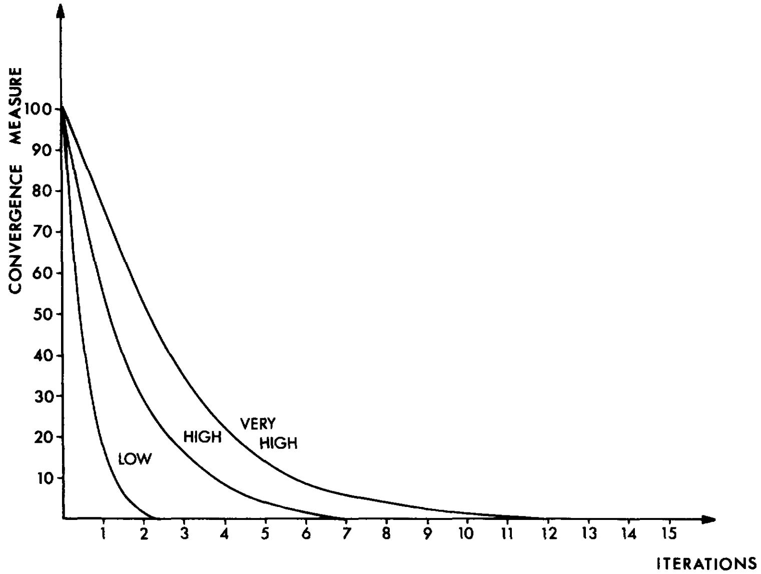


Figure 5.2 Convergence rate of the convex combinations algorithm for various congestion levels. Congestion is measured in terms of $\sum_a x_a t_a(x_a)/\sum_a x_a t_a(0)$, where x_a and t_a are the equilibrium flow and travel times. Convergence is measured by the value of the objective function, normalized between the initial iteration and the 30th one. At iteration n this measure is $\{1 - [z(\mathbf{x}^0) - z(\mathbf{x}^n)]/[z(\mathbf{x}^0) - z(\mathbf{x}^{30})]\}100$.

5.3 SHORTEST PATHS OVER A NETWORK

As demonstrated in this chapter, both the optimization and heuristic approaches to the solution of the UE problem require an iterative all-or-nothing assignment. This assignment includes loading the trips between each origin–destination pair on to the shortest travel time path connecting this pair. The problem is thus that of finding the minimum-travel-time paths connecting each O–D pair for a given set of link travel times. The solution algorithm has to be very efficient since, in the course of finding the user equilibrium, the minimum-path problem has to be solved over and over again for each O–D pair at different flow levels.

The notation used up to this point will be slightly modified for the purposes of the discussion in this section. Computer memory, when storing a network, does not keep separate lists of nodes and links, but rather a list of nodes only. Links are identified by their end nodes. If i and j are nodes (i.e., i, $j \in \mathcal{N}$), the link pointing from i to j is indexed by ij (i.e., $ij \in \mathcal{A}$). Accordingly, x_{ij} and t_{ij} denote the flow and travel time, respectively, on link ij.

Shortest-Path Algorithm

The algorithm presented here is known in the operations research literature as the label-correcting method. It finds the shortest path from a given origin (root) node to all other nodes in the network. Accordingly, it has to be used for each origin in turn for a complete all-or-nothing assignment to be performed. The explanation that follows pertains to the calculation of such a minimum-path *tree*, that is, the shortest path from one root node to all other nodes.

The algorithm essentially scans the network nodes in an iterative manner. At each iteration the algorithm tries to find a path from the root to the node being scanned that is better (shorter) than the current path. The algorithm terminates when no better path can be found from the root to any of the other nodes in the network.

Assume that the network is stored in the computer as a list of links identified by their end nodes. A travel time (or length) t_{ij} is associated with each link ij. In addition, two pieces of information are stored for every node i in the network: 1) the (current) *label* of this node, l_i, and 2) its (current) *predecessor* node, p_i. The label of node i is the distance from the root node to node i along the (current) shortest path. The predecessor of node i is the node just preceding node i along the (current) shortest path. A list of the predecessor nodes (the p_i's) is continuously updated so that the minimum paths can be traced once the algorithm is terminated. (These paths are traced backward from every node to the root.) The algorithm requires an examiı.ation of all the network nodes at least once. To help manage and keep track of the nodes, the algorithm uses an additional list called the sequence list. This list includes all the nodes that have yet to be examined as well as the nodes requiring further examination.

The algorithm is initialized by setting all labels (which are arranged in a label list) to infinity (or, in practice, to a very large number), setting all predecessor nodes (in the predecessor list) to zero, and placing the origin node, r, on the sequence list with label $l_r := 0$.

Each iteration starts with the selection of a node (say, i) from the sequence list for examination. (At the first iteration, the root node is the only one on the sequence list and therefore is examined first.) All nodes, j, that can be reached from i by traversing only a single link are tested in the examination process. If the minimum path to j through i is shorter than the previous path to j, then l_j is updated. In other words, if

$$l_i + t_{ij} < l_j \qquad [5.9]$$

then the current shortest path from the root node to j can be improved by going through node i. To reflect this change, the label list is updated by setting $l_j := l_i + t_{ij}$, the predecessor list is updated by setting $p_j := i$, and the sequence list is updated by adding j to it (since this change may affect nodes that can be reached from j). Once all the nodes j (that can be reached from i) are tested, the examination of node i is complete and it is deleted from the sequence list. The algorithm terminates when the sequence list is empty. At this point, the shortest path from the root to any other node can be found by tracing the predecessor list back to the root node.

Note that the algorithm "fans out" from the root node. It first places all the nodes that can be reached from the origin by traversing only a single link on the sequence list, and eliminates the origin itself from the list. The algorithm then keeps adding nodes to the list if the label test is met and deleting nodes already examined. Note that a node eliminated from the sequence list can reappear on it in a later stage.

Example

Consider the simple network depicted in Figure 5.3. This network includes one O–D pair and four links. (The number shown next to each link indicates the travel time associated with it.) The label-correcting algorithm is used below to find the shortest path from node 1 to node 4.

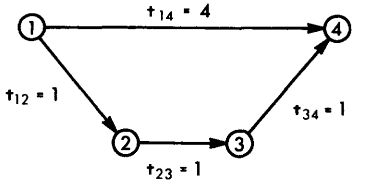


Figure 5.3 Network example for the minimum-path algorithm.

First, all labels are initialized to ∞ and all predecessors to zero. Next, the label of node 1 is changed to zero and it is placed on the sequence list. The sequence list is then scanned and node 1 is chosen for examination (at this point there are no other choices). Two nodes can be reached from node 1 (2 or 4); assume that 4 is considered first. Now, since $l_1 + t_{14} = 4 < l_4 = \infty$, the label of node 4 is changed to 4 and this node is placed on the sequence list. Node 1 is not yet erased from the list, since link 1 → 2 must be considered next. This is then followed by considering links 2 → 3 and 3 → 4. The contents of the label list, the predecessor list, and the sequence list for these iterations are given in Table 5.5. At the fourth iteration, the label of node 4 is changed from 4 to 3. The fifth iteration only verifies that no link emanates from node 4. This node is then removed from the sequence list and the algorithm terminates (since the sequence list is empty). The minimum path can now be traced backward from node 4 by using the predecessor list.

TABLE 5.5 Contents of the Label, Prediction, and Sequence Lists

		Label List				Predecessor List				
Iteration	Link Tested	Node 1	Node 2	Node 3	Node 4	Node 1	Node 2	Node 3	Node 4	Sequence List
Initialization	—	0	∞	∞	∞	0	0	0	0	1
1	1 → 4	0	∞	∞	4	0	0	0	1	1, 4
2	1 → 2	0	1	∞	4	0	1	0	1	2, 4
3	2 → 3	0	1	2	4	0	1	2	1	3, 4
4	3 → 4	0	1	2	3	0	1	2	3	4
5	—	0	1	2	3	0	1	2	3	

Note on Computer Implementation†

The label-correcting algorithm discussed here can efficiently determine the minimum-travel-time path from an origin node to all other nodes in a network. The algorithm can be applied to very large networks, but care must be taken in coding the algorithm. This section discusses some computer implementation issues and gives some guidelines for efficient coding of the label-correcting method.

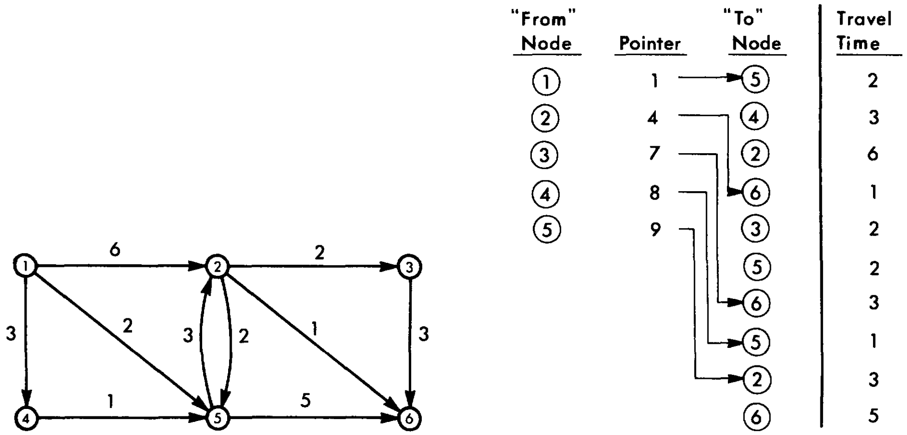
The first issue of concern is the method of storing the network. In describing the algorithm, it was assumed that a list of the network links is available. Consider, for example, the network depicted in Figure 5.4 on page 126. This network can be described by the list of its links identified by their end nodes. For example:

"From" Node	"To" Node
1	2
2	3
3	6
1	4
4	5
5	6
1	5
5	2
2	6
2	5

In addition, each link is associated with some travel time, as shown in the figure. At each iteration of the algorithm, all links emanating from a particular node have to be tested. Thus, to avoid repeated searches of the link list, it is advantageous to sort the links so that all links emanating from the same node are stored adjacently: that is,

"From" Node	"To" Node
1	2
1	4
1	5
2	3
2	6
2	5
3	6
4	5
5	2
5	6

†This section can be skipped without loss of continuity.



"From" Node	Pointer	"To" Node	Travel Time
1	1	5	2
2	4	4	3
3	7	2	6
4	8	6	1
5	9	3	2
		5	2
		6	3
		5	1
		2	3
		6	5

Figure 5.4 Network example demonstrating a forward star network presentation. (The numbers on the links show the links' travel times.)

where "from" nodes are arranged in ascending order. To save space in computer memory, this list can be stored in a "forward star" form, where each arc is represented only by its ending node. A pointer is kept for each node, indicating the position, in the link list, of links beginning with this node. The right-hand side of Figure 5.4 demonstrates this pointer representation. For the example under consideration, this pointer can be thought of as an array, **B**, of length 5 with the following entries:

$$B(1) = 1$$

$$B(2) = 4$$

$$B(3) = 7$$

$$B(4) = 8$$

$$B(5) = 9$$

Instead of keeping two link-length arrays,† the network can thus be stored with one link-length and one node-length array. Using this forward star form, all the links emanating from each node can be processed easily.

The management of the sequence list is another area in which significant computational gains can be made. First, note that *all* the nodes that represent links emanating from a particular node under examination should be tested consecutively. This keeps the sequence list small and avoids unnecessary searches for the location of a particular link.

To avoid duplicating the computation, nodes should be added to the sequence list only if they are not already on it. As it turns out, it is advanta-

†A link-length array includes as many components as the number of links in the network. Similarly, a node-length array includes as many entries as there are nodes in the network.

geous to process the sequence list (i.e., to choose candidate nodes for examination) in a particular order. In general, the sequence list is processed from the top down, and nodes are added to the list by placing them at the bottom. (In other words, the sequence is treated as a queue.) If, however, a node to be added has already been on the list (and examined and removed from it), it should be placed on the top (meaning that it would be examined next). This strategy of managing the sequence list has been shown to be most effective for computing shortest paths over transportation networks.

Figure 5.5 depicts a flowchart of the label-correcting algorithm which utilizes the forward star form and uses the aforementioned strategy for managing the sequence list. Note that, in the computer, this list is not managed by physically moving all nodes in order to place a node at the top of the list. Instead, the sequence list is a node-length array, say **s**, containing the following information regarding the status of each node in the sequence list:

$$s(i) = \begin{cases} -1 & \text{if node } i \text{ was previously on the list but} \\ & \text{is no longer on it} \\ 0 & \text{if node } i \text{ has never been on the list} \\ +j & \text{if node } i \text{ is on the list and } j \text{ is the next} \\ & \text{node on the list} \\ +\infty & \text{if node } i \text{ is on the list and it is the last} \\ & \text{node on the list} \end{cases}$$

In addition, the top and bottom of the list are identified by special pointers. Placing a node, j, at the top of the list means only that $s(j) := m$ if m was the previous first node on the list, and the top pointer is changed to point at j. Placing a node, j, at the bottom of the list is done by setting $s(j) := \infty$, $s(n) := j$ if n was the previous last node on the list, and the bottom pointer is changed to point at j.

This concludes the description of the shortest path algorithm. The remainder of this section discusses some related numerical issues regarding the solution of user-equilibrium problems.

Transportation networks are characterized by a number of links/number of nodes ratio of approximately 4. For such networks, the computational effort associated with the identification of each minimum-path tree grows linearly (on the average) with the size of the network analyzed. In other words, the computer CPU time† needed is proportional to the number of nodes (and links) in the network. This observation can be used to assess the computational effort associated with the solution of the UE program (using the algorithm outlined in Section 5.2).

Finding the minimum paths is the most computation-intensive component of each iteration of most UE solution procedures (and, in particular, the convex combinations algorithm). The other components (loading the mini-

†CPU (central processing unit) time is a measure of the effort required by the computer to execute a program.

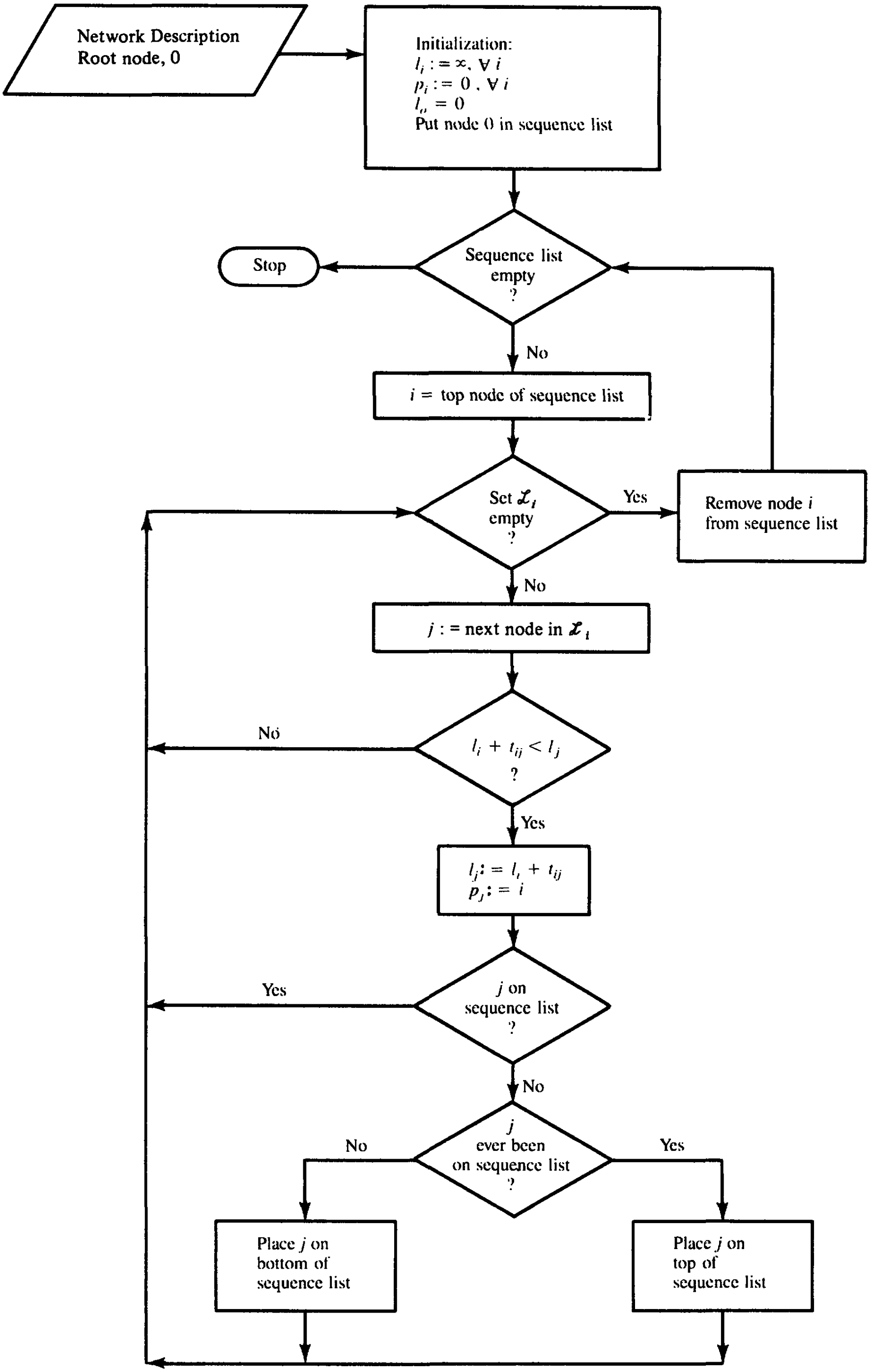

Network Description
Root node, 0
Initialization:
$l_i := \infty, \forall i$
$p_i := 0, \forall i$
$l_o = 0$
Put node 0 in sequence list
Sequence list empty ?
Stop
No
i = top node of sequence list
Set $\mathcal{L}_i$ empty ?
Yes
Remove node i from sequence list
No
j : = next node in $\mathcal{L}_i$
No
$l_i + t_{ij} < l_j$?
Yes
$l_j := l_i + t_{ij}$
$p_j := i$
Yes
j on sequence list ?
No
No
j ever been on sequence list ?
Yes
Place j on bottom of sequence list
Place j on top of sequence list

Figure 5.5 Flowchart of the label-correcting shortest-path algorithm. (The set $\mathcal{L}_i$ includes all nodes that can be reached from i by traversing a single link.)

mum paths, the line search, the updates, and the convergence checks) do not require more than a few percentages of the total CPU time. Consequently, the computational effort associated with this application of the convex combinations algorithm is proportional to the product of the number of iterations, the number of origins† (which determines the number of minimum-path trees to be calculated at each iteration) and the number of nodes (which determines the effort needed to calculate each tree). In other words,

$$\begin{pmatrix}\text{computational}\\ \text{costs}\end{pmatrix} = K\begin{pmatrix}\text{number of}\\ \text{iterations}\end{pmatrix}\begin{pmatrix}\text{number of}\\ \text{origins}\end{pmatrix}\begin{pmatrix}\text{number of}\\ \text{nodes}\end{pmatrix} \qquad [5.10]$$

where K is a constant of proportionality that is computer specific.

5.4 SUMMARY

The focus of this chapter is on finding the user-equilibrium flow pattern over a transportation network. The first section reviews the two most widely used heuristic techniques for solving this network equilibrium problem. These include capacity restraint methods, which are not guaranteed to converge, and incremental assignment methods, which may "converge" to a nonequilibrium solution. The inadequate performance of these heuristics has motivated the development of the equivalent minimization approach used in this text.

The convex combinations algorithm described in Section 4.3 can be easily applied to the solution of the UE equivalent minimization program. The solution of the linear program associated with each step of this algorithm requires merely an all-or-nothing network loading. Such a loading involves the assignment of all flow between each O–D pair to the minimum-travel-time path connecting this O–D pair. The problem of finding the set of all relevant minimum paths is a well-known problem in operations research for which many efficient algorithms are available.

One of the most efficient algorithms for finding the minimum path from a single node to all other network nodes is the label-correcting algorithm described in Section 5.3. This algorithm is extremely efficient and its execution can be expedited even further by careful coding and list-processing procedures.

The use of the convex combinations method in conjunction with the label-correcting (or any other minimum-path) algorithm for the direction-finding step provides an easy and efficient approach to the minimization of the equivalent UE program. The convergence of the convex combinations method is asymptotic in nature; the marginal contribution of each additional iteration

†If the number of destinations is smaller than the number of origins, the all-or-nothing assignment can be carried out by rooting the minimum-path trees at the destinations. Each tree would then give the minimum path from each of the network nodes to the destination root. The total computational costs associated with solving the UE program are, then, proportional to the product of the number of iterations, the number of nodes, and the minimum of the number of origins and number of destinations.

to the reduction in the value of the objective function is decreasing. The number of iterations required for convergence is primarily a function of the congestion over the network. The computational effort needed for each iteration is proportional to the number of origins (or number of destinations, if it is smaller) and the size of the network.

5.5 ANNOTATED REFERENCES

The two heuristic method presented in Section 5.1 are embedded in many of the early transportation planning computer packages. The U.S. Federal Highway Administration (FHWA) includes a modified capacity restraint method in its Urban Transportation Planning Package (Federal Highway Administration, 1977). The incremental assignment method was used in The DODOTRANS package developed at M.I.T. by Manheim and Ruiter (1970). Both types of procedures were criticized by a number of researchers on the grounds mentioned in this chapter. For example, Sheffi and Daganzo (1978) showed the divergence property of the capacity restraint method for a contrived network, while Ferland et al. (1975) demonstrated the problems associated with the use of incremental assignment. The example used in Section 5.1 to demonstrate both the problems associated with the use of heuristic methods and the convergence of the convex combinations method follows the exposition by Eash et al. (1979). Currently, the network assignment package UROAD, which is part of the UMTA Transportation Planning System (UTPS) supported by the U.S. Urban Mass Transportation Administration, includes the convex combinations algorithm.

The application of Frank and Wolfe's convex combinations method to the solution of transportation network equilibrium was first suggested by Bruynooghe et al. (1968) and applied by Murchland (1969). Shortly thereafter it was used by LeBlanc et al. (1975), who coded and tested the algorithm for a small city. At the same time, Nguyen (1974b) suggested the use of the convex simplex method for solving the UE equivalent minimization program. Nguyen (1974a) also suggested the use of the reduced gradient method and a modified reduced gradient method for this purpose. In some side-by-side comparative experiments, Florian and Nguyen (1976) found that even though the convex simplex method converges somewhat faster than the convex combinations method, it requires more computer memory. Consequently, the overall computational effort required by both methods is similar. The latter reference also includes an interesting validation study, where the equilibrium flows were found to be in satisfactory agreement with ground counts in the city of Winnipeg, Canada. Bovy and Jansen (1981) also demonstrate a high level of agreement between ground counts taken in the city of Eindhoven, the Netherlands, and the flows resulting from a user-equilibrium assignment. Figure 5.2 is taken from Mimis (1984), who used a "hub and spokes" simulated network to study some numerical issues in conjunction with equilibrium assignment algorithms.

The minimum-path algorithm presented in Section 5.3 is the label-correcting algorithm suggested by Moore (1957), with the modification for handling the sequence list suggested by Pape (1974). The presentation of the algorithm and the discussion of the computer implementation issues follow Dial et al. (1977), who tested a number of the most widely used minimum-path algorithms and concluded that the Moore–Pape algorithm is the most efficient for transportation networks. This reference includes a discussion and detailed presentations of many of these algorithms.

5.6 PROBLEMS

5.1. Show how the convex combinations algorithm can be used to solve the system optimization assignment problem discussed in Section 3.4. Describe the algorithm in detail. Should the objective function values be used as the basis for a convergence criterion?

5.2. Find the user-equilibrium flow pattern over the network shown in Figure P5.1. The O–D flows are:

1 → 2	2 flow units
3 → 2	2 flow units

and the volume–delay curves are $t_a = 1 + 0.15(x/a)^4$, where a is the link's number shown in the figure. Perform three iterations of the convex combinations method.

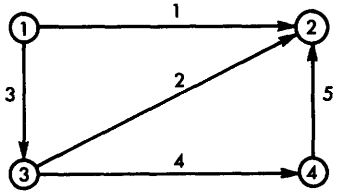


Figure P5.1

5.3. Streets and roadways cannot carry more than a certain amount of vehicular flow known as "capacity."

(a) Formulate the UE problem assuming that each network link is associated with a capacity c_a.

(b) Using the ideas outlined in the description of the feasible direction methods in Section 4.2, devise a modification to the convex combinations method described in Section 5.2 so that the algorithm would solve the capacity-constrained UE problem. You can assume that an initial *feasible* solution is given and that the performance curves are asymptotic to the capacity flow, for example, $t_a(x_a) = 1/(c_a - x_a)$.†

(c) Devise a method for finding an initial feasible solution for your algorithm.

†Mathematically, it is required that the performance curves would satisfy

$$\lim_{x_a \to c_a} \int_0^{x_a} t_a(\omega)\, d\omega = \infty$$

as shown by Daganzo (1977c).

5.4. Program [5.4] defines the linearization step of the convex combinations algorithm in terms of path flows. Outline all the other algorithmic steps associated with the application of this algorithm to the UE equivalent program, in terms of path flows. Can the algorithm be executed in terms of path flows?

5.5. In transportation planning applications the analyst may delete and add links to the network. Discuss the problems that the forward star representation poses for easy addition and deletion of links. How can these problems be overcome?

5.6. Use the label-correcting algorithm to find the minimum path from node 1 to all other nodes in the network depicted in Figure 5.4.

5.7. Explain why the particular strategy described in the text for managing the sequence list is advantageous for transportation networks. (As an aside, note that it is not particularly advantageous for other types of networks, such as communication networks.)

5.8. Write a subroutine that would take as input a network in forward star representation (transferred in a COMMON block or through GLOBAL variables) and an origin node (transferred in the calling statement), and would compute the minimum paths from the origin to all other nodes. The result is to be returned in another COMMON block or a set of GLOBAL variables (of predecessors).

5.9. Write an input subroutine that would read network link information (including starting node, end node, and performance function parameters) in random order and store it in a forward star representation. Include a flowchart.

5.10. Use the results of Problems 5.8, 5.9, and 4.10 (or 4.11) to write a computer code that will find the equilibrium flows over a transportation network.

5.11. The operation research literature includes algorithms for finding the minimum path from every node of a network to every other node. Indicate under which circumstances such algorithms will be useful and explain why they are not used in transportation planning applications.

III

Extensions of User Equilibrium

Part III extends in several directions the UE framework discussed in previous chapters. Chapter 6 focuses on the variable-demand problem, in which the O–D trip rates are not assumed to be fixed but rather a function of the equilibrium travel times. The concepts developed there are used to formulate a model in which the network includes the transit mode, in addition to the automobile network, and the modal split is modeled explicitly. Chapter 7 extends the original UE framework in another direction by assuming that the total number of trips leaving each origin node is known but that their destination is to be determined in conjunction with the equilibration process. Chapter 8 removes the assumption used throughout Part II that links are independent of each other, and Chapter 9 develops a framework in which many travel choice dimensions (i.e., whether to take a trip, where to go, by which mode, and what route to use) can be modeled simultaneously.

6

User Equilibrium with Variable Demand

The formulation of the user-equilibrium problem given in Chapter 3 assumes that the trip rate between every origin and every destination is fixed and known. In reality, however, these trip rates may be influenced by the level of service on the network. For example, as congestion increases, motorists may decide to use a different mode of travel (e.g., subway), shift the time of travel (outside the design period), or forego some trips altogether.

In order to take this phenomenon into account, the trip rate, q_{rs}, between every O–D pair r–s in the network can be assumed to be a function of the travel time between r and s. In other words,

$$q_{rs} = D_{rs}(u_{rs}) \qquad \forall\ r, s$$

where u_{rs} is the minimum travel time between r and s, and $D_{rs}(\cdot)$ is the demand function (for vehicular trips) between r and s. Typically, the form of the demand function will be the same for all O–D pairs. It can include, however, O–D specific parameters reflecting population size, income distribution, and vehicle ownership for origin nodes, as well as employment intensity or retail activity variables for destination nodes. For example, the demand function may be $q_{rs} = A_r B_s f(u_{rs})$, where A_r and B_s are known parameters associated with origin r and destination s, respectively, and $f(u_{rs})$ is a function of u_{rs}. The parameters of the demand function may also reflect other travel opportunities between O–D pair r–s by representing, for example, the level of service by other modes of travel (such as transit) and at other times of day, between r and s. The only argument of the demand function between r and s, however, is the O–D travel time, u_{rs}.

The demand function can be expected to be monotonically decreasing (or at least nonincreasing) in the O–D travel time. In other words, as u_{rs} grows, q_{rs} decreases, and vice versa. This function is also bounded from above. For example, the maximum number of vehicular trips generated between a given origin and a given destination in a certain design period may be bounded by the population size at the origin.

The problem addressed in this chapter is that of finding the link flows, the link travel times, and the *O–D trip rates* that satisfy the UE condition. At equilibrium, the travel times on all used paths between any O–D pair are equal, and are also equal to or less than the travel times on any unused paths. In addition, the O–D trip rates satisfy the demand functions. These conditions define the user equilibrium with variable demand (which is also known as the problem of UE with elastic demand).

The heuristic algorithms described in Section 5.1 can easily be modified and applied to the problem of determining the flow pattern associated with the variable-demand UE conditions. The results, however, would prove to be as unsatisfactory as before. This chapter formulates this problem as an equivalent minimization program and shows how this program can be solved. This equivalent minimization formulation is described in Section 6.1, which also discusses the related equivalence and uniqueness conditions. The solution algorithm is described in Section 6.2. Section 6.3 demonstrates how algorithms for equilibration of networks with fixed demand can be used to solve the variable-demand equilibrium problem. Finally, these ideas are applied in Section 6.4 to the formulation and solution of an extended equilibrium problem that includes two travel modes.

6.1 EQUIVALENT MINIMIZATION FORMULATION

This section presents the equivalent UE minimization program for the variable-demand case. It demonstrates that minimizing this program is, in fact, equivalent to solving the equilibrium equations and that the program has a unique solution.

Using the notation introduced in Chapter 3, consider the program

$$\min z(\mathbf{x}, \mathbf{q}) = \sum_a \int_0^{x_a} t_a(\omega)\, do - \sum_{rs} \int_0^{q_{rs}} D_{rs}^{-1}(\omega)\, d\omega \qquad [6.1a]$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\ r, s \qquad [6.1b]$$

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [6.1c]$$

$$q_{rs} \geq 0 \qquad \forall\ r, s \qquad [6.1d]$$

where $D_{rs}^{-1}(\cdot)$ is the inverse of the demand function associated with O–D pair r–s, $\mathbf{q} = (\ldots, q_{rs}, \ldots)$, and $x_a = \sum_{rs} \sum_k f_k^{rs} \delta_{a,k}^{rs}$, $\forall\ a$, as before.

The constraint set for the variable-demand problem (Eqs. [6.1b] and [6.1c]) is similar to the one used for the fixed-demand formulation (see Equations [3.3]), with the added requirement that q_{rs} be nonnegative. This requirement highlights the fact that q_{rs} is a variable in this formulation, rather than a constant.† The objective function in Eq. [6.1a] includes an additional term (as compared with Eq. [3.3a]), which is the sum of the integrals of the inverse demand functions. Note that $D_{rs}^{-1}(\cdot)$ is a decreasing function of the O–D travel times. This condition is used later in this section to establish the uniqueness of the minimum of the equivalent minimization program [6.1].

Motivation and Interpretation

Before the equivalence between the solution of program [6.1] and the network equilibrium conditions is shown formally, it is illustrated for the simple network shown in Figure 6.1a. This network is composed of exactly one link connecting a single O–D pair. The performance function of this link is $t = 1 + x$, where x is the flow, and the demand function is given by $x = 5 - t$. Figure 6.1b depicts this performance function and the (inverse) demand function. At the equilibrium point both of these equations hold and the solution of this simple system of equations is $x = 2$ and $t = 3$.

If the equivalent mathematical programming formulation is valid, this same solution should be obtained by minimizing program [6.1] for the simple network at hand. Given that the inverse demand function is $t = 5 - x$, the equivalent program is formulated as

$$\min z(x) = \int_0^x (1 + \omega)\, d\omega - \int_0^x (5 - \omega)\, d\omega \qquad [6.2]$$

In this case, x denotes both the link flow and the O–D flow and, naturally, no flow conservation constraints are needed. The flow that minimizes this expression is found by differentiating $z(x)$ to obtain (as the reader can check)

$$\frac{dz}{dx} = 2x - 4$$

and setting $dz/dx = 0$. This yields $x = 2$ (corresponding to $t = 3$), as expected.

The minimization represented by Eq. [6.2] can be viewed graphically as a maximization of the difference between the area under the inverse demand curve and the area under the performance function. [Recall that the minimization of $-f(\mathbf{x})$is equivalent to the maximization of$f(\mathbf{x})$.]

Figure 6.2a on page 138 illustrates this difference in areas for a case where the flow, x (a possible solution of the system), is smaller than x^* (the point where the demand and the performance lines intersect). Obviously, this area can be enlarged by increasing x (i.e., moving toward x^*). If, however,

†The nonnegativity of q_{rs} does not have to be required explicitly in this formulation since constraint [6.1d] is satisfied automatically if both constraints [6.1b] and [6.1c] are satisfied.

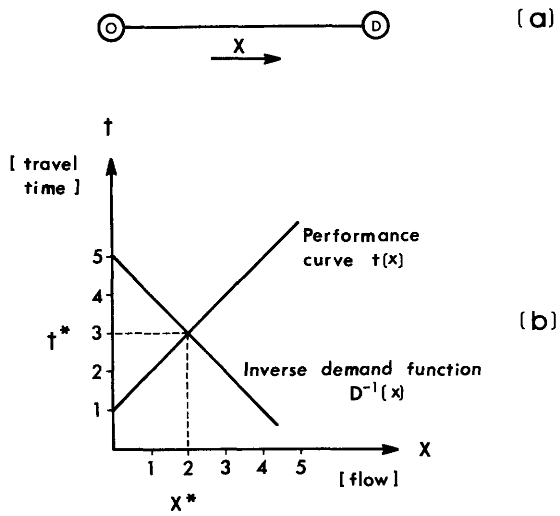


Figure 6.1 Equilibrium conditions with variable demand: (a) a one-link network; (b) performance and inverse demand functions for the one-link network in (a).

$x > x^*$, then one of the resulting triangular areas will have a minus sign and the resulting net area is that shown in Figure 6.2b (if $x \gg x^*$, the net area difference will be negative). This area, then, can be enlarged by decreasing x (again, moving toward x^*). The point at which the area difference is maximized is thus $x = x^*$; the associated area is depicted in Figure 6.2c. Maximizing this area therefore brings about the same result as the solution of the equilibrium equations.†

The remainder of this section shows formally that the solution of the minimization program [6.1] is equivalent to the user-equilibrium conditions when demand is variable. This program is also shown to have a unique solution in terms of the link flows and the O–D trip rates.

Equivalence Conditions

As with the simple UE program, in order to ensure that the equilibrium conditions are met at the point where program [6.1] is minimized, the first-order conditions of the program must be equivalent to the equilibrium conditions. The general form of these conditions for a mathematical program with equality and nonnegativity constraints was derived in Section 2.3. Using the same procedure here, the Lagrangian of the program is set up, differentiated, and solved.

†Note that this area cannot be interpreted as consumer's surplus. As noted by Beckman et al. (1956), the link performance functions reflect average and not marginal travel time (or travel cost). Such an interpretation would, therefore, be wrong.

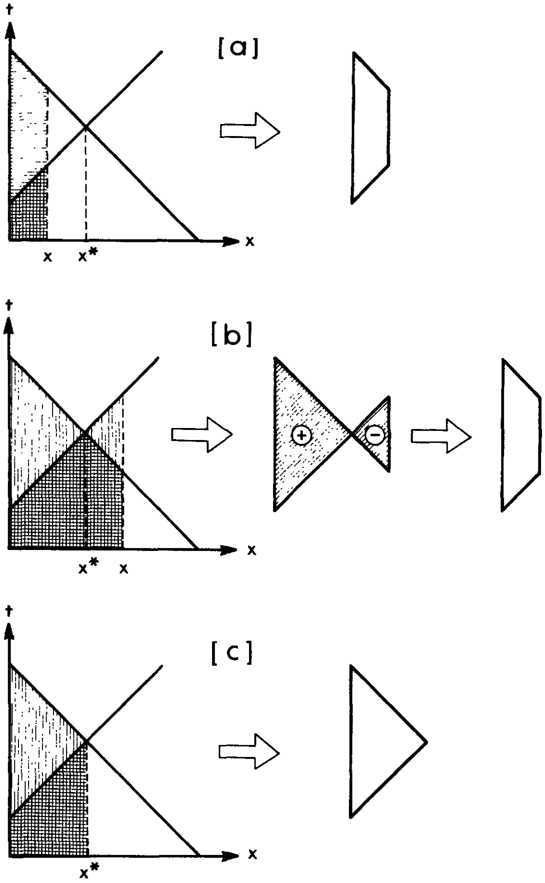


Figure 6.2 Interpretation of the variable-demand objective function as maximization of the areas between the inverse demand and the performance functions.

Let the objective function in program [6.1] be decomposed as follows:

$$\min z(\mathbf{x}, \mathbf{q}) = \min \{z_1(\mathbf{x}) - z_2(\mathbf{q})\} \qquad [6.3]$$

where

$$z_1(\mathbf{x}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega$$

$$z_2(\mathbf{q}) = \sum_{rs} \int_0^{q_{rs}} D_{rs}^{-1}(\omega)\, d\omega$$

Note that $z_1(\mathbf{x})$ can be written in terms of path flows by using the incidence relationships, and thus $x_a = x_a(\mathbf{f})$. The Lagrangian associated with [6.1] can then be written as

$$L(\mathbf{f}, \mathbf{q}, \mathbf{u}) = z_1[\mathbf{x}(\mathbf{f})] + z_2(\mathbf{q}) + \sum_{rs} u_{rs}\left(q_{rs} - \sum_k f_k^{rs}\right) \qquad [6.4a]$$

where $\mathbf{u} = (\ldots, u_{rs}, \ldots)$ is the vector of Lagrange multipliers associated with constraints [6.1b]. This Lagrangian should be minimized with respect to the flow variables (and maximized with respect to the dual variables) subject to the following constraints:†

$$f_k^{rs} \geq 0 \qquad \forall\, k, r, s \tag{6.4b}$$

$$q_{rs} \geq 0 \qquad \forall\, r, s \tag{6.4c}$$

The first-order conditions for this program are:

$$f_k^{rs} \frac{\partial L(\mathbf{f}, \mathbf{q}, \mathbf{u})}{\partial f_k^{rs}} = 0 \qquad \forall\, k, r, s \tag{6.5a}$$

$$\frac{\partial L(\mathbf{f}, \mathbf{q}, \mathbf{u})}{\partial f_k^{rs}} \geq 0 \qquad \forall\, k, r, s \tag{6.5b}$$

$$q_{rs} \frac{\partial L(\mathbf{f}, \mathbf{q}, \mathbf{u})}{\partial q_{rs}} = 0 \qquad \forall\, r, s \tag{6.5c}$$

$$\frac{\partial L(\mathbf{f}, \mathbf{q}, \mathbf{u})}{\partial q_{rs}} \geq 0 \qquad \forall\, r, s \tag{6.5d}$$

$$\frac{\partial L(\mathbf{f}, \mathbf{q}, \mathbf{u})}{\partial u_{rs}} = 0 \qquad \forall\, r, s \tag{6.5e}$$

$$f_k^{rs} \geq 0 \qquad \forall\, k, r, s \tag{6.5f}$$

$$q_{rs} \geq 0 \qquad \forall\, r, s \tag{6.5g}$$

Conditions [6.5a] and [6.5b] replace the standard "zero derivatives" condition due to the nonnegativeity constraint on the path-flow variables, $\{f_k^{rs}\}$. The same holds true for constraints [6.5c] and [6.5d], which play the same role for the O–D flow variables, $\{q_{rs}\}$. Condition [6.5e] results from the Lagrangian being maximized with respect to **u**, which is unconstrained. In order to apply these conditions to the program at hand, the derivatives of the Lagrangian with respect to the path flow and the O–D trip rates should be calculated explicitly.

The derivative of the Lagrangian with respect to a general path-flow variable, f_l^{mn}, is

$$\frac{\partial L(\cdot)}{\partial f_l^{mn}} = \frac{\partial}{\partial f_l^{mn}} \left\{ z_1[\mathbf{x}(\mathbf{f})] - z_2(\mathbf{q}) + \sum_{rs} u_{rs}\left(q_{rs} - \sum_k f_k^{rs} \right) \right\}$$

The second term in the brackets, $z_2(\mathbf{q})$, is not a function of **f** and can therefore be dropped from the derivative. What is left is identical to the derivative of the

†The nonnegativity of q_{rs} should be explicitly required here since the problem of finding the stationary point of the Lagrangian is not subject to constraints [6.1b]. These constraints are part of the Lagrangian function itself.

original UE program with respect to a path-flow variable (see Eq. [3.15]). Consequently,

$$\frac{\partial L(\mathbf{f}, \mathbf{q}, \mathbf{u})}{\partial f_l^{mn}} = \frac{\partial L(\mathbf{f}, \mathbf{u})}{\partial f_l^{mn}} = c_l^{mn} - u_{mn} \qquad [6.6]$$

where c_l^{mn} is the travel time on the lth path connecting origin m to destination n.

The derivative of the Lagrangian with respect to a typical O–D trip rate, q_{mn}, can be simplified to $\partial L(\mathbf{q}, \mathbf{u})/\partial q_{mn}$, since $z_1(\mathbf{x})$ can be dropped from the derivative. This derivative is therefore

$$\frac{\partial L(\mathbf{q}, \mathbf{u})}{\partial q_{mn}} = \frac{\partial}{\partial q_{mn}}\left[-\sum_{rs}\int_0^{q_{rs}} D_{rs}^{-1}(\omega)\, d\omega + \sum_{rs} u_{rs}\left(q_{rs} - \sum_k f_k^{rs}\right)\right]$$

$$= -D_{mn}^{-1}(q_{mn}) + u_{mn} \qquad [6.7]$$

The first-order conditions are thus (see Eqs. 6.5)

$$f_k^{rs}(c_k^{rs} - u_{rs}) = 0 \qquad \forall\ k, r, s \qquad [6.8a]$$

$$c_k^{rs} - u_{rs} \geq 0 \qquad \forall\ k, r, s \qquad [6.8b]$$

$$q_{rs}[u_{rs} - D_{rs}^{-1}(q_{rs})] = 0 \qquad \forall\ r, s \qquad [6.8c]$$

$$u_{rs} - D_{rs}^{-1}(q_{rs}) \geq 0 \qquad \forall\ r, s \qquad [6.8d]$$

$$q_{rs} - \sum_k f_k^{rs} = 0 \qquad \forall\ r, s \qquad [6.8e]$$

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [6.8f]$$

$$q_{rs} \geq 0 \qquad \forall\ r, s \qquad [6.8g]$$

Equations [6.8a] and [6.8b] state the user-equilibrium conditions as in the fixed-demand case analyzed in Chapter 3 (see Eqs. [3.16a] and [3.16b]).† These equations specify that the travel time on any path connecting a given O–D pair must equal the minimum-path travel time if the flow on this path is positive. If the flow on this path is zero, its travel time must be greater than (or equal to) the travel time on the shortest path.

Conditions [6.8c] and [6.8d] have a very similar interpretation. If the O–D trip rate between a given O–D pair, say r–s, is positive, then, for Eq. [6.8c] to hold, Eq. [6.8d] must hold as an equality. In other words,

$$u_{rs} = D_{rs}^{-1}(q_{rs}) \qquad [6.9a]$$

Both sides of this equation can be inverted to obtain

$$D_{rs}(u_{rs}) = q_{rs} \qquad [6.9b]$$

†The variables in Eqs. [6.8] indicate values at the optimal solution point. The asterisks are omitted for clarity of presentation.

Thus, if $q_{rs} > 0$, it must be given by the demand function. If, however, $q_{rs} = 0$, then from Eq. [6.8d]

$$u_{rs} \geq D_{rs}^{-1}(q_{rs})$$

meaning that the O–D travel time may be too high to induce any O–D flow.

Uniqueness Conditions

To prove that the equivalent variable-demand UE program [6.1] has a unique solution, it is sufficient to show that the objective function is strictly convex everywhere. The first part of the objective function,

$$z_1(\mathbf{x}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega$$

is strictly convex if the functions $t_a(x_a)$ are increasing in x_a, as shown in Section 3.3.

The demand function for each O–D pair, $D_{rs}(u_{rs})$, is a monotonically decreasing function of its argument. It follows that its inverse, $D_{rs}^{-1}(\cdot)$, should also be a decreasing function. The integral of a decreasing function is strictly concave and the sum of strictly concave functions is a strictly concave function. Thus

$$z_2(\mathbf{q}) = \sum_{rs} \int_0^{q_{rs}} D_{rs}^{-1}(\omega)\, d\omega$$

is strictly concave. The negative of a strictly concave function is, however, a strictly convex function. Thus

$$z(\mathbf{x}, \mathbf{q}) = z_1(\mathbf{x}) - z_2(\mathbf{q})$$

is the sum of two strictly convex functions, meaning that $z(\mathbf{x}, \mathbf{q})$ is strictly convex.

The strict convexity of $z(\mathbf{x}, \mathbf{q})$ implies that program [6.1] has a unique solution in terms of O–D trip rates and link flows. As with the simpler fixed-demand problem discussed in Chapter 3, the solution of the variable-demand program (and the equilibrium condition itself) is not unique in terms of path flows. In other words, the equilibrium link flows can be achieved through many combinations of path flows.

6.2 SOLUTION ALGORITHM

This section demonstrates how the equivalent minimization program representing the variable-demand equilibrium problem can be solved. The formulation of this program was shown in the preceding section to be

$$\min z(\mathbf{x}, \mathbf{q}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega - \sum_{rs} \int_0^{q_{rs}} D_{rs}^{-1}(\omega)\, d\omega \qquad [6.10a]$$

subject to

$$f_k^{rs} \geq 0 \qquad \forall\ r, s \qquad [6.10b]$$

$$q_{rs} \leq \bar{q}_{rs} \qquad \forall\ r, s \qquad [6.10c]$$

$$q_{rs} = \sum_k f_k^{rs} \qquad \forall\ r, s \qquad [6.10d]$$

Constraint [6.10c] was added to this formulation as an upper bound on the total flow from origin r to destination s (recall that the demand function was assumed to be bounded from above). This constraint is added for computational reasons (as explained below); it obviously does not change the problem if it is set high enough (so that it would never be binding).

Program [6.10] can be formulated exclusively in terms of path flow variables. The link flows are then given by the incidence relationships,

$$x_a = x_a(\mathbf{f}) = \sum_{rs} \sum_k f_k^{rs} \delta_{a,k}^{rs} \qquad \forall\ a$$

and the path-flow variables are given by the flow conservation constraint

$$q_{rs} = q_{rs}(\mathbf{f}^{rs}) = \sum_k f_k^{rs} \qquad \forall\ r, s$$

Using these transformations, the equivalent program for user equilibrium with variable demand can be expressed as

$$\min z(\mathbf{f}) = \sum_a \int_0^{x_a(\mathbf{f})} t_a(\omega)\, d\omega - \sum_{rs} \int_0^{q_{rs}(\mathbf{f}^{rs})} D_{rs}^{-1}(\omega)\, d\omega \qquad [6.11a]$$

subject to

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [6.11b]$$

$$\sum_k f_k^{rs} \leq \bar{q}_r \qquad \forall\ r, s \qquad [6.11c]$$

The approach used to minimize this program is based, again, on the convex combinations algorithm. As shown in Section 4.1, this algorithm is a feasible descent method. The descent vector at the nth iteration is given by the direction from the current solution to the point that solves the linear approximation of the objective function. This linear approximation results in a linear program that is expressed in terms of a set of auxiliary variables (see Eqs. [4.23]). For the UE program the variables are the path flows, $\{f_k^{rs}\}$, and the associated link flows, $\{x_a\}$. The corresponding sets of auxiliary variables are $\{g_k^{rs}\}$ and $\{y_a\}$, as shown in Section 5.2 (see Eqs. [5.3]). The minimum of the resulting auxiliary LP was found by solving

$$\min z^n(\mathbf{f}) = \sum_{rs} \sum_k \frac{\partial z(f^n)}{\partial f_k^{rs}} g_k^{rs} \qquad [6.12]$$

subject to the original constraints, in terms of $\{g_k^{rs}\}$ (see Eqs. [5.4]). The minimum of this program, was then used to define the descent direction.

As in Section 5.2, the variable g_k^{rs} denotes the auxiliary path flow corresponding to f_k^{rs}. In other words, $\sum_{rs} \sum_k g_k^{rs}\delta_{a,k}^{rs} = y_a$, $\forall\ a$, where y_a represents the auxiliary link-flow variable associated with link a. Note that unlike the case in Eqs. [5.3] (where $\sum_k g_k^{rs} = q_{rs}$, $\forall\ r, s$), the O–D flows are variables in this program and thus $\sum_k g_k^{rs} = v_{rs}$, where v_{rs} is the auxiliary O–D flow variable corresponding to q_{rs}.

To solve program [6.11] by using the convex combinations method, partial derivatives of the objective function $z(\mathbf{f})$ have to be calculated since those are the coefficients of the auxiliary LP. A representative partial derivative of $z(\mathbf{f})$ at the nth iteration of the algorithm is given by

$$\frac{\partial z[x(\mathbf{f}^n), q(\mathbf{f}^n)]}{\partial f_l^{mn}} = c_l^{mn}(\mathbf{f}^n) - D_{mn}^{-1}(\mathbf{f}^n) \qquad [6.13]$$

The linear program corresponding to the nth iteration of the solution of program [6.11] can now be expressed as

$$\min z^n(\mathbf{g}) = \sum_{rs} \sum_k [c_k^{rsn} - D_{rs}^{-1}(q_{rs}^n)]g_k^{rs} \qquad [6.14a]$$

subject to

$$g_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [6.14b]$$

$$\sum_k g_k^{rs} \leq \bar{q}_{rs} \qquad \forall\ r, s \qquad [6.14c]$$

where c_k^{rsn} is the travel time on path k between origin r and destination s at the nth iteration [i.e., $c_k^{rsn} = c_k^{rs}(\mathbf{f}^n)$], and q_{rs}^n is the value of the O–D flow between r and s at the nth iteration of the algorithm.

In solving program [6.14], note that the term in brackets in the objective function (Eq. [6.14a]) is constant with respect to $\mathbf{g}$. Furthermore, constraint [6.14c] applies to each origin–destination pair independently. These two observations imply that $z^n(\mathbf{g})$ can be minimized by concentrating on each O–D pair individually. The problem for O–D pair r–s is given by

$$\min z^{rsn}(\mathbf{g}^{rs}) = \sum_k [c_k^{rsn} - D_{rs}^{-1}(q_{rs}^n)]g_k^{rs} \qquad [6.15a]$$

subject to

$$g_k^{rs} \geq 0 \qquad \forall\ k \qquad [6.15b]$$

$$\sum_k g_k^{rs} \leq \bar{q}_{rs} \qquad [6.15c]$$

The set of path flows, $\mathbf{g}^{rs}$, that solves program [6.15] can be determined simply by inspection. The solution requires assignment of as much flow as possible (i.e., $\bar{q}_{rs}$) to the path k for which $[c_k^{rsn} - D_{rs}^{-1}(q_{rs}^n)]$ is the smallest. If this quantity is positive for all paths, no flow should be assigned to any path. The use of such a procedure will result in the lowest possible value of objective function [6.15a]. For a given O–D pair, the path for which $[c_k^{rsn} - D_{rs}^{-1}(q_{rs}^n)]$ is the most negative is m such that $c_m^{rsn} = \min_{\forall k} \{c_k^{rsn}\}$ (c_m^{rsn} also equals the O–D travel time at the nth iteration, u_{rs}^n). This minimization rule can, then, be stated formally as follows:

$$\text{If } c_m^{rsn} < D_{rs}^{-1}(q_{rs}^n), \text{ set } g_m^{rsn} = \bar{q}_{rs} \quad \text{and} \quad g_k^{rsn} = 0 \quad \forall\, k \neq m \qquad [6.16a]$$

$$\text{If } c_m^{rsn} > D_{rs}^{-1}(q_{rs}^n), \text{ set } g_k^{rsn} = 0 \quad \forall\, k \qquad [6.16b]$$

In case $c_m^{rsn} = D_{rs}^{-1}(q_{rs}^n)$, either [6.16a] or [6.16b] can be followed. The same procedure can be repeated for each origin–destination pair in order to minimize program [6.14].†

Once this assignment process is completed, the auxiliary link-flow and O–D trip-rate variables can be calculated as follows:

$$y_a^n = \sum_{rs} \sum_k g_k^{rsn} \delta_{a,k}^{rs} \qquad \forall\, a \qquad [6.17a]$$

$$v_{rs}^n = \sum_k g_k^{rsn} \qquad \forall\, r, s \qquad [6.17b]$$

With these variables the descent direction is a vector $\mathbf{d}^n$ with elements $(y_a^n - x_a^n)$ and $(v_{rs}^n - q_{rs}^n)$.

Note that the direction-finding step requires computation of the shortest paths connecting all O–D pairs at each iteration, as was the case with the fixed demand problem. The additional computational effort required for the variable demand problem is very small; it includes only a comparison of the travel time on the current shortest path with the current value of the inverse demand function (see Eqs. [6.16]).

The move size in the convex combinations method is determined by finding a scalar, α_n, which minimizes the objective function along the descent direction. In other words, α_n is the value of α that minimizes $z(\mathbf{x}^n + \alpha\mathbf{d}^n)$. For the variable-demand minimization [6.10], this one-dimensional minimization program is given by

$$\min z(\alpha) = \sum_a \int_0^{x_a^n + \alpha(y_a^n - x_a^n)} t_a(\omega)\, d\omega - \sum_{rs} \int_0^{q_{rs}^n + \alpha(v_{rs}^n - q_{rs}^n)} D_{rs}^{-1}(\omega)\, d\omega \qquad [6.18a]$$

subject to

$$0 \leq \alpha \leq 1 \qquad [6.18b]$$

This problem can be solved by using any of the interval reduction methods discussed in Section 4.1.

Once the optimal move size, α_n, is found, the link and O–D flow variables can be updated by setting

$$x_a^{n+1} = x_a^n + \alpha_n(y_a^n - x_a^n) \qquad [6.19a]$$

$$q_{rs}^{n+1} = q_{rs}^n + \alpha_n(v_{rs}^n - q_{rs}^n) \qquad [6.19b]$$

†The need for the upper bound constraint [6.10c] in the original formulation is now clear; without it, program [6.14] may be unbounded.

If convergence is not achieved, the link travel times and the inverse demand function are updated [i.e., $t_a(x_a^n)$ and $D_{rs}^{-1}(q_{rs}^n)$ are computed], the new minimum path connecting each O–D pair is calculated and a new set of auxiliary flows is obtained by following Eqs. [6.16].

Convergence can be checked for by using the objective function values at successive iteration or a flow-based criterion, as discussed in conjunction with fixed-demand UE problems. The equilibrium conditions can, however, be checked for directly, by calculating the closeness of the O–D travel time to those implied by the (inverse) demand function, in addition to the closeness of consecutive O–D travel times (see Eq. [5.8a]). In other words, at convergence

$$u_{rs}^n \simeq u_{rs}^{n-1} \qquad \forall\, r, s$$

and

$$D_{rs}^{-1}(q_{rs}^n) \simeq u_{rs}^n \qquad \text{for all } r\text{–}s \text{ for which } q_{rs}^n > 0.$$

These requirements imply a convergence criterion such as

$$\sum_{rs} \frac{|D_{rs}^{-1}(q_{rs}^n) - u_{rs}^n|}{u_{rs}^n} + \sum_{rs} \frac{|u_{rs}^n - u_{rs}^{n-1}|}{u_{rs}^n} \leq \kappa \qquad [6.20]$$

where κ is a dimensionless convergence criterion and the first summation includes only the O–D pairs with positive flow. (Why?) In practice, each of the sums in Eq. [6.20] can be used independently as a basis for a convergence criterion.

The algorithm can then be summarized as follows:

Step 0: *Initialization.* Find an initial feasible flow pattern $\{x_a^n\}$, $\{q_{rs}^n\}$. Set $n := 1$.

Step 1: *Update.* Set $t_a^n = t_a(x_a^n)$, $\forall\, a$; compute $D_{rs}^{-1}(q_{rs}^n)$, $\forall\, r, s$.

Step 2: *Direction finding.* Compute the shortest path, m, between each O–D pair r–s based on $\{t_a^n\}$; execute the assignment rule shown in Eqs. [6.16]. Following the transformation of Eqs. [6.17], this assignment yields an auxiliary flow pattern $\{y_a^n\}$, $\{v_{rs}^n\}$.

Step 3: *Move size.* Find α_n that solves program [6.18].

Step 4: *Flow update.* Find $\{x_a^{n+1}\}$ and $\{q_{rs}^{n+1}\}$ by using Eqs. [6.19].

Step 5: *Convergence criterion.* If inequality [6.20] holds, terminate. Otherwise, set $n := n + 1$ and go to step 1.

The next section presents a different perspective on the solution algorithm, that of using a special network representation. This approach to solving network equilibrium problems is used extensively in the remainder of the text.

6.3 SOLUTION BY NETWORK REPRESENTATION

Adapting the convex combinations algorithm to the solution of the variable-demand assignment problem is a relatively straightforward task. Even this adaptation, however, may be unnecessary; a simple change in the representation of the problem can make it amenable for solution with fixed-demand equilibration algorithms. This section deals with this topic, and includes two such formulations, the zero-cost overflow formulation and the excess-demand formulation. In both cases, the approach requires the addition of "virtual" or "dummy" nodes and links to the network. These network modifications make it possible to solve a fixed-demand UE problem over the augmented network, obtaining a solution for the variable-demand problem over the original network.

The Zero-Cost Overflow Formulation

Figure 6.3a depicts a network representing an urban street system, and highlights a particular origin–destination pair, r–s. Figure 6.3b shows a modification of this network in which every O–D pair is augmented to include a "dummy" destination node (designated r' in Figure 6.3b) in addition to node s.† The modified representation of O–D pair r–s also includes a "generating link" (indexed sr') leading from s to r' and a "zero-cost overflow link" (indexed rr') leading directly from the origin r to the new dummy destination node, r'. The performance curves for these new links are‡

$$t_{sr'} = -D_{rs}^{-1}(\cdot) \qquad \forall\ r, s \tag{6.21a}$$

for the generating link, and

$$t_{rr'} = 0 \qquad \forall\ r, s \tag{6.21b}$$

for the (zero-cost) overflow link. To help distinguish between the various components of the modified network, the original graph representing the streets and intersections will be referred to from now on as the *basic network*. Thus the basic network corresponding to the one in Figure 6.3b is that shown in Figure 6.3a.

Assume now that a fixed number of trips, $\bar{q}_{rs}$, as to be assigned between

†As it turns out, it is sufficient to have one dummy destination node associated with every origin. This is the reason for the notation used for these nodes. The explanation in this section may be easier to understand though, if the reader assumes that each destination dummy node is unique to each O–D pair.

‡Even though the travel time on link s–r' may be negative, the modified network includes no negative cycles, by construction. In other words, there are no situations in which it is advantageous to keep traversing a given path since the net travel time along this path is negative. Such situations may arise in some networks containing negative-travel-time links, in cases where these links are part of a path leading back to its point of origin (a cyclic path, or a cycle). The modified network cannot, however, contain cyclic paths that are based on the dummy links.

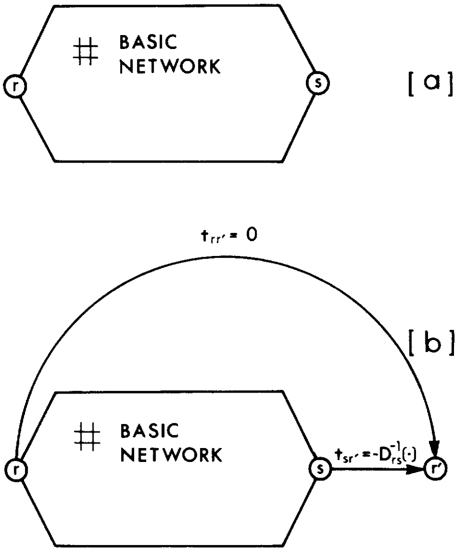


Figure 6.3 The zero-cost overflow network representation: (a) the basic network; (b) the added node and links for O–D pair *r–s*.

every origin and destination of the *modified* network (e.g., between r and r' for the O–D pair shown in the figure). This assignment problem can be formulated as an equivalent minimization program (as was done in Chapter 3) and solved with an appropriate algorithm (e.g., the convex combinations method). The formulation would be

$$\min z(x) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega + \sum_{rs} \int_0^{x_{sr'}} t_{sr'}(\omega)\, d\omega + \sum_{rs} \int_0^{x_{rr'}} t_{rr'}(\omega)\, d\omega \qquad [6.22a]$$

subject to

$$\sum_k f_k^{rs} + x_{rr'} = \bar{q}_{rs} \qquad \forall\ r, s \qquad [6.22b]$$

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [6.22c]$$

$$x_{rr'} \geq 0 \qquad \forall\ r, s \qquad [6.22d]$$

The definition of the performance functions and flow variables related to the dummy links (see Eqs. [6.21]) imply that $t_{sr'}(\cdot) = -D_{rs}^{-1}(\cdot)$, $x_{sr'} = q_{rs}$, and $t_{rr'}(\cdot) = 0$. Thus objective function [6.22a] is identical to [6.10a]. Furthermore, the network representation guarantees that the flow over the basic network is $q_{rs} = \sum_k f_k^{rs}$ and that the excess demand, $x_{rr'} = \bar{q}_{rs} - q_{rs}$, overflows onto the added link $r \rightarrow r'$. Consequently, the equivalent UE program [6.22] for the *modified* network is identical to the equivalent variable-demand UE program for the *basic* network as formulated in Eqs. [6.10]. (Constraint [6.22b] in conjunction with [6.22d] is equivalent to [6.10c], with $x_{rr'}$ representing the slack flow.)

Program [6.22] can be solved with a standard convex combinations algorithm by applying it to the modified network. At the solution point the travel time over all used paths has to be equal to u_{rs} for each O–D pair r–s and thus, at equilibrium.

$$t_{rr'} = u_{rs} + t_{sr'} \qquad \forall\ r, s$$

Since $t_{rr'} = 0$ and $t_{sr'} = -D_{rs}^{-1}(q_{rs})$, this means that at equilibrium

$$u_{rs} = D_{rs}^{-1}(q_{rs}) \qquad \forall\ r, s$$

Thus, by adding one node and two links (per O–D pair) to the network description, any fixed-demand traffic assignment algorithm can be used to solve the problem of equilibrium assignment with variable demand.

Excess-Demand Formulation

The variable-demand problem can be solved with an even more efficient fixed-demand formulation, again through a network representation. In this representation, the variable e_{rs} denotes the excess demand, that is, trips not accommodated between origin r and destination s ($e_{rs} = \bar{q}_{rs} - q_{rs}$). Let $W_{rs}(\cdot)$ denote the argument-complementing function of the inverse demand, that is, $W_{rs}(e_{rs}) = D_{rs}^{-1}(q_{rs})$, $\forall\ r, s$. These two function are illustrated in Figure 6.4.

Consider now the objective function of the equivalent variable-demand formulation [6.1a]:

$$\min z(\mathbf{x}, \mathbf{q}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega - \sum_{rs} \int_0^{q_{rs}} D_{rs}^{-1}(\omega)\, d\omega$$

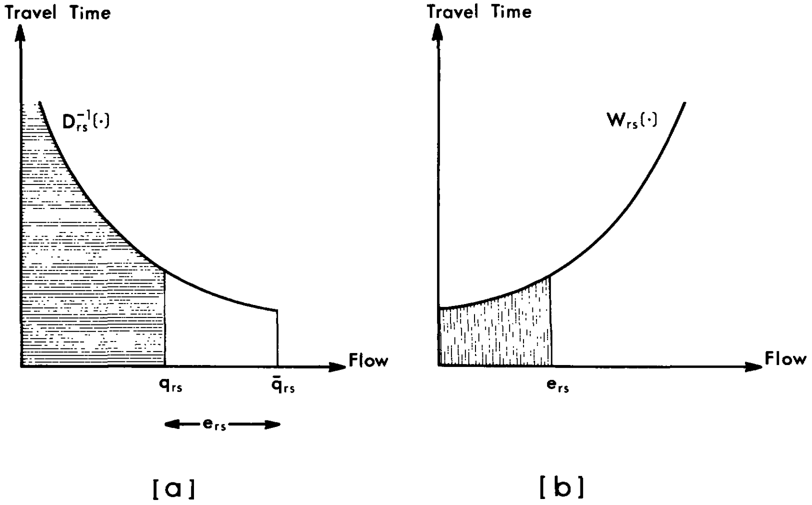


Figure 6.4 Inverse demand function and the associated argument-completing function.

Each term in the second sum can be decomposed into two integrals (see Figure 6.4a) as follows:

$$\int_0^{q_{rs}} D_{rs}^{-1}(\omega)\,d\omega = \int_0^{\bar{q}_{rs}} D_{rs}^{-1}(\omega)\,d\omega - \int_{q_{rs}}^{\bar{q}_{rs}} D_{rs}^{-1}(\omega)\,d\omega \qquad [6.23a]$$

The first integral on the right-hand side of Eq. [6.23a] is a constant that can be dropped from the objective function (it would not affect the result of the minimization). The second integral can be expressed in terms of the excess demand with a simple change of variable:† $e_{rs} = \bar{q}_{rs} - q_{rs}$. The new variable of integration is $v = \bar{q}_{rs} - \omega$. The limits of the integration become $(\bar{q}_{rs} - q_{rs})$ and zero and thus

$$\begin{aligned} -\int_{q_{rs}}^{\bar{q}_{rs}} D_{rs}^{-1}(\omega)\,d\omega &= -\int_{\bar{q}_{rs}-q_{rs}}^{0} D_{rs}^{-1}(\bar{q}_{rs} - v)(-dv) \\ &= -\int_0^{e_{rs}} W_{rs}(v)\,dv \qquad [6.23b] \end{aligned}$$

With the integral decomposition [6.23a] and the last expression [6.23b], the objective function of the equivalent variable-demand formulation can be expressed as

$$\min z(\mathbf{x}, \mathbf{e}) = \sum_a \int_0^{x_a} t_a(\omega)\,d\omega + \sum_{rs} \int_0^{e_{rs}} W_{rs}(v)\,dv \qquad [6.24a]$$

where $\mathbf{e} = (\ldots, e_{rs}, \ldots)$ denotes the vector of excess-demand flows. This minimization is subject to the following constraints:

$$\sum_k f_k^{rs} + e_{rs} = \bar{q}_{rs} \qquad \forall\, r, s \qquad [6.24b]$$

$$f_k^{rs} \geq 0 \qquad \forall\, k, r, s \qquad [6.24c]$$

$$e_{rs} \geq 0 \qquad \forall\, r, s \qquad [6.24d]$$

Obviously, the incidence relationships hold as well. By inspecting objective function [6.24a] for this program, it becomes clear that the function $W_{rs}(e_{rs})$ has all the properties of a link performance function. An additional link that would carry the flow e_{rs} can thus be defined. The positioning of this link can be inferred from the flow conservation constraint (Eq. [6.24b]); this link should carry the excess-demand flow and therefore should directly connect the origin to the destination for each O–D pair. The resulting network representation is shown in Figure 6.5.

†Recall that a change of variable operation for definite integrals is based on the formula

$$\int_a^b f[g(x)]\left(\frac{dg}{dx}\right)dx = \int_{g(a)}^{g(b)} f(g)\,dg$$

where $g(\cdot)$ and its derivative are continuous functions defined on the interval $[a, b]$, and $f(\cdot)$ is a continuous function in the interval $[g(a), g(b)]$ [which is the range of $g(\cdot)$].

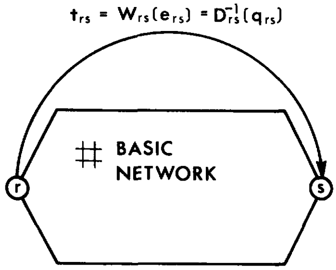


Figure 6.5 Excess-demand network representation for O–D pair *r–s*.

Program [6.24], which applies to the excess-demand network representation, can be solved with a standard UE minimization method, such as the convex combinations algorithm. At equilibrium, $W_{rs}(e_{rs}) = u_{rs}$ (see Figure 6.5), and since $W_{rs}(e_{rs})$ equals $D_{rs}^{-1}(q_{rs})$ by definition, the equilibrium conditions for the variable-demand equilibrium are met. This formulation is more efficient than the zero-cost overflow representation since it requires less computer storage due to the smaller number of additional links used in the representation. It should, therefore, always be preferred to the zero-cost overflow formulation, the merit of which is mostly pedagogical.

It is important to realize that each link in the network representation corresponds to a flow variable. This is true for any network problem. This observation is the basis for both the zero-cost overflow and the excess-demand formulations, as well as several other formulations and problem representations discussed throughout this text.

Example

The excess-demand formulation can be used to solve the simple network example introduced in Section 6.1 (see Figure 6.1). In this one-link "network," the performance curve was given by $t = 1 + x$ and the demand curve by $x = 5 - t$. To solve this problem with the excess-demand flow representation, a flow bound (of, say, $\bar{q} = 5$ flow units) has to be introduced. Let the inverse demand function, $D^{-1}(\cdot)$, be $t = 5 - x$ and let the excess-demand function, $W(\cdot)$, be given by $t = e$, since $e = 5 - x$.

The problem can now be represented by the two-link network shown in Figure 6.6. The performance function for the added link is $t = e$ and the equivalent UE minimization program can be stated for this example as follows

$$\min z(x, e) = \int_0^x (1 + \omega)\, d\omega + \int_0^e \omega\, d\omega$$

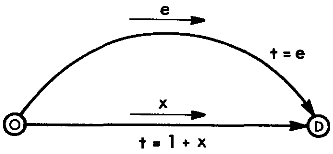


Figure 6.6 Excess-demand representation for a one-link problem with variable demand.

subject to

$$x + e = 5$$

$$x, e \geq 0$$

For this example the objective function is

$$\min z(x, e) = x + \tfrac{1}{2}x^2 + \tfrac{1}{2}e^2$$

The minimum of this function can be found by substituting $e = 5 - x$, differentiating, and solving in terms of x to obtain $x = 2$. This x value is identical to the one obtained in Section 6.1 by using a direct solution method. The associated value of the excess demand in this example is $e = 3$. If the demand bound had been chosen to be, for example, $\bar{q} = 20$ or $\bar{q} = 4$, the resulting solution would have been $e = 18$ or $e = 2$, respectively, with $x = 2$ in both cases. Thus the demand bound does not change the problem or its solution when it is set high enough.

Computational Considerations

Using the convex combinations algorithm, the computational effort required to solve variable-demand UE programs is significantly higher than that required for solving fixed-demand UE programs over the same networks. This phenomenon, however, is not obvious since there are many aspects of these two problems from which one could have concluded that the difference in computational efforts is modest.

To begin with, the number of minimum path trees computed for both problems is identical. The difference in the computational effort may, therefore, be rooted either in the added calculation associated with the flow assignment between each O–D pair, in the calculation of the optimal move size, or in the number of iterations of the algorithms.

The first factor is the added computational effort associated with each O–D pair. The added effort associated with variable-demand problems involves one comparison (see Eq. [6.16]) for each O–D pair. Even though the number of O–D pairs may be large, the total added effort is small. This computational effort argument can be viewed in the context of the excess-demand network representation. Here the variable-demand problem is solved as a fixed-demand problem over a larger network. The effect of the added (excess-demand) links on the calculation of the minimum path trees is smaller than that of regular network links. The reason is the special position of the excess-demand links, which implies that each such link is used in the calculation of only one minimum path tree. Consequently, their impact is not as large as implied by the (possibly) large number of these links.

The second factor is the execution of the one-dimensional minimization in the process of calculating the move size. This minimization may be slow relative to the one-dimensional search in the fixed-demand case due to the added terms in the objective function. In absolute terms, however, the one-dimensional search constitutes only a small fraction of the computational effort associated with solving network-equilibrium problems. Thus the overall effect of a slower one-dimensional search is small.

The main reason that the convex-combination algorithm requires a large computational effort to solve variable-demand problems is the number of iterations required. This algorithm takes a significantly larger number of iterations to converge for the variable-demand problem than for a compatible fixed-demand problem. The reason for this can be intuitively understood by considering, again, the excess-demand formulation. The position of the excess-demand links means that flow can be assigned to each of these links only when the O–D pair to which it belongs is considered. Thus each excess-demand link is being considered for flow assignment only once at each iteration (unlike most other links in the network which carry flow between many origins and many destinations). Consequently, many iterations may be required before the flow on the excess links can stabilize and equilibrium be obtained.

Interestingly, the number of iterations required for convergence of the variable-demand UE problem can be influenced by the analyst through the choice of the upper bounds on the O–D flows, $\{\bar{q}_{rs}\}$. As the algorithm proceeds these bounds are assigned either to the network or to the excess-demand links. Relaxed bounds (i.e., large values for $\{\bar{q}_{rs}\}$) mean that the excess flow is high and the excess-demand links are always "congested". As shown in Chapter 5, the convex combinations algorithm is slow to converge for congested networks. It is therefore a good practice to set $\{\bar{q}_{rs}\}$ as low as possible. A good bound can be obtained from the minimum travel-time path connecting each O–D pair at free-flow conditions. In other words, for each O–D pair r–s, $\bar{q}_{rs}$ is set equal to $D_{rs}(u_{rs}^0)$, where u_{rs}^0 is the travel time on the minimum path connecting r to s when $t_a = t_a(0)$, $\forall\ a$. Obviously, the final O–D flow will never be larger (due to network congestion) and thus this bound will not be binding at the solution point. Such a bound, however, still may not be very tight and better convergence characteristics can be achieved by varying the bounds at each iteration. In other words, the bounds at each iteration, $\{\bar{q}_{rs}^n\}$, can be set lower than $D_{rs}(u_{rs}^0)$. These bounds are then examined at each iteration. If $q_{rs}^n = \bar{q}_{rs}^n$, $\bar{q}_{rs}^n$ is increased at the next iteration whereas if $q_{rs} \ll \bar{q}_{rs}^n$, $\bar{q}_{rs}^n$ is decreased at the next iteration. Note that Chapter 7 describes an algorithm that can be adapted to solve variable-demand UE problems even though it was originally developed for joint distribution/assignment problems (which are explored in that chapter). As it turns out, it is more efficient to use that algorithm than the convex combinations method for solving variable demand problems.

To conclude this discussion note that one concern that is significantly more fundamental than the computational issues, is the availability of appropriate data. In assessing the viability of using a variable-demand model, a key factor is the availability of the demand functions, or of the data needed to estimate such functions. The computational issues considered in this section are dwarfed in comparison to the cost of obtaining these data. Further data and input considerations are discussed in Chapter 13.

The next section uses the methodology developed in this chapter to formulate and solve a problem that includes an alternative mode of travel in addition to the automobile.

6.4 MODE CHOICE WITH INDEPENDENT MODES

In most cases, the design period for which urban networks are analyzed is the morning or afternoon peak-demand period. During these times, most of the trips cannot be easily forgone or shifted because of the purpose for which they are conducted (e.g., a large portion of these trips are work related). The number of trips taken between each origin and destination can thus be regarded as fixed. Some of these trips, however, may not be conducted over the road network but on alternative modes of transportation, such as transit. This section analyzes a network equilibrium problem in which the network includes both automobile and transit modes. The solution includes the flow of transit patrons between each O–D pair in addition to the vehicular flow pattern over the network. This problem is referred to as the combined modal split/traffic assignment problem.

To analyze the combined problem, assume that some of the network O–D pairs are connected by transit. At this point in the text, the level of service offered by the transit system is assumed to be independent of either the automobile flow or the transit patronage. The following analysis is thus applicable only in situations where the transit vehicles are moving on dedicated guideways (e.g., subway tunnels, elevated guideways, reserved bus lanes, etc.) and where the transit capacity is large enough so that congestion effects on the transit routes can be assumed not to occur. (A more general analysis of this topic is presented in Chapter 9.) For the purposes of this discussion, the transit level of service between origin r and destination s is summarized by a constant, $\hat{u}_{rs}$, expressed in travel time units. The transit flow between r and s is denoted by $\hat{q}_{rs}$. The transit service can thus be represented by a single link $r \rightarrow s$ connecting each O–D pair served by transit.

The combined automobile and transit network is illustrated in Figure 6.7, which depicts a single O–D pair. The network structure shown in this figure is similar to the excess-demand network representation discussed in the

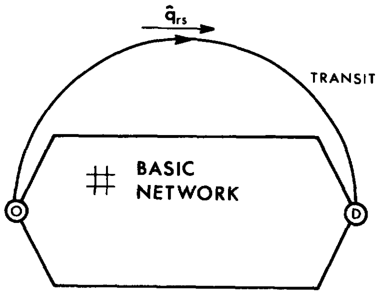


Figure 6.7 Excess-demand representation for a simple joint modal split/traffic assignment problem.

preceding section. The difference between the two representations is that in the mode-choice problem, the total O–D demand, $\bar{q}_{rs}$, is not an arbitrary upper bound but one that corresponds to the actual total trip rate (by both modes) for each O–D pair. Note that, for consistency, both automobile and transit flows should now be expressed in terms of persons per unit of time. Since the basic network is analyzed in terms of automobile flows, a vehicle occupancy factor must be used to convert person flow to vehicular flow over this network. For simplicity, it is assumed here that the automobile occupancy factor is 1. (The flow over the transit network is measured in persons per time unit since the flow of the transit vehicles is assumed exogenous to the problem.)

Equilibrium between Modes

At equilibrium, the transit and automobile flows and travel times between each O–D pair should satisfy some equilibrium definition. One possibility is to define a UE condition between the two flows in addition to the equilibrium over the basic network. In other words, for each O–D pair, the travel times on both the transit and the road network should be equal if both are used. If only one mode is used, the travel time on it should be lower than the travel time on the unused mode. If this definition is adopted, the added transit link should be treated in the same fashion as any other link in the network. Assuming that all O–D pairs are connected by transit, the appropriate equivalent minimization program would then be

$$\min z(\mathbf{x}, \hat{\mathbf{q}}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega + \sum_{rs} \hat{q}_{rs} \hat{u}_{rs} \qquad [6.25a]$$

subject to

$$\sum_k f_k^{rs} + \hat{q}_{rs} = \bar{q}_{rs} \qquad \forall\, r, s \qquad [6.25b]$$

$$f_k^{rs}, \hat{q}_{rs} \geq 0 \qquad \forall\, k, r, s \qquad [6.25c]$$

where $\hat{\mathbf{q}} = (\ldots, \hat{q}_{rs}, \ldots)$ and $\hat{u}_{rs}$ is the transit travel time between origin r and destination s.† The solution to this program can be found with a standard UE algorithm.

The fault of the analysis leading to program [6.25] lies in its failure to account for observed conditions. Transit lines are always used, even though the transit travel times tend to be longer than automobile travel times for the same O–D pairs. Even in cases in which the transit and auto travel times represent general travel impedance, the UE conditions are not a reasonable model of modal split since they imply that the same determinant of mode

†The variable $\hat{u}_{rs}$ can be assumed to include factors other than travel time, and to stand for a "generalized transit impedance" expressed in time units. As mentioned in Chapter 1, the travel times over the automobile network can be interpreted in this fashion as well.

choice (the travel impedance) can be defined for all individual travelers. In such a model travelers would consistently choose the mode with the lower impedance, in the same way that it is assumed that they choose the route with the lowest travel time. Mode choice studies, however, indicate clearly that this is not the case.

The issue of modal split between transit and automobile has been thoroughly researched and modeled by transportation engineers and planners, who have long recognized that mode choice results from a complex decision process. This process is influenced by a large number of factors, many of which are difficult to quantify and measure. To account for these factors in practice, special "mode split" functions have been developed. These functions take into account the transit and automobile travel times but allow for situations in which these times are not equal at equilibrium. Given a set of travel times, these functions approximate the appropriate flow of transit and automobile travelers for each O–D pair. One of the most widely used mode split functions is the logit formula:

$$q_{rs} = \bar{q}_{rs} \frac{1}{1 + e^{\theta(u_{rs} - \hat{u}_{rs})}} \qquad [6.26]$$

where θ is a positive parameter (estimated from data), u_{rs} is the (minimum) travel time by auto between O–D pair r–s, and $\hat{u}_{rs}$ is the transit travel time for this O–D pair. Note that this functional form also ensures that both q_{rs} and $\hat{q}_{rs}$ are positive and less than $\bar{q}_{rs}$. At equilibrium, then, the automobile and transit trip rates should satisfy Eq. [6.26]. In addition, the UE conditions have to be satisfied over the basic (automobile) network.

Formulation and Solution Algorithm

The logit model is discussed in detail in Chapter 10. At this point it should be regarded simply as a formula for the demand for *vehicular* trips between origin r and destination s. Because the transit travel time, $\hat{u}_{rs}$, is fixed, Eq. [6.26] gives q_{rs} as a function of u_{rs}. Furthermore, this function is decreasing in u_{rs} and bounded from above (by $\bar{q}_{rs}$). Consequently, the logit model (in Eq. [6.26]) satisfies the requirements of a demand function, as introduced in Section 6.1. The equilibrium over the road network can now be viewed as a variable demand UE problem with the demand functions, $q_{rs} = D_{rs}(u_{rs})$, given by Eq. [6.26] for each O–D pair. Furthermore, the problem can be solved with the variable-demand methods introduced in the preceding section. As mentioned at the beginning of this section, the transit link that connects each O–D pair parallels the excess-demand link introduced in Section 6.3. In order to apply that solution methodology, however, the demand model has to be inverted and expressed in terms of the excess-demand variable (this variable, denoted by e_{rs} in the preceding section, represents in this case the transit ridership, denoted here by $\hat{q}_{rs}$).

The inverse demand function, $D_{rs}^{-1}(\cdot)$, is given by

$$u_{rs}(q_{rs}) = \frac{1}{\theta} \ln \left(\frac{\bar{q}_{rs}}{q_{rs}} - 1 \right) + \hat{u}_{rs} \qquad \forall\ r, s \qquad [6.27a]$$

The excess demand function [denoted $W(\cdot)$ in the preceding section] can be derived by changing the argument in the last function to $\hat{q}_{rs} = \bar{q}_{rs} - q_{rs}$, that is,

$$W_{rs}(\hat{q}_{rs}) = \frac{1}{\theta} \ln \frac{\hat{q}_{rs}}{\bar{q}_{rs} - \hat{q}_{rs}} + \hat{u}_{rs} \qquad \forall\ r, s \qquad [6.27b]$$

The argument of this function, $\hat{q}_{rs}$, is the excess vehicular demand, which in this case represents the flow of transit trips between r and s. The formulation of the equivalent program can now be expressed as

$$\min z(\mathbf{x}, \hat{\mathbf{q}}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega + \sum_{rs} \int_0^{\hat{q}_{rs}} \left(\frac{1}{\theta} \ln \frac{\omega}{\bar{q}_{rs} - \omega} + \hat{u}_{rs} \right) d\omega \qquad [6.28a]$$

subject to

$$\sum_k f_k^{rs} + \hat{q}_{rs} = \bar{q}_{rs} \qquad \forall\ r, s \qquad [6.28b]$$

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [6.28c]$$

$$0 < \hat{q}_{rs} < \bar{q}_{rs} \qquad \forall\ r, s \qquad [6.28d]$$

The reader should, at this point, be able to verify that the Kuhn–Tucker conditions of program [6.28] guarantee that its solution is a two-mode equilibrium flow pattern (see Problem 6.10). Note that due to the logarithm in the objective function, the transit O–D flows can attain neither their upper nor their lower limit (i.e., $0 < \hat{q}_{rs} < \bar{q}_{rs}$ as stated in constraint [6.28d]). In fact, the aforementioned functional form of the objective function implies that this constraint will never be binding and therefore it does not have to be included in the first-order conditions of this program.

The application of the standard (fixed-demand) convex combinations method to this problem is straightforward. The transit alternative is represented by a link directly connecting each O–D pair served by transit. The performance curve for this link, $W_{rs}(\hat{q}_{rs})$, is given by Eq. [6.27b], which is plotted in Figure 6.8 for certain values of its parameters (i.e., $\theta = 1.0$, $\bar{q}_{rs} = 1.0$, and $\hat{u}_{rs} = 1.0$). It is important to note that this function is increasing. A detailed investigation of the properties of this function is left to the reader (see Problem 6.14).

This representation clarifies the reason constraint [6.28d] is not binding at the optimal solution. If $\hat{q}_{rs} \rightarrow 0$, the equivalent performance function over the transit link, $W_{rs}(\hat{q}_{rs})$, will tend to $-\infty$. More flow will then tend to use the transit link, so $\hat{q}_{rs}$ will increase. If $\hat{q}_{rs} \rightarrow \bar{q}_{rs}$, $W_{rs}(\hat{q}_{rs})$ will tend to ∞ and the flow over the transit link will decrease. Consequently, neither of these two boundary values can be in the equilibrium solution of the problem.

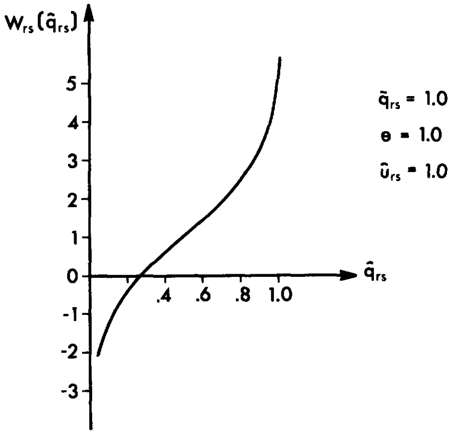


Figure 6.8 The equivalent link performance function used for the additional transit link to represent the logit modal split function. It parallels the argument-completing function associated with the inverse demand function (see Figure 6.4 and the related discussion).

The point where caution should be exercised is in initializing the algorithm. It is customary to start the procedure with an empty (zero-flow) network. The equivalent performance function over the dummy link, however, is undefined for $\hat{q}_{rs} = 0$ (or $q_{rs} = \bar{q}_{rs}$), and thus other values should be used. A good starting point can be obtained by the values of $\hat{q}_{rs}$ and q_{rs} resulting from an application of the logit modal split function to the travel time over an empty basic network. Once an initial feasible solution is found, the algorithm will not generate another solution with either $\hat{q}_{rs} = 0$ or $\hat{q}_{rs} = \bar{q}_{rs}$.

At the point of convergence of the algorithm, equilibrium is obtained and the travel time through the basic network should equal the "equivalent travel time" $W_{rs}(\hat{q}_{rs})$ on the parallel (transit) route for each O–D pair. In other words, the following condition must hold:†

$$\frac{1}{\theta} \ln \frac{\hat{q}_{rs}}{\bar{q}_{rs} - \hat{q}_{rs}} + \hat{u}_{rs} = u_{rs} \qquad \forall\ r, s \qquad [6.29a]$$

This equation, however, can be rewritten as

$$\frac{\hat{q}_{rs}}{\bar{q}_{rs}} = \frac{1}{1 + e^{\theta(\hat{u}_{rs} - u_{rs})}} \qquad \forall\ r, s \qquad [6.29b]$$

which is the logit formula for the transit volume (compare this expression to Eq. [6.26]). Thus, at equilibrium, the modal split is given by the logit model and the UE conditions are satisfied over the basic network.

Example

Consider an elementary network consisting of one link connecting an origin to a destination. In addition, the origin is connected to the destination by a transit route (link) as shown in Figure 6.9. Let $\hat{t}$ and $\hat{x}$ denote the travel

†Again, the asterisks denoting equilibrium values have been omitted from the variables ($\hat{q}_{rs}$, and u_{rs}) in this equation for clarity of notation.

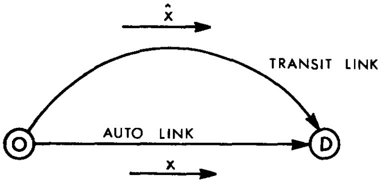


Figure 6.9 Simple network including one O–D pair connected by one automobile route and one transit line.

time and the flow, respectively, over the transit link. Similarly, let t and x denote the travel time and flow, respectively, over the automobile link, where $t = t(x)$. Denoting the total O–D volume by $\bar{q}$, the flows of automobile users and transit passengers are given by the logit formula with parameter θ, that is,

$$\hat{x} = \bar{q}\,\frac{1}{1 + e^{\theta(\hat{t}-t)}}$$

$$x = \bar{q} - \hat{x}$$

Assume for this example that the road performance function is given by $t = 10 + 0.2x^2$, the transit travel time is $\hat{t} = 20$ time units, the total O–D flow is $\bar{q} = 10$ person units/time unit, and the logit parameter is $\theta = 0.1$ time unit^{-1}.

The equilibrium equations in this case would be:

$$\text{Mode split:} \qquad \hat{x} = \frac{10}{1 + \exp\,[0.1(20 - t)]} \qquad [6.30a]$$

$$\text{Road performance:} \quad t = 10 + 0.2x^2 \qquad [6.30b]$$

$$\text{Flow conservation:} \quad x = 10 - \hat{x} \qquad [6.30c]$$

The flow conservation condition can be substituted in the road performance equation and the resulting expression for t (in terms of $\hat{x}$) can then be substituted into the mode split equation. This process yields the following equation for $\hat{x}$:

$$\hat{x} = \frac{10}{1 + \exp\,[0.1(-0.2\hat{x}^2 + 4\hat{x} - 10)]} \qquad [6.31]$$

The solution of this equation can be found numerically (using a programmable calculator) to be $\hat{x} = 4.2$ flow units.† This transit flow implies an automobile flow of $x = 5.8$ flow units.

To solve this example as an equivalent fixed-demand problem, the problem should be set as a two-link network with the performance function on the one link being

$$t(x) = 10 + 0.2x^2 \qquad [6.32a]$$

and the performance function on the other link being (see Eq. [6.27b])

$$W(\hat{x}) = \frac{1}{\theta} \ln \frac{\hat{x}}{10 - \hat{x}} + 20 \qquad [6.32b]$$

†More accurately, the solution is $\hat{x} = 4.193113$ flow units.

Let $\hat{t}' = W(\hat{x})$ be the "equivalent travel time" on the transit link. Using this approach, the modified network [which includes the "equivalent performance function" $W(\hat{x})$] is solved for *user equilibrium* (i.e. at equilibruium $t = \hat{t}'$ even though $t \neq \hat{t}$). Assuming that both links will be used at equilibrium, the solution can be found by setting $t = \hat{t}'$, or $t(x) = W(\hat{x})$, that is,

$$10 + 0.2x^2 = \frac{1}{\theta} \ln \frac{\hat{x}}{10 - \hat{x}} + 20$$

To solve this problem, the flow conservation constraint, $\hat{x} = 10 - x_1$ can be substituted for the transit flow on the right-hand side of this equation to obtain the equation

$$10 + 0.2x^2 = \frac{1}{\theta} \ln \frac{10 - x}{x} + 20$$

Using $\theta = 0.1$, the last equation can be written as

$$0.2x^2 - 20 - 10 \ln \frac{10 - x}{x} = 0 \qquad [6.33]$$

Equation [6.33] can be solved numerically to obtain $x = 5.8$, implying $\hat{x} = 4.2$, a solution that is identical to the flows obtained directly from the equilibrium conditions. Thus the equivalent minimization gives the same results as the direct solution, as expected.

At equilibrium, the travel time on the road link is $t(5.8) = 16.7$, and the equivalent travel time on the transit link is $\hat{t}' = W(4.2) = 16.7$ time units as well. The actual travel time on the transit link, however, is $t = 20$ time units. At equilibrium, then, the travel times on both modes are not equal even though both are used. Instead, the flow is given by a special mode split function, the argument of which is the equilibrium travel time.

This example can also be solved by applying the convex combinations method to the two-link representation, which includes performance functions [6.32]. For this small problem the algorithm would converge in one iteration. (Why?) Starting with $x^0 = 5$ and $\hat{x}^0 = 5$, the auxiliary solution would be $y^0 = 10$ and $y^0 = 0$. The move size would be $\alpha_0 = 0.16$† (show it), resulting in the equilibrium solution $x = 5.8$ and $\hat{x} = 4.2$ flow units.

6.5 SUMMARY

This chapter introduced the problem of user equilibrium with variable demand. The origin–destination trip rates which are an input to the original UE problem are replaced here by O–D demand functions. These functions give the trip rate between each O–D pair as a function of the travel time between the origin and the destination. The demand functions are assumed to be decreasing and bounded from above.

†More precisely, $\alpha_0 = 0.161377$.

Problems of equilibrium with variable demand can be formulated as equivalent minimization programs in a fashion analogous to the formulation of fixed-demand problems. The important difference between the formulations of the two problems is that the O–D trip rates are treated as variables (as opposed to exogeneous constants) in the variable-demand formulation. The first-order conditions for the equivalent program are identical to the equilibrium equations, and the program has a unique solution in terms of the link flows and the O–D trip rates.

The convex combinations algorithm can readily be adapted to the solution of the variable-demand equivalent program. The adaptation requires one additional check in the direction finding stage and, of course, the use of the variable-demand objective function in the move-size-determination stage. These modifications to the algorithm, however, are unnecessary because the network representation of the problem can be augmented to generate an equivalent fixed-demand UE problem to which any UE algorithm can be applied. Section 6.3 described two different network representations that can be used to develop such an equivalent UE program. These are the zero-cost overflow formulation and the excess-demand formulation. In each case the modified network includes some additional "dummy" link with "equivalent performance functions." The excess-demand representation is the more efficient of the two since it requires less computer memory.

The approach used in the formulation and solution of the excess-demand network representation can also be used to solve problems combining modal split and traffic assignment. Section 6.4 demonstrates how the excess-demand link can represent the transit flow between each O–D pair and how the "equivalent performance function" for this link is obtained. The formulation of the joint equilibrium/mode choice problem is based on the logit mode split formula, due to its widespread use in practice for this purpose. Note that even though the UE conditions hold throughout the modified network at the solution point, the relative volumes of transit and auto are given by the logit formula. This, of course, is due to the particular formulation of the "equivalent performance function" over the transit link connecting each O–D pair.

The problem of determining, jointly, the equilibrium flows and the modal splits, as formulated and solved in Section 6.4, is quite restrictive. It applies only to noninteracting modes and in cases in which the transit travel time is not flow dependent. Chapter 9 offers a more general treatment of the joint traffic assignment/mode split problem.

6.6 ANNOTATED REFERENCES

The formulation of the problem of user equilibrium with elastic demand, as an equivalent minimization program, appears in the work of Beckmann et al. (1956). The original formulation in that book includes both the variable-demand case and the fixed-demand case. The fixed-demand case, however, was not treated separately as is done in this text.

One of the first approaches to the solution of this problem was offered by

Beckmann et al. in the aforementioned reference. Their heuristic is a hybrid between capacity restraint and incremental assignment method; it determines the minimum paths based on the travel times of the last iteration but assigns only a predetermined portion of the flow to this minimum path, combining the old and new flow solutions in fixed proportions. This technique is similar to the modified capacity restraint method discussed in Chapter 5. The method of incremental assignment was investigated by Martin and Manheim (1965) and implemented in the DODOTRANS urban planning package (which included variable demand) developed by Manheim and Ruiter (1970). Another heuristic was suggested by Bruynooghe et al. (1968) based on computation and adjustment of shortest and longest O–D routes. Wigan (1971) used a capacity restraint method for the determination of O–D flows, while the network itself was solved using the convex combinations method. Rigorous minimization-based methods were suggested by many researchers, including Wilkie and Stefanek (1971), who proposed a Newton–Raphson procedure similar to the one mentioned in Chapter 4 of this book. This procedure does not, however, exploit the special network structure of the problem, and consequently is impractical. A more efficient algorithm was suggested by Florian (1977) based on Benders's decomposition method. Unfortunately, the traffic assignment part of their procedure is based on the reduced gradient method, which requires path enumeration; this requirement makes it infeasible for large networks.

The presentation in this chapter follows the exposition of the use of dummy network representation and fixed-demand algorithms given by Gartner (1980). This exposition is based on the works of Murchland (1970), who first suggested the use of dummy links associated with inverse-demand functions, and Dantzig et al. (1976), who showed how fixed-demand algorithms can be applied to such representations. The comments regarding numerical consideration with respect to the variable demand problem are based on the works of LeBlanc and Farhangian (1981), and Mimis (1984).

The idea of representing many travel decisions as choices between paths in an abstract network was suggested by Dafermos (1976), who assumed user-equilibrium conditions to prevail throughout the abstract network. One of the first treatments of the joint modal split and traffic assignment problems in the context of mathematical programming-based equilibrium methods is offered by Florian (1977). The literature on mode choice models is too vast to enumerate here. Many of the approaches to that issue are covered in the books by Stopher and Meyburg (1975), and Kanafani (1983). The logit model used in this section is described in greater detail in Chapter 10, which includes more references.

6.7 PROBLEMS

6.1. Show that the Hessian of the objective function of the equivalent variable-demand program [6.1] is positive definite. Why is the problem unique in the O–D trip rates but not in path flows?

6.2. Develop a detailed flowchart of the application of the convex combinations algorithm to the minimization of the equivalent program for user equilibrium with variable demand (Program [6.10]). Include in your flowchart a detailed description of the initialization procedure and the convergence test.

6.3. Solve *graphically* for the equilibrium flows in a network, including two links connecting one O–D pair. The data for this problem are as follows:

$$\begin{aligned}&\text{Link 1 performance:} && t_1 = 1 + \tfrac{1}{2}x_1^2\\ &\text{Link 2 performance:} && t_2 = 3 + x_2\\ &\text{Demand} && q = \frac{5}{u} - 1\\ &\text{Flow conservation:} && q = x_1 + x_2\end{aligned}$$

where t_1 and t_2 are the travel times on link 1 and link 2, respectively, x_1 and x_2 are the flows on these links, u is the O–D travel time, and q is the O–D flow.

6.4. Formulate Problem 6.3 as an equivalent minimization. State the first-order conditions and solve them for the equilibrium flows.

6.5. Formulate Problem 6.3 as a fixed-demand problem and solve it graphically as such.

6.6. Consider the zero-cost overflow formulation.

(a) Show that the mathematical programming formulation in Eqs. [6.22] reduces to the original formulation of the variable-demand program in Eqs. [6.10].

(b) At equilibrium, what is the travel time on link $r \rightarrow r'$? What is the implication of this on the equilibrium conditions?

(c) Can you think about situations in which the path $r \rightarrow r'$ is not used at equilibrium? Explain.

(d) What is the basis for the name "generating link" for link $s \rightarrow r'$?

6.7. Solve the simple one-link example of Section 6.1 using $t(x) = 1 + x^2$ and $D(t) = 10 - 2t$. Use the excess-demand formulation.

6.8. Solve for the equilibrium flow over a one-link network with

$$t = 1 + x$$

$$x = \begin{cases} 3 & \text{for } t \leq 2\\ 5 - t & \text{for } t \geq 2\end{cases}$$

using the excess-demand formulation.

6.9. Suggest two ways to modify programs [6.25] and [6.28] to account for O–D pairs that are not served by transit.

6.10. Show that the first-order conditions of the equivalent variable-demand formulation (program [6.28]) give the proper equilibrium/mode split conditions.

6.11. Solve the integral in Eq. [6.28a] and formulate the equivalent minimization for the two-modes problem explicitly in terms of **x**, **q**, and $\hat{q}$. Formulate the Lagrangian of this program and investigate its first-order conditions. (Recall that $\int_0^x \ln \omega \, d\omega = x \ln x - x$.)

6.12. Formulate the problem of equilibrium and mode split when the transit travel time is flow dependent. Is there any difficulty in formulating the network representation of the problem?

6.13. Formulate the problem of equilibrium with mode split when the logit model is given by $\hat{q}_{rs} = \bar{q}_{rs}/[1 + e^{\theta_1 + \theta_2(\hat{u}_{rs} - u_{rs})}]$. (As in Section 6.4, assume independence of the transit mode and no congestion on that mode.) Also, suggest an interpretation of the parameters of this logit function and compare it to the one used in Section 6.4.

6.14. Investigate the inverse demand function, $W_{rs}(\hat{q}_{rs})$, used in the excess-demand formulation of the mode split problem (Eq. [6.27b]).

(a) Show graphically the effect of θ, $\bar{q}_{rs}$, and $\hat{u}_{rs}$. Where is the inflection point, and where is the root of the function?

(b) Over some of its range, $W_{rs}(\cdot) < 0$. This is not typical of other performance curves in the network. Is this a problem? Why?

(c) Describe qualitatively how the sequence of assignments will progress as the convex combinations algorithm is applied to the solution of the combined mode split/traffic assignment problem.

6.15. Describe how the convex combinations algorithm should be modified to solve the joint mode split/traffic assignment problem if the network representation formulation is not used.

6.16. Formulate the example in Section 6.4 as a minimization problem using the excess-demand formulation. Solve it and check that your solution coincides with the answer given in that section.

6.17. Solve the one link–two modes example in Section 6.4 for other values of the logit parameter (use $\theta = 0, 0.3, 0.5, 1.0, 2.0$). Compare the solution in each case with the UE-type solution (which assumes that if both modes are used at equilibrium, the transit and automobile travel times will be the same). What is the effect of θ? How would you use field data for model selection in joint modal split/traffic assignment studies?

7

Trip Distribution and Traffic Assignment Models

The standard UE network assignment problem can be viewed as a model of motorists' choice of path from origin to destination. Chapter 6 considered an additional choice dimension. This dimension can be looked upon as either the choice of not to travel or the choice of an alternative travel mode. The former results in the variable-demand equilibrium assignment problem, while the latter gives rise to a joint mode split/equilibrium assignment problem. In each of these problems, the O–D trips rates are determined independently for each O–D pair.

This chapter introduces a new dimension of choice and a somewhat different problem. Here (unlike in Section 6.4) the focus is on vehicular trips only, but the total number of trips originating from each node is assumed fixed. The question then becomes one of how these trips are distributed among the various destination nodes, an issue that is affected by motorists' choice of destination. The determination of motorists' choice of destination and the path used to get there is known as the joint distribution/assignment problem. In this case, the O–D flows, $\{q_{rs}\}$, are variables, the value of which have to be determined subject to the constraint

$$\sum_{s} q_{rs} = O_r \qquad \forall\, r \tag{7.1}$$

where O_r is the (fixed and known) total number of trips originating at node r.

In this problem, it is assumed that each destination, s, is associated with some attraction measure, M_s, reflecting the activity opportunities available there. Accordingly, it can include variables such as retail activity, employment density, and so on. Several destination choice criteria can be defined in conjunction with this measure. For example, motorists might be assumed to choose the destination with the highest M_s value. This would mean, however, that travel time is not a consideration in their choice of destination, implying that the problem can be separated from the equilibrium assignment issue. Such a criterion would be a degenerate case of the more general problem discussed in this text.† A more plausible assumption would be that motorists try to accomplish two goals: travel to the destination with the highest attraction measure while spending the least possible time in travel. Under this assumption the choice of destination is the result of a trade-off between attraction and travel time. Such a model can be formulated in terms of an overall UE-type set of equilibrium conditions. This model, which is presented in Section 7.1, suffers from several shortcomings; it is used in this chapter only to illustrate some concepts that are important for other models. The practice in the transportation planning profession is to use special functions (known as trip distribution or destination choice models) to distribute the trips. Section 7.2 presents the formulation and solution of the resulting distribution/assignment problem.

Section 7.3 addresses yet another trip distribution criterion. In this case, not only is the total number of trips originating at each node fixed, but the number of trips destined for any node is fixed as well. Thus, in addition to Eq. [7.1], the O–D flow rates have to satisfy the constraint

$$\sum_r q_{rs} = D_s \quad \forall\, s \qquad [7.2]$$

where D_s is the (fixed and known) number of trips destined for node s.

Each of the models discussed in this chapter is associated with an equivalent minimization program and algorithmic solution procedures. The last model discussed, however, requires the solution of a special subproblem known as *Hitchcock's transportation problem.* An algorithmic solution to this problem is outlined in Section 7.4.

7.1 THE UE DISTRIBUTION/ASSIGNMENT PROBLEM

The problem analyzed in this section assumes that each destination, s, is associated with an attraction measure, M_s. The total flow originating at each node, r, is fixed to be O_r. Travelers are assumed to choose destinations that are attractive (high M_s value), on the one hand, and close by (low u_{rs} value), on the other. (As before, u_{rs} is the travel time on the minimum path from r to s.) In this model destinations are chosen so that the difference, $(M_s - u_{rs})$, is maximized. Alternatively, the destination choice criterion can be stated as a minimization of the *net travel impedance*, $(u_{rs} - M_s)$. In other words, all motorists at r will choose the destination with the lowest net travel impedance. At

†It means, for example, that a person is equally likely to buy groceries at a local store as he or she is to drive across town to purchase the same items at a similar store.

equilibrium, the net travel impedance to all destinations visited from r (that is, destinations r–s for which $q_{rs} > 0$) will be equal to or lower than the net travel impedance to all destinations not visited from r. Notwithstanding this condition, the regular UE conditions should hold among all paths connecting each O–D pair. This formulation is known as the UE (or Wardropian) distribution/assignment model. It is important to note that the attraction measures, $\{M_s\}$, have to be specified in travel time units in order to be compatible with the network travel times.†

Problem Formulation

Based on the formulation of the (fixed-demand) equivalent user-equilibrium program [3.1], the combined distribution/assignment problem can be formulated as follows:

$$\min z(\mathbf{x}, \mathbf{q}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega - \sum_{rs} M_s q_{rs} \qquad [7.3a]$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\, r, s \quad (u_{rs}) \qquad [7.3b]$$

$$\sum_s q_{rs} = O_r \qquad \forall\, r \quad (\mu_r) \qquad [7.3c]$$

$$f_k^{rs} \geq 0 \qquad \forall\, k, r, s \qquad [7.3d]$$

$$q_{rs} \geq 0 \qquad \forall\, r, s \qquad [7.3e]$$

In this program, O_r is a constant for each origin in the network and the dual variables (Lagrange multipliers) are given in parentheses next to the corresponding constraints. The incidence relationships have been omitted from the formulation to simplify the presentation, but it is always understood that when the formulation includes both path- and link-flow variables, these relationships hold.

For program [7.3] to be an equivalent distribution/assignment minimization, its first-order conditions must be identical to the equilibrium equations. As in previous cases, these conditions can be derived by forming and analyzing the Lagrangian, which, for this program, is given by

$$L(\mathbf{x}, \mathbf{q}, \mathbf{u}, \boldsymbol{\mu}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega - \sum_{rs} M_s q_{rs} + \sum_{rs} u_{rs}\left(q_{rs} - \sum_{rs} f_k^{rs}\right) + \sum_r \mu_r\left(O_r - \sum_s q_{rs}\right) \qquad [7.4a]$$

†The conversion of attraction and other factors affecting travel demand into travel-time units is discussed in Chapter 13. The conversion factor is determined typically in the process of estimating the attraction measures, or in general, the travel demand models discussed in that chapter.

The minimum of this Lagrangian with respect to the flow variables has to be subject to the following constraints:

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [7.4b]$$

$$q_{rs} \geq 0 \qquad \forall\ r, s \qquad [7.4c]$$

Its maximum, with respect to **u** and **μ**, is unconstrained. The first-order conditions for a saddle point of this Lagrangian program are given by

$$\frac{\partial L(\cdot)}{\partial f_k^{rs}} \geq 0, \quad \frac{\partial L(\cdot)}{\partial f_k^{rs}} f_k^{rs} = 0 \qquad \forall\ k, r, s$$

$$\frac{\partial L(\cdot)}{\partial q_{rs}} \geq 0, \quad \frac{\partial L(\cdot)}{\partial q_{rs}} q_{rs} = 0 \qquad \forall\ r, s$$

$$\frac{\partial L(\cdot)}{\partial u_{rs}} = 0 \quad \forall\ r, s$$

$$\frac{\partial L(\cdot)}{\partial \mu_r} = 0 \quad \forall\ r$$

and the nonnegativity constraints [7.4b] and [7.4c]. The derivatives of the Lagrangian with respect to the flow variables are as follows:

$$\frac{\partial L(\cdot)}{\partial f_k^{rs}} = c_k^{rs} - u_{rs} \qquad \text{and} \qquad \frac{\partial L(\cdot)}{\partial q_{rs}} = -M_s + u_{rs} - \mu_r$$

The derivatives with respect to the dual variables are the associated constraints themselves. The first-order conditions of program [7.3] are then

$$(c_k^{rs} - u_{rs}) f_k^{rs} = 0 \qquad \forall\ k, r, s \qquad [7.5a]$$

$$c_k^{rs} - u_{rs} \geq 0 \qquad \forall\ k, r, s \qquad [7.5b]$$

$$[(u_{rs} - M_s) - \mu_r] q_{rs} = 0 \qquad \forall\ r, s \qquad [7.5c]$$

$$(u_{rs} - M_s) - \mu_r \geq 0 \qquad \forall\ r, s \qquad [7.5d]$$

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\ r, s \qquad [7.5e]$$

$$\sum_s q_{rs} = O_r \qquad \forall\ r \qquad [7.5f]$$

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [7.5g]$$

The first two conditions spell out the familiar user-equilibrium condition over the network. Equation [7.5b] means that, at the saddle point of the Lagrangian, u_{rs} can be interpreted as the travel time over the shortest path connecting O–D pair r–s, whereas Eq. [7.5a] guarantees that if the flow on any given path, k, is positive, the travel time on this path will equal the travel time on the shortest path. Consequently, the travel time on all paths with positive flow between each O–D pair will be equal. Furthermore, this travel time will be no greater than the travel time on any unused paths.

Equations [7.5c] and [7.5d] exhibit the same structure and can be interpreted in a similar fashion. Equation [7.5d] means that μ_r is the smallest net travel impedance, $(u_{rs} - M_s)$, of all destinations that can be visited from origin r. Equation [7.5c] guarantees that if destination s is actually visited from r (i.e., $q_{rs} > 0$), the net travel impedance to this destination will equal μ_r. Consequently, the net travel impedance to all destinations visited from each origin will be equal. Furthermore, this net travel impedance is less than or equal to the net travel impedance to any destination not visited from that origin.

It can be concluded, then, that program [7.3] is an equivalent minimization program to the UE distribution/assignment problem. This type of equilibrium includes the usual UE conditions in terms of the paths' travel times for each O–D pair, and similar UE conditions in terms of the net travel impedance to all destinations, for each origin. The reader can verify that program [7.3] is strictly convex and has, therefore, a unique minimum.

Algorithm

The solution for program [7.3] can be obtained by using the convex combinations method. The linear program which has to be solved in the direction finding step of this algorithm can be formulated by first writing program [7.3] in terms of path-flow variables as done in Section 6.2 for the variable-demand problem. The linear program (at the nth iteration of the algorithm) then becomes

$$\min z_n(\mathbf{g}) = \sum_{rs} \sum_{k} \frac{\partial z[\mathbf{x}(\mathbf{f}^n), \mathbf{q}(\mathbf{f}^n)]}{\partial f_k^{rs}} \, g_{rs} = \sum_{rs} \sum_{k} (c_k^{rs^n} - M_s) g_k^{rs} \qquad [7.6a]$$

subject to

$$\sum_{s} \left(\sum_{k} g_k^{rs} \right) = O_r \qquad \forall\ r \qquad [7.6b]$$

$$g_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [7.6c]$$

where $c_k^{rs^n}$ is the travel time on path k between r and s at the nth iteration (i.e., $c_k^{rs^n} = \sum_a t_a^n \delta_{a,k}^{rs}$). Since the flow conservation constraint [7.6b] holds for each origin, the problem can be decomposed by origin nodes. For each r, then, the objective function is minimized by assigning all the available flow (O_r) to the route, l, leading from r to destination m for which the net travel impedance is the smallest (i.e., l, m such that $c_l^{rm^n} - M_m = \min_{k,s} \{c_k^{rs^n} - M_s\}$). All other routes are assigned zero flow. The direction-finding step at the nth iteration can, then, be summarized as follows. For each origin, r:

1. Compute the minimum path to all destinations based on the current set of link times, $\{t_a^n\}$. Denote the travel times on these paths by $u_{rs}^n, \forall\ s$.

2. Assign flow O_r to the minimum path of the "most attractive destination" in terms of net travel impedance (i.e., to m such that $u_{rm}^n - M_m = \min_s \{u_{rs}^n - M_s\}$).

This yields (after the flow from all origins has been assigned) an auxiliary flow pattern, $\{g_k^{rsn}\}$, which can be expressed in terms of auxiliary link and O–D flows, $(\{y_a^n\}, \{v_{rs}^n\})$ by using the relationships $y_a^n = \sum_k g_k^{rsn} \delta_{a,k}^{rs}$ and $v_{rs}^n = \sum_k g_k^{rsn}$.

The move size is determined by finding a scalar, α_n, that solves the program

$$\min_{0 \leq \alpha \leq 1} z(\alpha) = \sum_a \int_0^{x_a^n + \alpha(y_a^n - x_a^n)} t_a(\omega)\, d\omega - \sum_{rs} M_s[q_{rs}^n + \alpha(v_{rs}^n - q_{rs}^n)] \qquad [7.7]$$

Once α_n is found, the new solution can be obtained by the updating step given in Eqs. [6.19].

Solution by Network Representation

The adaptation of the convex combinations algorithm to the solution of the distribution/assignment program is straightforward. As was the case with the variable-demand problem, however, such an adaptation may not be necessary. A simple modification of the network representation can transform the distribution/assignment problem into an equivalent UE problem, which can be solved with any standard UE equilibration algorithm.

The required modification is shown in Figure 7.1; the network is augmented by dummy links leading from each destination node to a dummy destination node, denoted r'. There is one dummy destination node associated with each origin node. Consequently, the total number of dummy links is the product of the number of origins and the number of destination. Figure 7.2 illustrates the augmented network for a case of two origins and three destinations. The flow on each of the dummy links, $s \rightarrow r'$, is q_{rs}, and the equivalent travel time on each of these dummy links is $-M_s$. Note that the actual network links are not shown in Figures 7.1 and 7.2; they are all part of the basic network, which is depicted only abstractly.

Consider now a UE problem defined over the modified network. The origin nodes of this network are identical to the origin nodes of the original

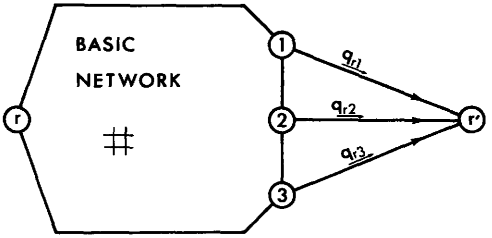


Figure 7.1 Network representation for solving a joint distribution/assignment problem; dummy links are added from each destination node to a new "origin-dummy" node.

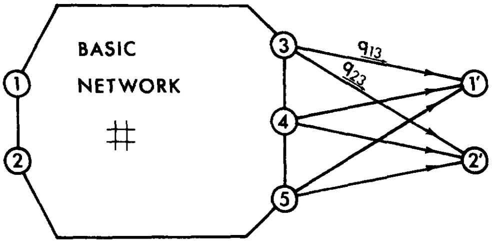


Figure 7.2 Network representation for the joint distribution/assignment problem, showing two origins.

(basic) network. The destination nodes, however, are the dummy destinations, $\{r'\}$. Assume now that the fixed O–D trip rates between each origin, r, and its associated dummy destination, r', is equal to O_r (all other trip rates are zero). The equivalent minimization for this UE program is identical to program [7.3]. To verify this, note that in setting up the UE objective function for the augmented network,

$$\sum_{rs} \int_0^{q_{rs}} (-M_s)\, d\omega = -\sum_{rs} M_s q_{rs} \qquad [7.8]$$

and [7.3a] follows immediately. Constraint [7.3b] is satisfied by the structure of the modified network and constraint [7.3c] is the flow conservation constraint for this network (which guarantees that the trip rates are assigned to the network).

The UE problem over the modified network can be solved by using any UE algorithm, such as the convex combinations method. The algorithm would essentially follow the same steps described in the preceding section. For example, at every direction-finding iteration, the algorithm assigns O_r flow units to the minimum "equivalent" travel-time path between r and r'. This path is, of course, identical to the path chosen with the aforementioned adaptation of this algorithm to the distribution/assignment model, and the sequence of link flows generated will thus be identical.

Note that in the network representation of the distribution/assignment problem, the O–D flows are link variables (the flow on links $s \rightarrow r'$). Since the UE program has a unique solution in terms of link flows, the solution of the distribution/assignment problem will be unique in terms of O–D flows as well as link flows. The uniqueness of q_{rs} can thus be established on the basis of the structure of the modified network.

The joint UE distribution/assignment discussed thus far assumes that motorists' choice of destination can be modeled by postulating that these individuals choose the destination with the minimum net impedance. This assumption implies that the attraction measure for a given destination is identical across all motorists. In practice, however, it is not clear that such measures can be calculated or estimated. Consequently, there may be no values $\{M_s\}$ for which the conditions outlined in Eqs. [7.5] hold. The value of this model then is mainly pedagogical and illustrative.

7.2 DISTRIBUTION/ASSIGNMENT WITH DESTINATION DEMAND FUNCTIONS

The limitations of the UE-based distribution model presented in Section 7.1 have led transportation planning professionals to use trip distribution functions. These functions determine the number of trips that should be assigned to each destination from each origin, given the respective travel times. Thus, at equilibrium,

$$q_{rs} = D_{rs}(u_{rs}) \qquad \forall\ r, s \tag{7.9}$$

where $D_{rs}(\cdot)$ is the demand function for O–D pair r–s, as in the variable-demand problem discussed in Chapter 6. (In addition, of course, the UE conditions have to be satisfied among all paths connecting each O–D pair.) The difference between the variable-demand problem and the problem of distribution/assignment with destination demand functions is that the latter includes the "trip production" constraint,

$$\sum_s q_{rs} = O_r \qquad \forall\ r \tag{7.10}$$

Conditions [7.9] and [7.10] both have to be satisfied at equilibrium. This system of equations, however, may not have a solution since the number of equations exceeds the number of variables. To see this, consider the simple network depicted in Figure 7.3. In this example, the link performance functions are given by $t_1 = 1 + 2x_1$ and $t_2 = 2 + x_2$. The destination demand functions are given by $x_1 = 4 - t_1$ and $x_2 = 4 - t_2$, and the trip production constraint is $x_1 + x_2 = 3$. These equations have to be solved simultaneously, but no solution exists. For example, the solution of the performance and destination demand functions is $x_1^* = 1$, $x_2^* = 1$. This solution, however, does not satisfy the trip production constraints.

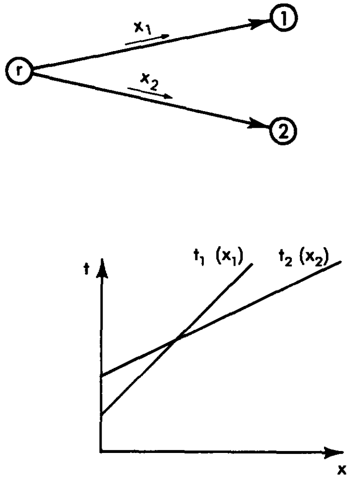


Figure 7.3 Network example with two O–D pairs and two links. The bottom figure shows the link performance functions.

As evident from Eqs. [7.9] and [7.10], the number of equations corresponding to each origin exceeds the number of O–D flow variables by one. This difficulty can be overcome by using destination demand functions which satisfy the trip production constraint. In other words, if the demand function is such that

$$\sum_s D_{rs}(u_{rs}) = O_r \qquad \forall\ r \qquad [7.11]$$

then the trip production constraint is redundant, and the demand conditions can be solved (given the O–D travel times) with no particular difficulty.

Distribution/Assignment with Logit Functions

One of the simplest ways to ensure that the trip production constraint is an integral part of the demand function is to specify this function as a share model. In a share model, the flow between origin r and destination s is given by

$$q_{rs} = O_r P_{rs}(\mathbf{u}_r) \qquad \forall\ r, s \qquad [7.12]$$

where $\mathbf{u}_r = (\ldots, u_{rs}, \ldots)$; that is, $\mathbf{u}_r$ is the vector of travel times between origin r and all destinations. The quantity $P_{rs}(\mathbf{u}_r)$ can be interpreted as the share of the total flow originating at r that destination s is attracting. P_{rs} satisfies the properties of a probability measure, that is,

$$0 \le P_{rs}(\cdot) \le 1 \qquad [7.13a]$$

$$\sum_s P_{rs}(\cdot) = 1 \qquad [7.13b]$$

Thus the trip production constraint (Eq. [7.10]) can be satisfied by using any share model in Eq. [7.9].

One of the most widely used share models is based on the logit formula introduced in Section 6.4. In the trip distribution problem, a *multinomial* logit formula is used, leading to the following destination demand model:

$$q_{rs} = O_r \frac{e^{-\gamma(u_{rs} - M_s)}}{\sum_{\forall m} e^{-\gamma(u_{rm} - M_m)}} \qquad \forall\ r, s \qquad [7.14]$$

where (the constant) M_s is the trip attraction measure associated with destination s and γ is a parameter of the logit model. It is reasonable to assume that $M_s \ge 0\ \forall\ s$ and that $\gamma > 0$ for the model specified in Eq. [7.14]. This means that the O–D flows increase when either the travel time to the destination decreases or the destination attraction increases (assuming that all other factors remain unchanged). Both types of parameters ($\{M_s\}$ and γ) can be estimated (calibrated) from observations regarding motorists' choices of destination. Issues associated with the preparation of data and estimation of logit models are discussed in Chapter 13.

At equilibrium, then, the flow pattern should be such that the trip rates satisfy Eqs. [7.14] and the UE conditions hold for all O–D pairs. As with the other equilibration problems discussed in this text, this problem can also be solved by minimizing an equivalent mathematical program. This approach is outlined below.

Consider the mathematical program†

$$\min z(\mathbf{x}, \mathbf{q}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega + \frac{1}{\gamma} \sum_{rs} (q_{rs} \ln q_{rs} - q_{rs}) - \sum_{rs} M_s q_{rs} \qquad [7.15a]$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\, r, s \quad (u_{rs}) \qquad [7.15b]$$

$$\sum_s q_{rs} = O_r \qquad \forall\, r \quad (\mu_r) \qquad [7.15c]$$

$$f_k^{rs} \geq 0 \qquad \forall\, k, r, s \qquad [7.15d]$$

The first-order conditions for this program include the familiar UE equations for the basic network. As before, these conditions can be obtained by differentiating the Lagrangian of [7.15] with respect to the path-flow variables, $\{f_k^{rs}\}$. The first-order conditions with respect to the O–D flow variables are

$$\frac{1}{\gamma} \ln q_{rs} + u_{rs} - M_s - \mu_r = 0 \qquad \forall\, r, s \qquad [7.16a]$$

These conditions can be rewritten as

$$q_{rs} = e^{-\gamma(u_{rs} - M_s - \mu_r)} \qquad \forall\, r, s \qquad [7.16b]$$

Equation [7.16b] can be interpreted as a demand model for trips between each O–D pair r and s. The dual variable μ_r, however, appears in the equation. The value of this variable is not known in advance of the equilibration process, and thus a demand model such as Eq. [7.16b] cannot be calibrated from observations of O–D travel times and flows. Furthermore, this O–D demand function would not satisfy the trip production constraints (Eqs. [7.15c]) which are part of the first-order conditions. For the production constraint to hold at equilibrium, Eq. [7.16b] can be substituted in [7.15c] to give

$$O_r = \sum_m e^{-\gamma(u_{rs} - M_m - \mu_r)} \qquad \forall\, r \qquad [7.17]$$

Equations [7.16b] and [7.17] can be divided by each other to obtain the multinomial logit formula, that is,

$$\frac{q_{rs}}{O_r} = \frac{e^{-\gamma(u_{rs} - M_s - \mu_r)}}{\sum_m e^{-\gamma(u_{rm} - M_m - \mu_r)}} = \frac{e^{-\gamma(u_{rs} - M_s)}}{\sum_m e^{-\gamma(u_{rm} - M_m)}} \qquad [7.18]$$

†Note that $\lim_{q_{rs} \to 0^+} q_{rs} \ln q_{rs} = 0$ and therefore $q_{rs} \ln q_{rs}$ can be assumed to be zero for $q_{rs} = 0$. Note also that the constraint $0 < q_{rs} < O_r$, $\forall\, r, s$, could be added to the formulation of program [7.15]. This constraint, however, does not change the first-order conditions given in Eqs. [7.16]. Due to the shape of the objective function, it will always be satisfied at the minimum.

as $e^{\gamma\mu_r}$ drops out of the ratio. Thus the first-order conditions of program [7.15] are the equilibrium conditions for the distribution/assignment problem with logit destination demand functions.

Direct Algorithm and Solution by Network Representation

The logit distribution/assignment problem formulated in Eqs. [7.15] can be solved by a simple adaptation of the convex combinations method. In fact, the LP solved in the direction-finding step is identical to the one discussed in Section 7.1 (see Eqs. [7.6] and the related discussion) except for an added term in the objective function. Again, the linear program that has to be solved at the nth iteration can be written in terms of path-flow variables, as follows:

$$\min z^n(\mathbf{g}) = \sum_{rs} \sum_{k} \frac{\partial z[\mathbf{x}(\mathbf{f}^n), \mathbf{q}(\mathbf{f}^n)]}{\partial f_k^{rs}} g_k^{rs}$$

$$= \sum_{rs} \sum_{k} [c_k^{rs^n} + \frac{1}{\gamma} \ln (q_{rs}^n) - M_s] g_k^{rs} \qquad [7.19a]$$

subject to

$$\sum_{s} \left(\sum_{k} g_k^{rs} \right) = O_r \qquad \forall\, r \qquad [7.19b]$$

$$g_k^{rs} \geq 0 \qquad \forall\, k, r, s \qquad [7.19c]$$

The solution of this program parallels the solution of program [7.6]. Its details are left to the reader (see Problem 7.8). Note only that once the auxiliary flows, $\{y_a^n\}$ and $\{v_{rs}^n\}$, have been obtained, the move size in the descent direction $\{y_a^n - x_a^n\}$ and $\{v_{rs}^n - q_{rs}^n\}$ is obtained by searching for the minimum of the objective function [7.17a] along the descent direction. In other words, the move size, α_n, is given by the solution of the program

$$\min_{0 \leqslant \alpha \leqslant 1} z(\alpha) = \sum_{a} \int_{0}^{x_a^n + \alpha(y_a^n - x_a^n)} t_a(\omega)\, d\omega$$

$$+ \frac{1}{\gamma} \sum_{rs} [q_{rs}^n + \alpha(v_{rs}^n - q_{rs}^n)]\{\ln [q_{rs}^n + \alpha(v_{rs}^n - q_{rs}^n)] - 1 - \gamma M_s\} \qquad [7.20]$$

The convergence criterion in this algorithm can be based on the distribution model. In other words, the algorithm can be terminated when

$$\frac{q_{rs}^{n+1}}{O_r} = \frac{e^{-\gamma(u_{rs}^n - M_s)}}{\sum_m e^{-\gamma(u_{rm}^n - M_m)}} \qquad \forall\, r, s \qquad [7.21]$$

An actual criterion can be based on the sum of the relative difference between both sides of Eq. [7.21] over all O–D pairs. This criterion is based on considerations similar to those used in Section 6.2 for variable-demand UE program (see Eq. [6.20] and the related discussion). As suggested there, this criterion can be used in conjunction with a criterion measuring the satisfaction of the UE conditions over the basic network.

The algorithm, then, can be summarized as follows:

Step 0: *Initialization.* Find a set of feasible flows $\{q_{rs}^n\}$, $\{x_a^n\}$. Set $n := 1$.

Step 1: *Travel-time update.* Calculate $t_a^n = t_a(x_a^n)$, $\forall\ a$.

Step 2: *Direction finding.* Find $\{g_k^{rs^n}\}$ that minimize program [7.19]. Set

$$y_a^n = \sum_{rs} \sum_k g_k^{rs^n} \delta_{a,k}^{rs}$$

$$v_{rs}^n = \sum_k g_k^{rs^n}$$

Step 3: *Move-size determination.* Find the α_n that solves program [7.20].

Step 4: *Flow update.* Set

$$x_a^{n+1} = x_a^n + \alpha_n(y_a^n - x_a^n)$$

$$q_{rs}^{n+1} = q_{rs}^n + \alpha_n(v_{rs}^n - q_{rs}^n)$$

Step 5: *Convergence test.* If convergence is not achieved, set $n := n + 1$ and go to step 1. Otherwise, terminate; the solution is $\{x_a^{n+1}\}$, $\{q_{rs}^{n+1}\}$.

The logit distribution/assignment problem can also be solved by using a standard UE algorithm, provided that the network representation of the problem is modified. The augmented network is identical to the one shown in Figure 7.2, including dummy (or virtual) links connecting every destination, s, to a dummy (virtual) destination, r'. The equivalent performance functions on these links are given by

$$t_{sr'} = \frac{1}{\gamma} \ln q_{rs} - M_s \qquad \forall\ r, s \qquad [7.22]$$

The equivalent UE minimization on this modified network (with O–D trip rates between r and r' given by O_r) is identical to the logit distribution/assignment program given in Eqs. [7.15], as the reader can check (see Problem 7.9a).† The equivalent UE program on the modified network can be solved with any standard UE algorithm. At equilibrium, the travel time on all used paths between each O–D pair of the augmented network ($s \rightarrow r'$) would exhibit the same ("equivalent") travel time. This condition on the augmented network is equivalent to the first-order conditions given in Eq. [7.18] (see Problem 7.9b).

Note that due to the logarithmic form of the performance curves on the

†Note that by the construction of the network in Fig. 7.2, $q_{sr'} = q_{rs}$ since only flow between r and r' can use the link from s to r'.

dummy links, these links cannot admit zero flow at any iteration. One safeguard against this is the truncation of the equivalent performance functions at an appropriate point. In other words, set $t_{sr'}$ to

$$t_{sr'} = \begin{cases} \dfrac{1}{\gamma} \ln q_{rs} - M_s & \text{for } q_{rs} > \epsilon \\ \dfrac{1}{\gamma} \ln \epsilon - M_s & \text{for } q_{rs} \leq \epsilon \end{cases}$$

where ϵ is a sufficiently small amount of flow.

Particular care should be taken during the initialization phase since the modified network cannot be initialized with zero flow. A good initialization procedure is to calculate the travel times on the minimum path connecting each O–D pair of the basic network, based on empty network travel times. These O–D travel times, $\{u_{rs}^0\}$, are then used to determine the initial O–D flow through an application of a logit distribution formula, that is,

$$q_{rs}^0 = \bar{q}_{rs} \frac{e^{-\gamma(u_{rs}^0 - M_s)}}{\sum_m e^{-\gamma(u_{rm}^0 - M_m)}} \qquad [7.23]$$

The nature of the flow sequence generated by the convex combinations algorithm will tend to avoid a solution in which $q_{rs}^n = 0$ once an initial feasible solution is given. At the equilibrium point itself it will always be true that $q_{rs} > 0$ as mentioned above (see footnote associated with Eq. [7.15]).

Example

Consider the simple network example shown in Figure 7.3. This network includes two links (indexed 1 and 2) leading to two destinations (denoted 1 and 2, respectively). The performance functions on these links are given by

$$t_1 = 1 + 2x_1 \qquad [7.24a]$$

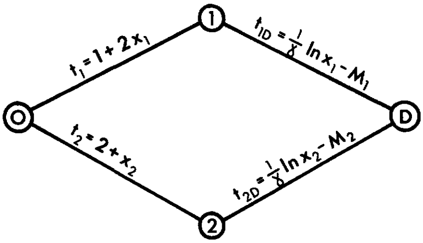
$$t_2 = 2 + x_2 \qquad [7.24b]$$

The total flow emanating from the origin is $O = 4$ units of flow. Assume now that at equilibrium, the flow from the origin to each of the destinations should conform to a logit model with $\gamma = 1.0$, and $M_s = 0$ for each destination, that is,

$$\frac{x_1}{4} = \frac{e^{-t_1}}{e^{-t_1} + e^{-t_2}} \qquad [7.24c]$$

$$\frac{x_2}{4} = \frac{e^{-t_2}}{e^{-t_1} + e^{-t_2}} \qquad [7.24d]$$

The system of equations [7.24] includes four equations in four unknowns (x_1, x_2, t_1, and t_2). Note that the trip production constraint is embedded in Eqs. [7.24c] and [7.24d]. (For example, Eq. [7.24d] can be replaced by the



1
$t_1 = 1 + 2x_1$
$t_{1D} = \frac{1}{\gamma} \ln x_1 - M_1$
O
D
$t_2 = 2 + x_2$
$t_{2D} = \frac{1}{\gamma} \ln x_2 - M_2$
2

Figure 7.4 Network representation for solving the joint distribution/assignment problem of Figure 7.3 using a UE algorithm.

condition $x_2 = 4 - x_1$.) These equations can be solved simultaneously by substituting the link performance functions and the flow conservation constraint in the logit formula [7.24c]. The result is a simple equation in terms of x_1, that is,

$$\frac{x_1}{4} = \frac{1}{1 + e^{3x_1 - 5}}$$

or

$$3x_1 - 5 - \ln\left(\frac{4}{x_1} - 1\right) = 0 \qquad [7.25]$$

The solution of this equation is $x_1 = 1.9$ flow units,† implying that $x_2 = 2.1$, $t_1 = 4.8$, and $t_2 = 4.1$.

This example can also be solved by using the equivalent minimization or network representation approach developed in this section. The modified network for this problem includes one dummy link connecting each of the destinations in Figure 7.3 to a "super destination," as shown in Figure 7.4. The performance functions for the dummy links are given by ln x_1 and ln x_2 for links 1 → D and 2 → D, respectively.

The equivalent UE minimization formulation for the modified network in Figure 7.4 is given by

$$\min z(x_1, x_2) = \int_0^{x_1} (1 + 2\omega)\, d\omega + \int_0^{x_2} (2 + \omega)\, d\omega + \int_0^{x_1} \ln \omega\, d\omega + \int_0^{x_2} \ln \omega\, d\omega \qquad [7.26a]$$

subject to

$$x_1 + x_2 = 4 \qquad [7.26b]$$

$$x_1, x_2 \geq 0 \qquad [7.26c]$$

When the objective function is integrated and the flow conservation constraint, $x_2 = 4 - x_1$, is substituted in, the result is

$$z(x_1) = 1\tfrac{1}{2}x_1^2 - 5x_1 + 16 + x_1 \ln x_1 - x_1 + (4 - x_1) \ln (4 - x_1) - 4 + x_1$$

†More accurately, $x_1 = 1.91261$ flow units.

The minimum of this function is found by solving

$$\frac{dz(x_1)}{dx_1} = 3x_1 - 5 + \ln x_1 - \ln (4 - x_1) = 0$$

This equation is identical to [7.25] and its solution is $x_1 = 1.9$ (a point that satisfies constraint [7.26c]). As expected, the solution obtained for this example by the equivalent network representation is indeed the correct equilibrium flow pattern.

Computational Issues

The solution of the joint distribution/assignment program requires more computational effort than the solution of the UE program. This effort can be discussed in the framework of the network representation of this problem. Since the distribution/assignment program requires the addition of a large number of dummy links (equal to the product of the number of origins and the number of destinations) and a number of dummy nodes (equal to the number of origins), the computer memory requirements are larger than those associated with the solution of a standard UE problem. In addition, the computation of each minimum path tree (in the process of solving the linear program) will require more CPU time than is needed with the fixed-demand UE case, due to the larger size of the equivalent network. The additional burden of the trip distribution model, however, is not limited to the core requirements and the computational effort needed to find the minimum path trees. In the case of the distribution/assignment model, the application of a standard UE equilibration method such as the convex combination algorithm may be quite inefficient (whether the algorithm is applied directly or to the augmented network representation). The reason is that at every iteration only one destination is loaded with flow from any given origin. In other words, only the destination m for which

$$u_{rm}^n + \frac{1}{\gamma} \ln q_{rm}^n - M_m = \min_{\forall s} \left\{ u_{rs}^n + \frac{1}{\gamma} \ln q_{rs}^n - M_s \right\} \qquad [7.27]$$

is loaded with flow. This flow, O_r, is assigned to the (current) shortest path between r and m (the travel time on which is u_{rm}^n). Consequently, the algorithm would not terminate before most destinations s, for which $q_{rs} > 0$ at equilibrium, have been "visited" (at least once) from origin r.

The implications of these computational requirements are that the size of networks for which joint distribution/assignment problems can be solved is substantially smaller than those for which a standard (fixed-demand) UE can be solved. These observations give rise to the use of another algorithm which does not suffer from the above-mentioned drawbacks. This algorithm is described in the following section.

Double-Stage Algorithm

The double-stage algorithm is similar to the convex combinations algorithm in all aspects but the direction-finding step. This algorithm uses a descent direction in which every destination is loaded with trips from each origin at every iteration. This can be accomplished by executing the direction-finding step in two stages, as follows:

Direction finding:

(a) Calculate the shortest travel-time path from each origin, r, to all destinations, based on $\{t_a^n\}$. Let u_{rs}^n denote the shortest travel time from r to s.

(b) Determine the auxiliary O–D flows by applying a logit distribution model, that is,

$$v_{rs}^n = \frac{O_r e^{-\gamma(u_{rs}^n - M_s)}}{\sum_m e^{-\gamma(u_{rm}^n - M_m)}}$$

(c) Assign v_{rs}^n to the minimum-travel-time path (identified above) between r and s. This yields (after all $\{v_{rs}^n\}$ have been assigned) a link-flow pattern $\{y_a^n\}$.

The rest of the algorithmic iterations are identical to those of the convex combinations method. The search direction is defined in terms of $\{(y_a^n - x_a^n)\}$ and $\{(v_{rs}^n - q_{rs}^n)\}$ and the new solution is determined by the optimal convex combination of the auxiliary and the last solution. This optimal combination is found by solving a one-dimensional program similar to the one shown in Eq. [7.20], updating the flow variables and checking for convergence.†

The search direction used in this (double-stage) algorithm can be regarded as the solution of the following mathematical program:

$$\min \hat{z}(\mathbf{y}, \mathbf{v}) = \sum_a t_a^n y_a + \frac{1}{\gamma} \sum_{rs} v_{rs}[\ln v_{rs} - 1] - \sum_{rs} v_{rs} M_s \qquad [7.28a]$$

subject to

$$\sum_k g_k^{rs} = v_{rs} \qquad \forall\ r, s \qquad [7.28b]$$

$$\sum_s v_{rs} = O_r \qquad \forall\ r \qquad [7.28c]$$

$$g_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [7.28d]$$

This program is convex (see Problem 7.10) and thus its first-order conditions

†The convergence check can be based on the objective function values, successive flow patterns (including both O–D and link flows), travel times over the basic network, or by a combined criterion.

are sufficient for a minimum. These first-order conditions are

$$(c_k^{rsn} - u_{rs}^n)g_k^{rsn} = 0 \qquad \forall\ k, r, s \qquad [7.29a]$$

$$c_k^{rsn} - u_{rs}^n \geq 0 \qquad \forall\ k, r, s \qquad [7.29b]$$

$$\frac{v_{rs}^n}{O_r} = \frac{e^{-\gamma(u_{rs}^n - M_s)}}{\sum_m e^{-\gamma(u_{rm}^n - M_s)}} \qquad \forall\ r, s \qquad [7.29c]$$

and constraints [7.28b] through [7.28d]. Thus, at the solution of program [7.28], the auxiliary O–D flows are given by the logit model. Given the (auxiliary) O–D flows, objective function [7.28a] can be minimizing by loading all $\{v_{rs}^n\}$ to the (current) minimum travel-time path between r and s. This will result in a situation in which only one path (the travel time on which is u_{rs}^n) between every O–D pair is loaded with flow and Eqs. [7.29a] and [7.29b] are satisfied. Thus the direction-finding step of the double-stage algorithm can be regarded as a solution of program [7.28], while the direction-finding step of the convex combination method can be regarded as a solution of program [7.19], which is derived from the linearized objective function of the equivalent minimization.

The proof that the double-stage algorithm minimizes the equivalent distribution/assignment program (Eqs. [7.15]) is given in the references mentioned in Section 7.6. The reader is asked (in Problem 7.10) to provide a part of this proof, showing that this algorithm generates a descent direction at every iteration.

Convergence of the double-stage algorithm is considerably faster than convergence of the convex combinations algorithm for solving logit-based distribution/assignment problems. In particular, when the network is lightly congested, the computational effort posed by this algorithm is comparable to the effort required to solve a standard (fixed-demand) UE problem. As congestion increases, the relative advantage of the double-stage algorithm over the convex combinations method decreases. For the range of congestion found in practice, however, the double-stage algorithm is still the preferred approach.

Note that a similar double-stage algorithm can also be used to minimize the equivalent program associated with the variable-demand user-equilibrium program discussed in Chapter 6. As the reader is asked to show (see Problem 7.11), the application of the double-stage algorithm to the variable-demand problem is virtually identical to the application of this algorithm to the joint distribution/assignment problem.

7.3 DOUBLY CONSTRAINED MODELS

The trip distribution models discussed thus far in this chapter assume that the number of trips leaving each origin is known and that the problem is one of finding how these trips are distributed among the various destinations. A different approach to modeling trip distribution assumes that both the total flow generated at each origin node and the total flow attracted to each destination node are fixed and known. The problem is still that of finding the O–D flows, but the flows become subject to the two sets of constraints (see Eqs. [7.1] and [7.2])

$$\sum_s q_{rs} = O_r \qquad \forall\ r \qquad [7.30a]$$

$$\sum_r q_{rs} = D_s \qquad \forall\ s \qquad [7.30b]$$

At equilibrium, the O–D flows are derived from some trip distribution model, and also satisfy these constraints. The distribution model used in conjunction with these constraints is explained in this section. This explanation leads to the formulation of an appropriate equivalent minimization for the doubly constrained distribution/assignment model.

It is reasonable to assume that the flow between any O–D pair r–s, q_{rs}, will be proportional to the total number of trips leaving the origin, O_r, and the total number of trips attracted to the destination, D_s. Naturally, the equilibrium O–D flow should also depend on the level of service between the origin and the destination. In other words, q_{rs} should also depend on $f(u_{rs})$, where $f(\cdot)$ is some function and u_{rs} is the O–D travel time. This dependence is reflected in the following trip distribution model:

$$q_{rs} = KO_rD_sf(u_{rs}) \qquad \forall\ r, s \qquad [7.31]$$

where K is a constant. One example of such a relationship is the widely used gravity model.† In this model $f(u_{rs}) = u_{rs}^{-2}$, that is,

$$q_{rs} = K\frac{O_rD_s}{u_{rs}^2} \qquad \forall\ r, s \qquad [7.32]$$

Constraints [7.30] can be satisfied by introducing sets of constants $\{A_r\}$ and $\{B_s\}$ associated with the origin and destination zones respectively. The trip distribution model then becomes

$$q_{rs} = A_rB_sO_rD_sf(u_{rs}) \qquad \forall\ r, s \qquad [7.33]$$

where

$$A_r = \frac{1}{\sum_s B_sD_sf(u_{rs})} \qquad \forall\ r \qquad [7.34a]$$

$$B_s = \frac{1}{\sum_r A_rO_rf(u_{rs})} \qquad \forall\ s \qquad [7.34b]$$

†This model assumes that the number of trips between any two urban zones parallels the gravitational force between two bodies. The number of O–D trips is assumed to be proportional to the total trip production of the origin and the total trip attraction of the destination (these are analogous to the masses of the two bodies), and inversely proportional to the square of the travel time between the origin and the destination (which is analogous to the distance between the bodies).

By substituting Eqs. [7.34] into [7.33], it can be easily verified that constraints [7.30] are satisfied. The problem addressed in this section is how to solve for the coefficients A_r and B_s in Eqs. [7.34], and what the proper function form of $f(u_{rs})$ is. This is described in the context of a specific approach to this problem—one that is based on entropy maximization principles.

Entropy Model

A given set of O–D flows $\{q_{rs}\}$ that satisfies the flow conservation constraints is the aggregate result of many individual travel decisions.† There can be many combinations of individual decisions (each such combination is called "a state" of the system) which result in the same set of O–D flows $\mathbf{q} = (\ldots, q_{rs}, \ldots)$. The basic assumption of the approach discussed here is that all states are equally likely to occur and thus the likelihood of a given distribution pattern, $\mathbf{q}$, occurring is proportional to the number of states that result in this pattern while satisfying the constraints. The number of states associated with a particular distribution, $N(\mathbf{q})$, is given by

$$N(\mathbf{q}) = \frac{Q!}{\prod_{r,s} q_{rs}!} \qquad [7.35]$$

where Q is the total number of trips in the system (i.e., $Q = \sum_r \sum_s q_{rs}$‡). The set of O–D flows with the highest likelihood of occurring is the set with the maximum number of states. This set of O–D flows can be determined by maximizing Eq. [7.35], or its logarithm,§ subject to the flow conservation constraints. The objective function is then

$$\max \ln N(\mathbf{q}) = \ln \frac{Q!}{\prod_{rs} q_{rs}} = \ln Q! - \sum_{rs} \ln q_{rs}! \qquad [7.36]$$

The term ln $Q!$ in Eq. [7.36] is a constant that can be dropped from the objective function. Furthermore, the maximization of ln $N(\mathbf{q})$ is equivalent to the minimization of $-\ln N(\mathbf{q})$, and thus the objective function can be ex-

†For the purposes of this discussion, the trip rate is assumed to be an integer representing the number of individuals traveling from r to s during the design period.

‡Formula [7.35] gives the number of permutations of Q objects of which each q_{rs} are alike. Here the objects are the O–D trips and the formula gives the number of permutations in which Q trips can produce a given set of O–D flows, $\{q_{rs}\}$.

§In many cases it is easier to maximize the logarithm of a function than the function itself. Since the logarithm is a monotonic transformation, the same values of the decision variables will maximize both the function and its logarithm. This approach is used widely in statistical application for maximizing likelihood functions.

Note that the logarithm of the probability of occurrence of the most likely molecular arrangement in a closed thermodynamic system is known as the *entropy* of the system. This analogy is the source of the model's name.

pressed as

$$\min \left[\sum_{rs} \ln q_{rs}! \right]$$

Using Stirling's formula,‡ this objective function becomes

$$\min \sum_{rs} (q_{rs} \ln q_{rs} - q_{rs}) \qquad [7.37]$$

The entropy trip distribution model therefore calls for the most likely O–D trip rate pattern (under the model's assumptions) subject to the trip production and attraction constraints, given by Eqs. [7.30a] and [7.30b], respectively.

Combining the entropy concept with the approach utilized throughout this text (that of using equivalent mathematical programs to solve for equilibrium), a joint entropy distribution/assignment program can be formulated. Consider the following minimization:

$$\min z(\mathbf{x}, \mathbf{q}) = \sum_a \int_0^{a} t_a(\omega)\, d\omega + \frac{1}{\zeta} \sum_{rs} (q_{rs} \ln q_{rs} - q_{rs}) \qquad [7.38a]$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \quad \forall\, r, s \quad (u_{rs}) \qquad [7.38b]$$

$$\sum_s q_{rs} = O_r \quad \forall\, r \quad (\mu_r) \qquad [7.38c]$$

$$\sum_r q_{rs} = D_s \quad \forall\, s \quad (\lambda_s) \qquad [7.38d]$$

$$f_k^{rs} \geq 0 \quad \forall\, k, r, s \qquad [7.38e]$$

In this program ζ is a parameter (calibrated from data) and the dual variables associated with the flow conservation constraints are given in parentheses next to these constraints. The solution of this program is a set of O–D trip rates and link flows that satisfies the following requirements: (1) the link flows are such that the UE criterion is satisfied; (2) the O–D flows satisfy a distribution model of the form given in Eq. [7.33], thereby satisfying the production and attraction constraint; and (3) the O–D flows satisfy a distribution model that is based on the entropy concept.

The last requirement stems from the inclusion of the entropy term in objective function [7.38a]. To see that the first two requirements are satisfied at the minimum of program [7.38], its first-order conditions are derived below, focusing on the O–D flows; as the reader can check (see Problem 7.14), the derivatives of the Lagrangian of program [7.38] with respect to the path-flow variables results in the familiar UE conditions for the basic network. The derivatives of this Lagrangian with respect to the O–D flow variable q_{rs} are

‡Stirling's approximation is log $x! \simeq x \log x - x$.

given by†

$$\frac{1}{\zeta}\ln q_{rs} + u_{rs} - \mu_r - \lambda_s = 0 \qquad \forall\, r, s \qquad [7.39a]$$

These equations imply that

$$q_{rs} = e^{-\zeta(u_{rs} - \mu_r - \lambda_s)} \qquad \forall\, r, s \qquad [7.39b]$$

When these expressions are substituted into the flow conservation constraints, the result is

$$e^{\zeta\mu_r} = \frac{O_r}{\sum_s e^{-\zeta(u_{rs} - \lambda_s)}} \qquad \forall\, r \qquad [7.40a]$$

$$e^{\zeta\lambda_s} = \frac{D_s}{\sum_r e^{-\zeta(u_{rs} - \mu_r)}} \qquad \forall\, s \qquad [7.40b]$$

To obtain the result in a more familiar form, denote

$$A_r = \frac{e^{\zeta\mu_r}}{O_r} \qquad \forall\, r \qquad [7.41a]$$

$$B_s = \frac{e^{\zeta\lambda_s}}{D_s} \qquad \forall\, s \qquad [7.41b]$$

Equation [7.39b] can now be written as

$$q_{rs} = A_r B_s O_r D_s e^{-\zeta u_{rs}} \qquad \forall\, r, s \qquad [7.42]$$

where

$$A_r = \frac{1}{\sum_s B_s D_s e^{-\zeta u_{rs}}} \qquad \forall\, r \qquad [7.43a]$$

$$B_s = \frac{1}{\sum_r A_r O_r e^{-\zeta u_{rs}}} \qquad \forall\, s \qquad [7.43b]$$

Thus the entropy-based equivalent minimization in program [7.38] results in an O–D flow pattern that is consistent with the requirements of a joint trip distribution/traffic assignment model. The entropy model is characterized by the exponential decay form of the travel-time function; that is, in this model $f(u_{rs}) = e^{-\zeta u_{rs}}$, where ζ is a model parameter.‡

†Recall that

$$\frac{d}{dx}(x \ln x - x) = \ln x$$

‡The value of ζ can be found by calibration methods, using a data set that includes an O–D matrix and O–D travel times. Some of these methods are mentioned in Chapter 13, and others can be found in the literature concerning the calibration of gravity models (see Section 7.5).

Solution by the Convex Combinations Algorithm

The entropy distribution/assignment model can be solved by using the convex combinations algorithm. Following the description of this algorithm given in Sections 6.2 and 7.2, let g_k^{rs} and v_{rs} represent the auxiliary flow variables corresponding to f_k^{rs} and q_{rs}, respectively. The linear program at the nth iteration of the algorithm can then be written as

$$\min z^n(\mathbf{g}, \mathbf{v}) = \sum_{rs} \sum_k \frac{\partial z(\mathbf{f}^n)}{\partial f_k^{rs}} g_k^{rs} = \sum_{rs} \sum_k \left[c_k^{rsn} + \frac{1}{\zeta} \ln q_{rs}^n \right] g_k^{rs} \qquad [7.44a]$$

subject to

$$\sum_s v_{rs} = O_r \qquad \forall\ r \qquad [7.44b]$$

$$\sum_r v_{rs} = D_s \qquad \forall\ s \qquad [7.44c]$$

$$\sum_k g_k^{rs} = v_{rs} \qquad \forall\ r, s \qquad [7.44d]$$

$$g_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [7.44e]$$

If the set of auxiliary O–D flows that solves program [7.44], $\{v_{rs}^n\}$, is known, this program can be decomposed by O–D pair. Clearly, the objective function of the linear program will be minimized, in this case, by assigning all O–D flows (i.e., v_{rs}^n) to the minimum-travel-time path connecting each O–D pair r–s. The problem is then to find $\{v_{rs}^n\}$. Let the travel time on the shortest path connecting r to s, at the nth iteration, be denoted by u_{rs}^n. The problem of finding $\{v_{rs}^n\}$ can now be written as (see program [7.44])†

$$\min z^n(\mathbf{v}) = \sum_{rs} \left[u_{rs}^n + \frac{1}{\zeta} \ln q_{rs}^n \right] v_{rs} \qquad [7.45a]$$

subject to

$$\sum_s v_{rs} = O_r \qquad \forall\ r \qquad [7.45b]$$

$$\sum_r v_{rs} = D_s \qquad \forall\ s \qquad [7.45c]$$

$$v_{rs} \geq 0 \qquad \forall\ r, s \qquad [7.45d]$$

The problem defined by Eqs. [7.45] can be described as follows: Find the optimal flow pattern by which a set of supply depots (indexed by r) can ship a given commodity to a set of demand depots (indexed by s). The total amount available at each supply depot is O_r, the total amount consumed at each demand depot is D_s, and the cost of transporting a unit of the commodity

†Based on the above-mentioned considerations, objective function [7.44a] can be written as $\sum_{rs} [(u_{rs}^n + (1/\zeta) \ln q_{rs}^n) \sum_k g_k^{rs}]$. Constraint [7.44d] can then be substituted in this objective function to obtain objective function [7.45a].

under consideration between depots r and s is $c_{rs}^n = u_{rs}^n + (1/\zeta) \ln q_{rs}^n$. The objective is to find the set of flow variables, $\{v_{rs}^n\}$, that minimizes the total transportation costs. This program is known in the operations research literature as "Hitchcock's transportation problem." Section 7.4 describes one of the most efficient algorithms for solving this program.

Once the auxiliary O–D flows, $\{v_{rs}^n\}$, are determined, the auxiliary link flows, $\{y_a^n\}$, can be found by assigning v_{rs}^n to the shortest path connecting each O–D pair r–s. The move size is determined by minimizing the objective function [7.38a] along the descent direction, and the convergence criterion can be set in a manner similar to the criteria used earlier in this text. The algorithm, then, can be summarized as follows:

Step 0: *Initialization.* Find a feasible solution $\{q_{rs}^n\}$, $\{x_a^n\}$. Set $n := 1$.

Step 1: *Travel-time update.* Set $t_a^n = t_a(x_a^n)$.

Step 2: *Direction finding.*

(a) Compute $\{u_{rs}^n\}$; set $c_{rs}^n = u_{rs}^n + (1/\zeta) \ln q_{rs}^n$.

(b) Solve Hitchcock's transportation problem with costs $\{c_{rs}^n\}$. This yields $\{v_{rs}^n\}$.

(c) Assign $\{v_{rs}^n\}$ to the minimum paths identified in (a). This yields $\{y_a\}$.

Step 3: *Move-size determination.* Find α_n that solves

$$\min_{0 \leqslant \alpha \leqslant 1} \sum_a \int_0^{x_a^n + \alpha(y_a^n - x_a^n)} t_a(\omega)$$

$$+ \frac{1}{\zeta} \sum_{rs} [q_{rs}^n + \alpha(v_{rs}^n - q_{rs}^n)]\{\ln [q_{rs}^n + \alpha(v_{rs}^n - q_{rs}^n)] - 1\}$$

Step 4: *Flow update.* Set

$$x_a^{n+1} = x_a^n + \alpha_n(y_a^n - x_a^n) \qquad \forall\ a$$

$$q_{rs}^{n+1} = q_{rs}^n + \alpha_n(v_{rs}^n - q_{rs}^n) \qquad \forall\ r, s$$

Step 5: *Convergence criterion.* If $\sum_a |x_a^{n+1} - x_a^n|/x_a^n \le \varepsilon$, stop. Otherwise, set $n := n + 1$ and go to step 1.

Note that the direction-finding step of this algorithm involves the storage of the shortest paths identified in step 2a while the transportation problem is being solved in step 2b. These paths are assigned flow only in step 2c. This increases the computer memory requirements of the algorithm. To avoid that, the minimum paths can be computed anew in step 2c, meaning that the computational effort is virtually doubled.

Solution of the doubly constrained model is likely to require a smaller number of iterations than the solution of the singly constrained model when the convex combinations method is used for both. The reason is that in typical networks more than one destination is being visited from each origin. When the convex combinations method is being applied to the doubly constrained model, each iteration involves flow assignment from each origin to two or more destinations (see Problem 7.18). By comparison, the solution of the singly constrained model with this method involves the assignment of flow to only one destination as mentioned before. Consequently, the former may involve a smaller number of iterations. The computational effort required per iteration, however, may be higher for the doubly constrained model due to the need to solve Hitchcock's problem at every iteration and the above-mentioned recalculation (or storage) of the shortest-path trees.

Double-Stage Algorithm

A double-stage algorithm similar to the one described in connection with the singly constrained model can be developed for solving this problem as well. At each iteration the direction-finding step would include identification of all the minimum O–D travel times $\{u_{rs}^n\}$. These values will then be used to solve a doubly constrained entropy model. In other words, the auxiliary O–D flows, $\{v_{rs}\}$, will be found by solving the program (see Eqs. [7.37] and [7.38])

$$\min z(\mathbf{v}) = \sum_{rs} u_{rs}^n v_{rs} + \frac{1}{\zeta} \sum_{rs} (v_{rs} \ln v_{rs} - v_{rs}) \qquad [7.46]$$

subject to constraints [7.38c], [7.38d], and nonnegativity (all expressed in terms of $\{v_{rs}\}$). The auxiliary O–D flow variables are then assigned to the minimum path connecting each O–D pair. The reader is asked (in Problem 7.15) to investigate the application of the double-stage algorithm to the doubly constrained distribution/assignment problem.

Note that unlike the case with singly constrained models, the double-stage algorithm has no apparent advantage over the convex combinations method, when both are applied to the solution of doubly constrained models. The reasons are twofold. First, in both algorithms the O–D flows from all origins to all destinations are determined simultaneously. In other words, even in the applications of the convex combinations method, more than one destination can be loaded from each origin at each iteration. Consequently, the convex combinations method is not as inefficient as it is in the case of the singly constrained model. Second, the auxiliary problem [7.46] is a nonlinear program representing the gravity model. It is considerably more difficult to solve than Hitchcock's transportation problem, which has to be solved as part of the convex combinations method.

These considerations imply, then, that the singly constrained model should be solved by using the double-stage algorithm, whereas the doubly constrained model should be solved by using the convex combinations method. Given this use of the algorithms, the doubly constrained model requires a higher computational effort to solve, than the singly constrained model. Even though it is not clear which model will require more iterations, the effort associated with each of the iterations required to solve the doubly constrained model is significantly higher.

Evaluation

Gravity models in general, and the entropy models in particular, are based on parallels drawn between individual travel decisions and the laws of physics. Many researchers argue that such parallels are not valid since urban flow patterns result from collective individual choices. Such travel choices should, according to this view, be modeled on the basis of economic theory and behavioral assumptions. Program [7.38] can nevertheless be viewed as a mathematical construct, used to solve doubly constrained distribution/assignment problems.

Yet even this view is unacceptable to some travel behavior analysts, who argue that singly constrained models are more valid than doubly constrained models. The reason for this is that the network flows result from the daily travel choice process of individuals. These individuals make their decisions as to which destination to visit, while at the trip origin. These trip origins are determined by long-term considerations, and thus when the daily travel choices are modeled, the total number of trips originating from each zone can be regarded as fixed. The total flow arriving at destination nodes, on the other hand, is determined by the sum of many decision processes conducted daily at the origin nodes. In fact, it is the outcome of these decision processes that is modeled in the joint distribution/assignment problem. Constraining the flow destined for a given node is thus a constraint on the outcome of the model, which can be compared to a constraint on the flow on a given link of the basic network. The point of view explained above may be particularly appropriate for modeling nonwork trips. For work trips, however, both trip origins (for example, residence locations) and trip destinations (for example, work places) are based on long-term decisions. Consequently, other researchers argue that in this case, doubly constrained models may be appropriate.

Another consideration in the choice between singly and doubly constrained models is the application of the model. Assume, for example, that the model is to be used for predicting the travel consequences of certain improvements to the transportation system. For both doubly and singly constrained models, the number of trips originating at each origin node have to be predicted exogenously to the model. If there are reasons to believe that the number of trips attracted to each destination will be fixed, and this number can be predicted before the model is used, the doubly constrained model is appropriate. If, on the other hand, the number of trips going into each destination is to be determined based on projections of destination attractions and transportation level of service considerations (i.e., congestion, parking availability, etc.), a singly constrained model is appropriate.

In general, there is no empirical basis for the preference of one model over another, and considerations such as those mentioned above should guide the model selection. The total flow attracted to a given destination node, if available in the data, does not necessarily mean that a doubly constrained model should be used. These data can be utilized a posteriori to validate the results of a singly constrained model.

As mentioned before, the solution of the doubly constrained model, with the convex combinations method, requires solving Hitchcock's transportation problem. The next section describes an algorithm for solving this problem.

7.4 SOLVING HITCHCOCK'S TRANSPORTATION PROBLEM†

Hitchcock's transportation problem can be described as a minimum-cost flow problem over the network depicted in Figure 7.5. This network includes I supply and J demand nodes connected by direct links. The notations used in this section to identify the links are the same as those used in conjunction with the minimum-path algorithm described in Section 5.4. Accordingly, let c_{ij} and x_{ij} denote the (fixed) cost (per unit of flow) and the flow, respectively, on the link leading from node i to node j. Furthermore, let O_i denote the total flow supplied by node i and let D_j denote the total flow required at node j. Using these notations, Hitchcock's transportation problem is

$$\min z(\mathbf{x}) = \sum_{i=1}^{I} \sum_{j=1}^{J} c_{ij} x_{ij} \qquad [7.47a]$$

subject to

$$\sum_j x_{ij} = O_i \qquad \forall\, i = 1, 2, \ldots, I \qquad [7.47b]$$

$$\sum_i x_{ij} = D_j \qquad \forall\, j = 1, 2, \ldots, J \qquad [7.47c]$$

$$x_{ij} \geq 0 \qquad \forall\, i, j \qquad [7.47d]$$

Assume further that the total supply equals the total demand,‡ that is,

$$\sum_i O_i = \sum_j D_j \qquad [7.48]$$

An important characteristic of the Hitchcock problem is that, at optimality, the number of links carrying flow equals the minimum number of links that can connect I supply nodes to J demand nodes. In other words, there should be $(I + J - 1)$ links for which $x_{ij} \geq 0$. On all other links $x_{ij} = 0$. This characteristic is the basis for the algorithm described here, an algorithm that entails four basic steps:

Step 1: Select an initial feasible solution with $I + J - 1$ flow-carrying links.

†This section can be skipped without loss of continuity.

‡This requirement guarantees the existence of a feasible solution to program [7.47].

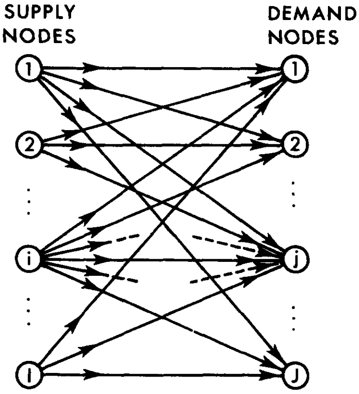


Figure 7.5 Classical "transportation problem" network, including I supply nodes and J consumption (demand) nodes.

Step 2: Check whether the solution can be improved by using a currently empty link. If not, stop; if yes, continue.

Step 3: Determine the amount of flow that can be assigned to the new link without violating any constraint.

Step 4: Adjust the flow on all other flow-carrying links and update the network. Go to step 2.

These steps are detailed below. It is important to remember that the algorithm keeps only $I + J - 1$ links loaded with flow. This set of links is known as the "basis." When a solution can be improved by using a (currently) empty link (identified in step 2) one (currently) loaded link will be emptied (in step 4). In other words, when one link enters the basis, another link leaves it.

Step 1. Step 1 of the process involves finding an initial feasible solution. The core of many of the initialization procedures suggested in the literature is in the link examination process. For a link $i \rightarrow j$, this process consists of the following steps:

(a) If $O_i < D_j$, assign flow O_i to link $i \rightarrow j$. Remove node i (and all links emanating from it) from further consideration. Set $D_j := D_j - O_i$.

(b) If $O_i > D_j$, assign flow D_j to link $i \rightarrow j$. Remove node j (and all links leading to it) from further consideration. Set $O_i := O_i - D_j$.

(c) If $O_i = D_j$, assign flow O_i to link $i \rightarrow j$. Remove node j (and all links leading to it) from further consideration. Set $O_i := 0$.†

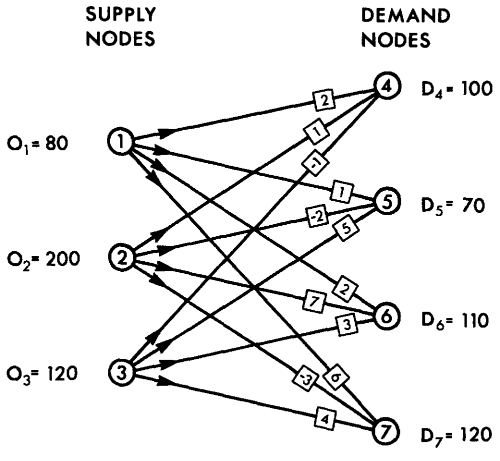
The initialization involves an iterative examination process. At each iteration a single link, which has not yet been considered (and was not discarded at an

†Alternatively, node i can be removed and D_j can be set to zero.

earlier iteration) is examined. The order in which links are examined can be arbitrary but a good ordering can lead to a better initial solution. The algorithm terminates when either a single demand or a single supply node is left. At this point the remaining flow is assigned either from the last supply node to the remaining demand nodes or from the remaining supply nodes to the last demand node.

One of the most efficient approaches for determining the order in which links are to be examined is known as the "penalty method." This approach is based on a "penalty" computed for each node. The penalty is the absolute value of the difference between the two smallest costs on links leading to the node (if it is a demand node) or emanating from it (if it is a supply node). In order to choose the link to be examined, the node with the highest penalty value (which may be either a supply or a demand node) is identified. The link to be examined is the smallest cost link leading to it (for a demand node) or emanating from it (if it is a supply node). Once a link is examined, the penalties are recomputed and a new link is chosen for examination.

As an example, consider the problem depicted in Figure 7.6. This problem involves three supply nodes and four demand nodes. The supply and demand values are shown next to each node and the shipping cost associated with each link (c_{ij}) is shown in the square drawn in that link. The table summarizes all these costs.



Link Costs

TO FROM	④	⑤	⑥	⑦
①	2	1	2	6
②	1	-2	7	-3
③	-1	5	3	4

Figure 7.6 Transportation problem network example with three supply nodes and four demand nodes. The figure shows supply and demand quantities as well as link costs (in the squares and in the bottom table).

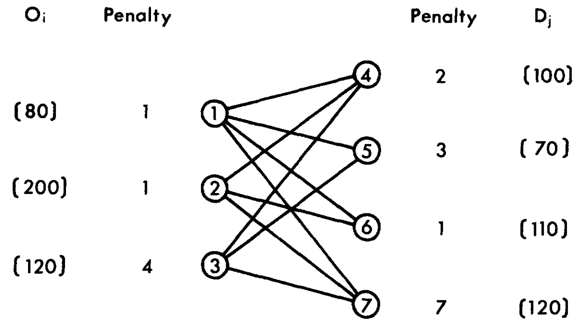


Figure 7.7 The original penalties for the network example. These penalties indicate that link 2 → 7 should be examined first. (Node 7 is associated with the highest penalty and link 2 → 7 with the lowest cost among all links entering node 7.)

Figure 7.7 depicts the original penalties and Figure 7.8 depicts the sequence of iterations of the initialization procedure. The final product of this process (which generates an initial feasible solution to the overall procedure) is given in Figure 7.9.

Step 2. The second step in the algorithm is to check whether the use of a currently empty link can improve the value of the objective function. An

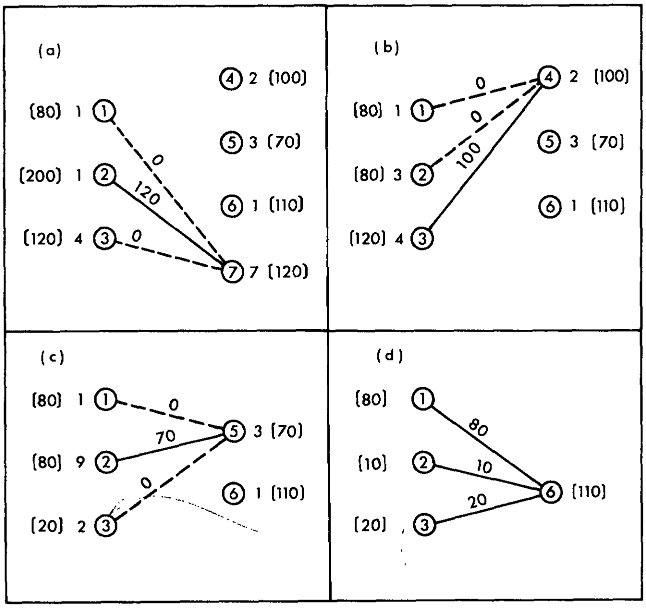


Figure 7.8 Iterations of the initialization procedure (using the penalty method). (a) Link 2 → 7 is examined and assigned a flow of 120 units. Links 1 → 7 and 3 → 7 are assigned zero flow. Node 7 is discarded from further considerations and the penalties are recomputed. (b) Link 3 → 4 is examined. (c) Link 2 → 5 is examined. (d) The remaining demand node is assigned, with the flow using the remaining links.

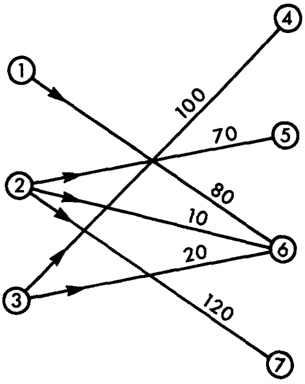


Figure 7.9 The initial solution.

elegant and efficient way of making this decision is to use the dual variables of program [7.47]. Let $\{\mu_i\}$ and $\{\lambda_j\}$ be the dual variables associated with the supply and demand nodes (i.e., with constraints [7.47b] and [7.47c]), respectively. The first-order conditions for this problem are

$$(c_{ij} - \mu_i - \lambda_j)x_{ij} = 0 \qquad \forall\ i, j \qquad [7.49a]$$

$$c_{ij} - \mu_i - \lambda_j \geq 0 \qquad \forall\ i, j \qquad [7.49b]$$

The rationale of the algorithm is to try to create a flow pattern for which conditions [7.49] will hold. At every step of the iterative process there are $I + J - 1$ loaded links (i.e., in the basis) and $IJ - I - J + 1$ empty ones. At optimality, Eqs. [7.49] should hold for all links in the problem. As shown below, the dual variables are constructed so that at each iteration of the algorithm, $c_{ij} - \mu_i - \lambda_j = 0$ for all links in the basis. Equations [7.49] are violated, therefore, only when

$$c_{kl} - \mu_k - \lambda_l < 0 \qquad [7.50]$$

for an empty link (not in the basis) kl. The quantity $c_{ij} - \mu_i - \lambda_j$ is known as the "reduced cost" for link ij. Sending flow over an empty link with negative reduced cost (i.e., bringing it in to the basis by setting $x_{kl} > 0$ and $c_{kl} - \mu_k - \lambda_l = 0$) is a clear move toward optimality (assuming no new violation was created). If several such links exist, any one can be chosen. If none exists, the algorithm terminates since the current solution must be optimal.

This approach requires, of course, that the current values of the dual variables μ_i and λ_i be known. The values of these variables can be obtained for a particular feasible solution by using Eq. [7.49a]. In other words, the values of the dual variables are set so that $c_{ij} = \mu_i + \lambda_j$ for all links in the basis. Starting with an arbitrary value† of one of these dual variables (say, $\mu_1 = 0$), all other values will be uniquely determined. Using this starting point, the

†An arbitrary starting value can be used since only differences between costs and dual variables values count.

Reduced Costs:

FROM \ TO	④	⑤	⑥	⑦
①	4	8	▨	14
②	-2	▨	▨	▨
③	▨	11	▨	11

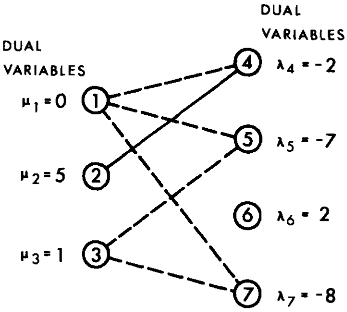


Figure 7.10 Step 2 of the algorithm. The values of the dual variables are shown next to the nodes; the links shown are those not in the basis. The table depicts the reduced costs, indicating that link 2→4 (shown as a solid line) should enter the basis.

value of the dual variables for the solution depicted in Figure 7.9 can be determined by the following progression:

$$\begin{array}{ccccccc} & & & & & & \lambda_4 = -2 \\ & & & & \mu_2 = 5 & & \\ \mu_1 = 0 & \Rightarrow & \lambda_6 = 2 & \Rightarrow & & \Rightarrow & \lambda_5 = -7 \\ & & & & \mu_3 = 1 & & \\ & & & & & & \lambda_7 = -8 \end{array}$$

The reduced costs can now be calculated for each empty link, as shown in Figure 7.10. The reduced cost for link 2→4 is negative and it should therefore enter the basis. The identification of this "candidate" link ends step 2.

Step 3. The third step determines how much flow can be loaded on the candidate link and which (currently loaded) link should be emptied (i.e., leave the basis). Both objectives are achieved by identifying a "closing chain" in the network. A closing chain is an undirected sequence of links,‡ currently in the basis, leading from the demand node to the supply node of the candidate link. Figure 7.11 demonstrates the closing chain based on candidate link 2→4. This closing chain includes node 2, node 6, node 3, and node 4.

To determine the amount of flow to be assigned to the candidate link, let the links along the closing chain be flagged as follows. Start from the supply node of the candidate link and traverse this link setting its flag to "T". Continue along the closing chain setting a link flag to "F" when going against the link direction and to "T" when going with the link direction. In the example shown in Figure 7.11, links 2→4 and 3→6 are flagged "T" while links 3→4 and 2→6 are flagged "F".

The amount of flow to be assigned to the candidate link is the smallest link flow of all the links with flag "F" included in the closing chain. In the example shown in Figure 7.11, this amount is min {10, 100} = 10 flow units.

‡In other words, the closing chain is determined with no regard to the directionality of each link.

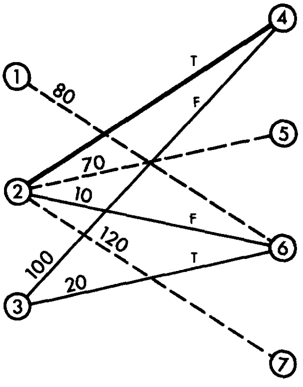


Figure 7.11 Closing chain (based on link 2→ 4) and the flags on the closing chain links.

The link with the minimum flow (2→ 6 in this example) will be emptied and leave the basis in the next step.

In order to identify a closing chain, note that the links in the basis always form a special graph structure (known as a "spanning tree"), in which there is a unique sequence of links connecting any two nodes (with no regard to link directionality). Starting with an arbitrary "root" node, each node can therefore be associated with a predecessor node. The list of predecessor nodes is similar to the one used in the representation of minimum-path trees described in conjunction with the shortest path algorithm in Section 6.4. Given a basis, the list of predecessors can be created in a fashion similar to the computation of the dual variables in step 2. Starting, for example, with node 1, the predecessor list for the basis shown in Figure 7.10 is the following:

Node	1	2	3	4	5	6	7
Predecessor	Root	6	6	3	2	1	2

The values of the predecessor nodes and the dual variables can be determined simultaneously by following the same progression.

Given the predecessor list, the closing chain resting on a given candidate link, $i \rightarrow j$, is found by tracing back the predecessors from node i and from node j until a common node is found.† In the example shown in Figure 7.11, the common node (known as the "pivot" node) is node 6. The closing chain includes all nodes traced from i and from j up to the pivot node.

By the end of step 3, a closing chain has been identified, the amount of flow to be reassigned has been determined, and the link leaving the basis has been marked.

†This tracing process can be expedited significantly by measuring the "depth" of each node from the root, along the spanning tree. The depth measures the number of links that have to be traversed to reach the node order consideration from the root. The usefulness of this pointer is that it can be used to limit the search since the pivot node must lie in the same depth for both branches of the spanning tree that are traced back.

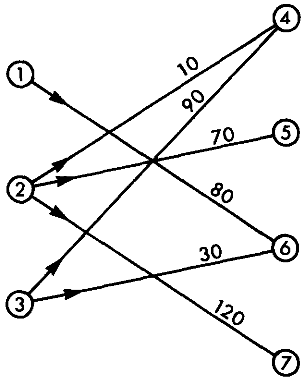
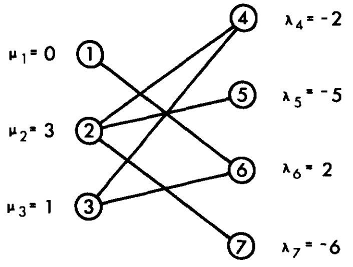


Figure 7.12 Optimal solution of the transportation problem example.

Step 4. In step 4 the flow is actually assigned to the candidate link and all other flows are adjusted accordingly. By inspecting the example, the reader can see that it is enough to adjust the flows only on the links in the closing chain. This can be easily done by adding the amount of flow identified in step 3 to all links flagged as "T" and subtracting this amount of flow from all links flagged as "F".

In the example shown in Figure 7.11, 10 units of flow are added to link 2→4, subtracted from 3→4, added to 3→6, and subtracted from 2→6. The new flow assignment is shown in Figure 7.12 with link 2→6 not included in the basis any more.

Note that only one link should leave the basis in step 4, even if the flow adjustment process causes other links in the basis to carry no flow. This ensures that the basis always includes $I + J - 1$ links.



Reduced costs :

TO / FROM	4	5	6	7
1	4	6	▨	12
2	▨	▨	7	▨
3	▨	9	▨	9

Figure 7.13 Dual variables and reduced costs at the optimal solution point.

The last function executed in step 4 is the updating of the dual variables values and the predecessor list. When step 4 is completed, step 2 is executed again. Figure 7.13 shows the values of the dual variables and the reduced costs associated with the basis shown in Figure 7.12. Since all the reduced costs are positive, the algorithm terminates. The value of the objective function at optimality is $z(\mathbf{x}^*) = -330$.

The algorithm, including some of the list-processing methods mentioned in earlier footnotes, is extremely fast and efficient. It can be used to determine the flows in a large network, including many thousands of links, in a fraction of a second of computer time.

7.5 SUMMARY

This chapter examined the joint distribution/assignment problem where link flows and O–D flows are determined simultaneously. The problem is different from the variable-demand equilibrium assignment discussed in Chapter 6 in that some restrictions are placed on the O–D flows.

The first two sections deal with joint distribution/assignment models for which the total number of trips leaving each origin node is fixed and known. Such models are known as singly constrained. Section 7.1 assumes that the trips are distributed from each origin to the various destinations on the basis of a user equilibrium criterion. Even though such a model is not realistic, it serves to introduce the problem and the solution approach, and to motivate the other models. Section 7.2 assumes that the O–D flows are determined by a special trip distribution function. In both cases the basic network is assumed to be in user equilibrium, and in both cases the problem can be solved by using a fixed-demand equilibration algorithm based on a proper network representation. The problem discussed in Section 7.2 can also be tackled with an efficient two-phase model which is similar to the convex combinations algorithm except that its descent direction is different.

Section 7.3 presents a formulation of a doubly constrained distribution model. In this model, not only the total number of trips leaving each origin node is known, but the total flow into each destination node is fixed and known as well. The problem can be solved, again, with the convex combinations algorithm. The direction finding step of this algorithm requires the solution of Hitchcock's transportation problem, an algorithmic solution procedure to which is given in Section 7.4. In a fashion similar to the solution of the singly constrained model, the direction-finding step of the doubly constrained model can be accomplished by a double-phase algorithm. Such an algorithm would require the solution of a gravity model (instead of a Hitchcock transportation problem) at every iteration.

The choice between singly and doubly constrained formulations is a modeling issue that is left for the analyst. This choice depends on the type of questions for which the model is supposed to provide an answer. In a doubly constrained model, the total number of trips attracted to each destination should be predicted exogenously, while in a singly constrained model, this number is one of the outputs of the analysis.

7.6 ANNOTATED REFERENCES

Most of the literature on the subject of the combined distribution/assignment program focuses on the gravity and entropy distribution models. A review of these types of models is given by Wilson (1970), who originally developed the applications of the entropy concept to urban and regional planning problems. The formulation of the combined distribution/assignment model was studied by Bruynooghe (1968), Florian et al. (1975), and Evans (1976). The solution approach given in Section 7.3 was used in a somewhat more general context by Florian and Nguyen (1978), Safwat and Magnanti (1982), and LeBlanc and Abdulaal (1982). The doubly constrained model formulation given in Section 7.3 and the double-stage solution algorithm were suggested by Evans (1976), who proved convergence of this algorithm. Murchland (1969) suggested a similar (double-stage) algorithm for the variable-demand problem and proved its convergence.

The algorithm for solving Hitchcock's transportation problem given in Section 7.4 is an adaptation of the simplex method for linear programming. This algorithm can be found in any standard textbook on operations research methods or network flows. For example, Wagner (1975) has a good discussion at an introductory level on the topic whereas Glover et al. (1974) give a network-oriented treatment to the problem, including implementation consideration. The example used in explaining the algorithm is taken from Cedar (1978).

7.7 PROBLEMS

7.1. Formulate a singly constrained UE-type distribution/assignment program with $M_s = 0, \forall\ s$. Compare to program [3.3] and comment on the differences between the formulations.

7.2. **(a)** Find the UE-type equilibrium flow for the network shown in Figure 7.3. The link performance functions are $t_1 = 3 + x_1$ and $t_2 = 2x_2$. Assume that $x_1 + x_2 = 6$ and $M_1 = M_2 = 0$.

(b) Find the equilibrium flows when the attraction of destination 1 (the top one) is 3 travel-time units (i.e., $M_1 = 3$, $M_2 = 0$).

7.3. Find the equilibrium flows in the example of Problem 7.2a if the flow distribution follows a logit model with parameter $\gamma = 0.5$. Then find the equilibrium if the attraction of destination 1 is 3 time units (as in Problem 7.2b). Comment on the differences between the solutions to this problem and the solutions of Problem 7.2.

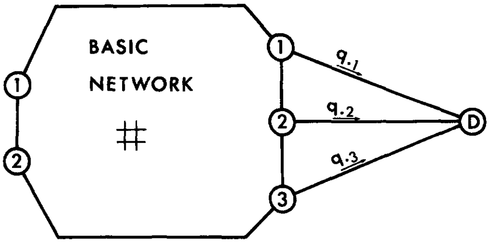


Figure P7.1

7.4. Consider the network representation of the joint UE distribution/assignment problem shown in Figure P7.1. In this augmented network there is only one dummy "supernode" D. Each destination is connected to D by a dummy link $s \rightarrow D$, the flow on which is $q_{\cdot s} = \sum_r q_{rs}$ and the travel cost on which is $-M_s$. Show that if the flow between each origin r and node D is O_r, the UE solution of this augmented network is equivalent to a joint UE distribution/assignment solution for the basic network. Compare this formulation with the one depicted in Figure 7.2.

7.5. **(a)** Write an equivalent minimization formulation for a UE-type distribution/assignment problem where each destination, s, is associated with a parking charge which can be expressed in time-unit equivalents as T_s. What are the first-order conditions for this problem? Interpret these conditions and the meaning of the dual variables.

(b) Assume that in addition to a parking charge, each destination is associated with a fixed parking capacity (including on-street and in lots). How would you model the problem now? Discuss the formulation of the equivalent problem, including possible difficulties. Show the equivalent network representation of the problem.

7.6. A trip generation model relates the number of trips originating at each centroid node r, O_r, to the level of service experienced by travelers out of this node. How would you model a UE-based combined trip generation, distribution, and assignment problem? (*Hint:* The answer lies in a proper definition of the attraction variables.) Can you think of an augmented network representation so that the problem can be solved with a UE algorithm?

7.7. Assume that a logit trip distribution model is specified in terms of (constant) attraction variables only. In other words,

$$q_{rs} = O_r \frac{e^{M_s}}{\sum_m e^{M_m}} \qquad \forall\ r, s$$

(a) Formulate an equivalent minimization program for the associated distribution/assignment program.

(b) Describe an algorithmic solution procedure.

(c) Why not use this instead of the logit model given in Eq. [7.14]?

7.8. **(a)** Explain the mechanics of the direction-finding step associated with the application of the convex combinations algorithm to the solution of the logit distribution/assignment program. In other words, show how program [7.19] is solved to obtain the auxiliary link and O–D flows.

(b) Outline the application of the convex combinations algorithm for this problem. In particular, pay attention to the initialization procedure and the convergence test.

7.9. **(a)** Show that the UE problem over the extended network given in Figure 7.2 with the volume delay functions given by Eq. [7.22] is equivalent to a distribution/assignment problem with logit distribution functions.

(b) Show that the UE conditions over this modified network are equivalent to the equilibrium conditions for the logit distribution/assignment problem.

***7.10.** Consider the double-stage algorithm discussed at the end of Section 7.2.

(a) Show that the auxiliary program solved in the direction-finding step (program [7.28]) is convex.

(b) Show that the double-stage algorithm generates a descent direction for program [7.15] at every iteration. [*Hint:* Show that $\nabla z(\mathbf{x}^n, \mathbf{q}^n) \cdot (\mathbf{y}^n - \mathbf{x}^n, \mathbf{v}^n - \mathbf{q}^n)^T < 0$ for $z(\cdot, \cdot)$ in [7.15a], where y^n and $\mathbf{v}^n$ are generated by the direction-finding step of the double-stage algorithm. In other words, show that

$$\sum_a t_a(x_a^n)(y_a^n - x_a^n) + \sum_{rs} \left(\frac{1}{\gamma} \ln q_{rs}^n - M_s \right)(v_{rs}^n - q_{rs}^n) < 0$$

Use the facts that in program [7.28] $\hat{z}(\mathbf{y}^n, \mathbf{v}^n) - \hat{z}(\mathbf{x}^n, \mathbf{q}^n) < 0$ if $(\mathbf{x}^n, \mathbf{q}^n)$ is not the optimum and that $\hat{z}(\mathbf{y}, \mathbf{v})$ is convex.]

***7.11.** Develop and outline a double-stage algorithm for solving the variable-demand UE program discussed in Chapter 6. Develop the auxiliary program solved at every iteration of the method; show that it is convex and that its solution is the auxiliary flow pattern used to define the descent direction. Also show that the search direction is, in fact, a descent direction.

7.12. Consider the problem

$$\min z(\mathbf{x}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\ r, s$$

$$\sum_s q_{rs} = O_r \qquad \forall\ r$$

$$\sum_r q_{rs} = D_s \qquad \forall\ s$$

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s$$

and the incidence relationships. How would you solve this problem? Does it correspond to any kind of reasonable equilibrium conditions over a transportation network? Explain your answers.

7.13. Interpret the coefficients A_r and B_s of the entropy model.

7.14. Derive the Lagrangian of program [7.38]. Show that the first-order conditions of program [7.38] include the UE equations as well as Eq. [7.39a] which leads to the entropy distribution model.

7.15. The doubly constrained distribution/assignment model (the equivalent minimization of which is given in Eqs. [7.38]) can be solved by using the double-stage algorithm.

(a) Describe the application of the double-stage algorithm to this problem in detail.

(b) Show that the auxiliary program that has to be solved at each direction-finding step is convex.

(c) Show that the objective function $\sum_a t_a^n y_a^n + (1/\zeta) \sum_{rs} (v_{rs} \ln v_{rs} - v_{rs})$, when minimized subject to constraints [7.38b] through [7.38e] (written in terms of the appropriate auxiliary variables), results in the same solution as the procedure of finding $u_{rs}^n$and minimizing Eq. [7.46].

(d) Suggest an algorithm for solving the program formulated in (c) (or for solving [7.46] subject to the appropriate constraints).

7.16. Consider a simple network, including two links connecting one O–D pair. The following data is given:

$$t_1 = 2.0\left[1 + 0.15\left(\frac{x}{12}\right)^4\right]$$

$$t_2 = 3.0\left[1 + 0.15\left(\frac{x}{8}\right)^4\right]$$

$$q = \frac{72}{t} - 1$$

(a) Use the convex combinations algorithm to solve for the equilibrium with variable demand. Show the results of five iterations.

(b) Use the double-stage algorithm for the same problem. Show the calculations. Compare the rate of convergence of both algorithms.

7.17. Formulate the equivalent minimization for a singly constrained entropy-based distribution/assignment problem. What do you get?

7.18. Consider the application of the convex combinations method to the solution of the doubly constrained distribution/assignment problem. Explain why the number of destinations loaded from each origin at each iteration is, on the average, two or higher.

7.19. Consider the following link choice mechanism in the initialization procedure for the Hitchcock transportation problem: Choose supply nodes in numerical order and examine the links emanating from each supply node in order of least cost.

(a) Write a flowchart of the initialization procedure with this link choice mechanism.

(b) Apply this mechanism to the network example in Figure 7.6 to get an initial feasible solution.

(c) Compare the objective function value at the initial solution obtained with this procedure to that obtained with the penalty method. Comment on the relative merits of the two procedures.

7.20. Consider the transportation problem example solved in Section 7.4. Interpret the meaning of the dual variables values at the optimal solution.

7.21. (*For those who know linear programming*) Show how the algorithm described in Section 7.4 for solving the transportation problem corresponds to the simplex method for solving LPs.

7.22. Use the algorithm described in Section 7.4 to solve a transportation problem with the following parameters:

$$\begin{array}{lllll}
O_1 = 100 & D_4 = 75 & c_{14} = 3 & c_{24} = 2 & c_{34} = 0 \\
O_2 = 130 & D_5 = 125 & c_{15} = 6 & c_{25} = 12 & c_{35} = 4 \\
O_3 = 70 & D_6 = 100 & c_{16} = 5 & c_{26} = 1 & c_{36} = 2
\end{array}$$

8

Equilibrium with Link Interactions

The mathematical programming formulations presented up to this point in the book assume that link performance functions are independent of each other. In other words, it is assumed that the travel time on a given link depends only on the flow through that link and not on the flow through any other link. This assumption is not always valid. Cases where flow interaction cannot be ignored include heavy traffic in two-way streets, unsignalized intersections, and left-turning movements in signalized intersections. This chapter presents methods for finding equilibrium when the interactions of flows and travel times across links are recognized explicitly.

Link interactions can be either symmetric or asymmetric. When the interactions are symmetric, the marginal effect of one link flow, say x_a, on the travel time on any other link, say t_b, is equal to the marginal effect of x_b on t_a. In this case, the equilibrium flow pattern can be found by applying the equivalent minimization approach utilized in the previous chapters. Section 8.1 presents this method through a special case of pairwise symmetric link intersections and then generalizes the results to any symmetric interaction pattern.

When the link interactions are asymmetric, there is no known equivalent minimization program that can be used to find the equilibrium flow pattern. Section 8.2 explains this case and outlines an algorithmic approach for finding the equilibrium flow pattern under these conditions.

The focus of this chapter is on the standard (fixed-demand) UE model presented in Chapter 3. The resulting methods are applicable to all other models that can be solved as equivalent UE programs, using appropriately modified networks.

8.1 TWO-WAY TRAFFIC INTERACTIONS

A simple case of link interaction involves pairwise relationships between the two (opposite direction) links representing two-way streets.† As traffic builds up in one direction, the delay to flow in the opposite direction increases due, for example, to the reduction in passing opportunities. This effect can become quite pronounced in moderate to heavy flow.

To model this interaction, let a' denote the link opposite to link a. [Since the prime on a subscript represents the opposite link, link $(a')'$ is link a itself.] The performance function on link a is given by

$$t_a = t_a(x_a, x_{a'}) \qquad [8.1a]$$

and obviously,

$$t_{a'} = t_{a'}(x_{a'}, x_a) \qquad [8.1b]$$

The travel time in a given direction is considered to be a function of the flow in both directions. This travel time, however, is assumed to be independent of any other link flow in the network.

In the case of two-way traffic, the interaction between the opposite direction links can be assumed to be symmetric. This assumption means that the effect of an additional flow unit along a particular link on travel time in the opposing direction equals the effect of an additional flow unit in the opposing direction on the travel time of the link under consideration. The symmetry condition can be expressed mathematically as

$$\frac{\partial t_a(x_a, x_{a'})}{\partial x_{a'}} = \frac{\partial t_{a'}(x_{a'}, x_a)}{\partial x_a} \qquad \forall\ a \qquad [8.2]$$

The equilibrium flow pattern can be found in this case by solving the following minimization program:

$$\min z(\mathbf{x}) = \frac{1}{2} \sum_a \left(\int_0^{x_a} t_a(\omega, x_{a'})\, d\omega + \int_0^{x_a} t_a(\omega, 0)\, d\omega \right) \qquad [8.3a]$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\ r, s \qquad [8.3b]$$

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [8.3c]$$

The objective function [8.3a] includes two terms for each link in the network. The first is the integral of the link's performance function, with respect to its own flow, when the flow in the opposite direction is held constant. The second term is the integral of the link's performance function when the flow on the

†The analysis in this section applies to any pairwise symmetric link interaction pattern, not only in two-way streets.

opposite link is held at zero. The objective function is halved so that the scale of the results will conform to the results of earlier chapters (this point is explained below). This section investigates program [8.3], establishing the equivalency between the solution of this program and the UE equations. It also determines the conditions for the uniqueness of this solution and describes a convergent algorithm for solving this program. Finally, the two-way traffic case is extended to general cases of symmetric link interactions.

Equivalency Conditions

The first-order conditions of program [8.3] involve the first derivatives of the objective function. The derivative of $z(\mathbf{x})$ with respect to the flow on a particular link (say, link b) is computed by differentiating, in turn, each term in Eq. [8.3a].

The derivative of the first sum in [8.3a] with respect to x_b is given by

$$\frac{\partial}{\partial x_b}\left[\sum_a \int_0^{x_a} t_a(\omega, x_{a'})\, d\omega\right]$$

$$= \frac{\partial}{\partial x_b}\int_0^{x_b} t_b(\omega, x_{b'})\, d\omega + \frac{\partial}{\partial x_b}\int_0^{x_{b'}} t_{b'}(\omega, x_b)\, d\omega \qquad [8.4]$$

The derivative of all other terms in the sum will be zero. Expression [8.4] includes two derivatives. The first one can be readily evaluated (by the fundamental theorem of calculus), that is,

$$\frac{\partial}{\partial x_b}\int_0^{x_b} t_b(\omega, x_{b'})\, d\omega = t_b(x_b, x_{b'}) \qquad [8.5]$$

The second term in Eq. [8.4] can be evaluated by moving the derivative inside the integral (since x_b is independent of $x_{b'}$) and the remaining problem is to evaluate

$$\int_0^{x_{b'}} \frac{\partial t_{b'}(\omega, x_b)}{\partial x_b}\, d\omega \qquad [8.6]$$

To get an expression that is easy to evaluate, the symmetry condition (Eq. [8.2]) can be introduced into the integrand of Eq. [8.6], that is,

$$\int_0^{x_{b'}} \frac{\partial t_{b'}(\omega, x_b)}{\partial x_b}\, d\omega = \int_0^{x_{b'}} \frac{\partial t_b(x_b, \omega)}{\partial x_{b'}}\, d\omega \qquad [8.7]$$

The integrand on the right-hand side of Eq. [8.7] cannot be taken out of the integral sign, since the derivative is taken with respect to the boundary of the integral. The integrand is then a function, and Eq. [8.7] has to be evaluated as a definite integral, that is,

$$\int_0^{x_{b'}} \frac{\partial t_b(x_b, \omega)}{\partial x_{b'}}\, d\omega = t_b(x_b, x_{b'}) - t_b(x_b, 0) \qquad [8.8]$$

The results shown in expressions [8.5] and [8.8] can be substituted now in Eq. [8.4] to give

$$\frac{\partial}{\partial x_b}\left[\sum_a \int_0^{x_a} t_a(\omega, x_{a'})\, d\omega\right] = 2t_b(x_b, x_{b'}) - t_b(x_b, 0) \qquad [8.9]$$

The derivative of the second term in objective function [8.3a] with respect to x_b is the following:

$$\frac{\partial}{\partial x_b}\left[\sum_a \int_0^{x_a} t_a(\omega, 0)\, d\omega\right] = \frac{\partial}{\partial x_b}\int_0^{x_b} t_b(\omega, 0)\, d\omega$$

$$= t_b(x_b, 0) \qquad [8.10]$$

Using the results shown in Eqs. [8.9] and [8.10], the derivative of the objective function, $z(\mathbf{x})$ in Eq. [8.3a], can be written as follows:

$$\frac{\partial z(\mathbf{x})}{\partial x_b} = \frac{1}{2}\frac{\partial}{\partial x_b}\left[\sum_a \int_0^{x_a} t_a(\omega, x_{a'})\, d\omega + \sum_a \int_0^{x_a} t_a(\omega, 0)\, d\omega\right]$$

$$= \frac{1}{2}\,[2t_b(x_b, x_{b'}) - t_b(x_b, 0) + t_b(x_b, 0)] = t_b(x_b, x_{b'}) \qquad [8.11]$$

Thus the derivative of the objective function with respect to the flow on a particular link is the travel time on that link. This result is similar to the one obtained in Chapter 3 for the standard UE program. The first-order conditions can easily be derived from this result by repeating the derivation in Eqs. [3.11] through [3.16]. These conditons are

$$(c_k^{rs} - u_{rs})f_k^{rs} = 0 \qquad \forall\ k, r, s \qquad [8.12a]$$

$$c_k^{rs} - u_{rs} \geq 0 \qquad \forall\ k, r, s \qquad [8.12b]$$

These first-order conditions mean that the flow pattern that minimizes program [8.3] also satisfies the user-equilibrium conditions for a network with two-way link interactions.

Two comments are in order here. First, the aforementioned conclusion is true only if the interaction is symmetric, that is, if the marginal effect of x_a on $t_{a'}$ is equal to the marginal effect of $x_{a'}$ on t_a, for every pair of links a and a' representing opposite flow directions. (This condition was used explicitly in Eq. [8.7] to derive the first-order conditions of program [8.3].) Furthermore, it is assumed that no other link interactions exist. In this context, the symmetry assumption may be plausible because of the symmetry of the physical phenomenon to which it is applied.

The second comment has to do with the derivation of the first-order conditions in this section. This derivation highlights the nature of objective function [8.3a] and of most of the equivalent mathematical programming formulations presented in this text. These programs are carefully constructed so that their first-order conditions yield the required equilibria. The first sum in Eq. [8.3a] parallels the UE objective function [3.3a]. The added sums are used to cancel out the terms resulting from the lower bound of the integrals. Such mathematical constructs are not generally based on any economic (or other behavioral) theories, and should be viewed strictly as tools for solving the equilibrium equations.

Uniqueness Conditions

The equivalent mathematical program given in Eqs. [8.3] has a unique solution under the following conditions:

1. The travel time on a link is a strictly increasing function of the flow on that link, that is,

$$\frac{\partial t_a(x_a, x_{a'})}{\partial x_a} > 0 \qquad \forall\, a \tag{8.13a}$$

2. An additional flow unit in a given direction will affect the travel time in that direction more than it would affect the travel time in the opposite direction. In other words,

$$\frac{\partial t_a(x_a, x_{a'})}{\partial x_a} > \frac{\partial t_{a'}(x_{a'}, x_a)}{\partial x_a} \qquad \forall\, a \tag{8.13b}$$

Actual link performance functions tend to satisfy both of these conditions. [Note that both $\partial t_a(\cdot, \cdot)/\partial x_a$ and $\partial t_{a'}(\cdot, \cdot)/\partial x_a$ are nonnegative for actual link performance functions.]

Using these two conditions, it can now be shown that objective function [8.3a] is strictly convex and therefore (since the constraint set is convex) that its minimum is unique. Any one of the convexity criteria mentioned in Section 2.2 can be used for this purpose. In this case, it is relatively easy to use the Hessian of $z(\mathbf{x})$ [which is the matrix of second derivatives of $z(\mathbf{x})$]. Showing that this Hessian is positive definite is sufficient for demonstrating that $z(\mathbf{x})$ is strictly convex.

To calculate the second derivatives of $z(\mathbf{x})$, recall first that (from Eq. [8.11]) the first derivative of $z(\mathbf{x})$ with respect to the flow on link a is given by

$$\frac{\partial z(\mathbf{x})}{\partial x_a} = t_a(x_a, x_{a'}) \tag{8.14}$$

The second derivative, with respect to the flows on links a and b, is then given by

$$\frac{\partial^2 z(\mathbf{x})}{\partial x_a \partial x_b} = \frac{\partial t_a(x_a, x_{a'})}{\partial x_b} = \begin{cases} \dfrac{\partial t_a(x_a, x_{a'})}{\partial x_a} & \text{if } b = a \\[2ex] \dfrac{\partial t_a(x_a, x_{a'})}{\partial x_{a'}} & \text{if } b = a' \\[2ex] 0 & \text{otherwise} \end{cases} \tag{8.15}$$

Assume, now, that the network links are ordered so that each link is adjacent to the link representing the opposite traffic movement. Given this order, the Hessian of $z(\mathbf{x})$, $\nabla^2 z(\mathbf{x})$, can be expressed as†

$$\nabla^2 z(\mathbf{x}) = \begin{bmatrix} \nabla^2_{1,1'} & \mathbf{0} & \cdots & \cdot & \cdots \\ \mathbf{0} & \nabla^2_{2,2'} & & \cdot & \\ \vdots & & \ddots & & \\ \cdot & \cdot & & \nabla^2_{a,a'} & \\ \vdots & & & & \ddots \end{bmatrix} \qquad [8.16a]$$

where

$$\nabla^2_{a,a'} = \begin{bmatrix} \dfrac{\partial t_a}{\partial x_a} & \dfrac{\partial t_{a'}}{\partial x_a} \\[2ex] \dfrac{\partial t_a}{\partial x_{a'}} & \dfrac{\partial t_{a'}}{\partial x_{a'}} \end{bmatrix} \qquad \forall\ a \qquad [8.16b]$$

The structure of the Hessian is "block diagonal," with each block given by the 2 × 2 matrix in Eq. [8.16b]. Each of these (symmetric) 2 × 2 matrices is positive definite if its diagonal entries and its determinant are positive.‡ To see that this is the case with $\nabla^2_{a,a'}$ in Eq. [8.16b], note first that

$$\frac{\partial t_a}{\partial x_a} > 0 \quad \text{and} \quad \frac{\partial t_{a'}}{\partial x_{a'}} > 0 \qquad [8.17a]$$

due to assumption [8.13a]. Second, note that the determinant of each block matrix is positive, that is,

$$\begin{vmatrix} \dfrac{\partial t_a}{\partial x_a} & \dfrac{\partial t_{a'}}{\partial x_a} \\[2ex] \dfrac{\partial t_a}{\partial x_{a'}} & \dfrac{\partial t_{a'}}{\partial x_{a'}} \end{vmatrix} = \frac{\partial t_a}{\partial x_a}\frac{\partial t_{a'}}{\partial x_{a'}} - \frac{\partial t_a}{\partial x_{a'}}\frac{\partial t_{a'}}{\partial x_a} > 0 \qquad [8.17b]$$

This inequality holds because of the second assumption (Eq. [8.13b]) on the link performance functions. Thus, Hessian [8.16a] is a block diagonal matrix in which each block is a positive-definite matrix. Such a Hessian is positive definite, as can be shown by the reader (see Problem 8.4).

The objective function is thus strictly convex with respect to the link flows and consequently, it can be concluded that program [8.3] has a unique minimum. As was the case with the standard UE problem, the solution of this program is not unique with respect to path flows, but only with respect to link flows.

†In Eq. [8.16a] **0** stands for a 2 × 2 matrix of zeros.

‡In Section 2.2 this condition was stated more generally, saying that a symmetric matrix is positive definite if all the leading principal minors are positive.

Algorithm

A number of algorithms can be used to minimize program [8.3]. The convex combinations method is again particularly applicable to this problem because the direction-finding step of this algorithm involves an all-or-nothing assignment, which is relatively easy to implement and solve.

The linear program that has to be solved at the direction-finding step involves the following objective function (see Eq. [8.14]):

$$\min z^n(\mathbf{y}) = \sum_a \frac{\partial z(\mathbf{x}^n)}{\partial x_a} y_a = \sum_a t_a^n y_a \qquad [8.18]$$

where $t_a^n = t_a(x_a^n, x_{a'}^n)$. This function should be minimized subject to the flow conservation constraints (in $\mathbf{y}$). The solution of this minimization program, $\{y_a^n\}$, which is found by a simple all-or-nothing assignment, is used to define a descent direction. The optimal move size in this direction is a scalar, α_n, which solves the following problem:

$$\min_{0 \leqslant \alpha \leqslant 1} z(\alpha) = \sum_a \int_0^{x_a^n + \alpha(y_a^n - x_a^n)} t_a[\omega, x_{a'}^n + \alpha(y_{a'}^n - x_{a'}^n)]\, d\omega$$

$$+ \sum_a \int_0^{x_a^n + \alpha(y_a^n - x_a^n)} t_a(\omega, 0)\, d\omega \qquad [8.19]$$

This problem is no more difficult to solve than the original UE objective function. All other algorithmic steps are similar to the ones described in Section 5.2 for solving the standard UE program.

Program [8.3] is applicable to any paired and symmetric interaction pattern. In other words, it holds whenever the travel time on a given link is a function of the flow on that link, and a single additional link. The travel time on that additional link is similarly influenced only by its own flow and the flow on the aforementioned link. Furthermore, the travel time on these two links is not influenced by the flow on any other link. This pair of links does not necessarily have to represent opposite flow directions as long as the symmetry holds. The following example illustrates the equivalent minimization approach for a simple equilibration problem of this type.

Example

Consider the simple three-link network depicted in Figure 8.1a. The intersection between the streets represented by links 1 and 2 is illustrated in Figure 8.1b. With this type of intersection, it can be expected that delays to traffic on link 2 would be influenced by the flow on link 1, and vice versa. Due to the geometry of the intersection, this interaction can be expected to be symmetric.

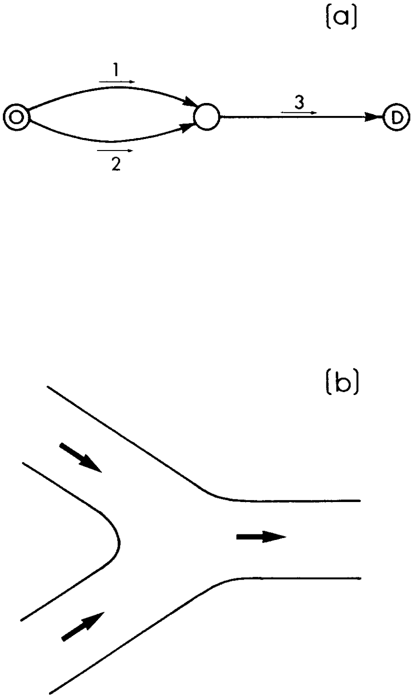


Figure 8.1 Network example with three links and one O–D pair: (a) the network representation; (b) the intersection represented by the middle node.

The link performance equations for this problem are the following:

$$t_1(x_1, x_2) = 2 + 3x_1 + 2x_2 \qquad [8.20a]$$

$$t_2(x_2, x_1) = 4 + 2x_2 + 2x_1 \qquad [8.20b]$$

$$t_3(x_3) = 3 + x_3 \qquad [8.20c]$$

Assume further that there are 5 units of flow going from the origin to the destination. In other words, the flow conservation constraint is $x_1 + x_2 = 5$ flow units.

In determining the equilibrium flows in this example, link 3 can be dropped from the problem, since it is clear that $x_3^* = 5$ and thus $t_3(x_3^*) = 8$ time units. The problem that remains is to find the distribution of flow between link 1 and link 2. Assuming that both links will be used at equilibrium, the solution to this example can be found by setting

$$t_1(x_1, x_2) = t_2(x_2, x_1) \qquad [8.21]$$

Substituting $x_2 = 5 - x_1$ into the link performance functions [8.20a] and [8.20b], Eq. [8.21] can be solved to obtain $x_1^* = 2$ flow units and, accordingly, $x_2^* = 3$ flow units, meaning that both links are, in fact, used at equilibrium. The travel times at equilibrium are then

$$t_1(2, 3) = t_2(3, 2) = 14$$

Before the equivalent minimization approach can be used for solving this problem, the symmetry conditions should be verified. The equivalent minimization approach is not valid if the link interaction pattern is not symmetric. In this example,

$$\frac{\partial t_1(x_1, x_2)}{\partial x_2} = \frac{\partial t_2(x_2, x_1)}{\partial x_1} = 2 \qquad [8.22]$$

meaning that the link interaction is symmetric.

The equivalent mathematical program for this example is given by Eqs. [8.3], as follows:

$$\min z(x_1, x_2, x_3) = \tfrac{1}{2}\int_0^{x_1} (2 + 3\omega + 2x_2)\, d\omega + \tfrac{1}{2}\int_0^{x_2} (4 + 2\omega + 2x_1)\, d\omega$$

$$+ \tfrac{1}{2}\int_0^{x_1} (2 + 3\omega)\, d\omega + \tfrac{1}{2}\int_0^{x_2} (4 + 2\omega)\, d\omega + \int_0^{x_3} (3 + \omega)\, d\omega \qquad [8.23a]$$

subject to

$$x_1 + x_2 = 5 \qquad [8.23b]$$

$$x_1 + x_2 = x_3 \qquad [8.23c]$$

$$x_1, x_2 \geq 0 \qquad [8.23d]$$

The Hessian of the objective function is given by

$$\nabla^2 z(\mathbf{x}) = \begin{bmatrix} 3 & 2 & 0 \\ 2 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which is positive definite. (Check it.) The solution of program [8.23] will therefore be unique.

Note that because of constraint [8.23c], the last term of the objective function can be dropped; it is a constant that does not affect the minimization. After carrying out the integration, then, the objective function of program [8.23] becomes

$$z(x_1, x_2) = \tfrac{1}{2}(2x_1 + \tfrac{3}{2}x_1^2 + 2x_2 x_1 + 4x_2 + x_2^2 + 2x_1x_2 + 2x_1 + \tfrac{3}{2}x_1^2 + 4x_2 + x_2^2)$$

Substitution of $x_2 = 5 - x_1$ into this function and collection of terms results in

$$z(x_1) = 2x_1 + \tfrac{3}{2}x_1^2 + 2x_1(5 - x_1) + 4(5 - x_1) + (5 - x_1)^2$$

The minimum is found by differentiating and finding the root of the resulting equation, that is, by setting

$$\frac{dz(x_1)}{dx_1} = 2 + 3x_1 + 2(5 - x_1) - 2x_1 - 4 - 2(5 - x_1) = 0$$

This equation implies that $x_1^* = 2$ and consequently $x_2^* = 3$. The solution of the minimization program is therefore the equilibrium flow pattern, as expected.

Extensions

As mentioned in the introduction of this chapter, interactions among traffic flows can occur not only on two-way streets and between pairs of links but in many other cases as well. The travel time on a particular link may therefore be a function of the flow on several links, meaning that, in general, $t_a = t_a(\dots, x_a, \dots)$, or $t_a = t_a(\mathbf{x})$. In cases where the link interaction pattern is symmetric, the equilibrium problem can be formulated and solved by using the equivalent minimization approach. The symmetry conditions on link interaction can be stated mathematically as

$$\frac{\partial t_a(\mathbf{x})}{\partial x_b} = \frac{\partial t_b(\mathbf{x})}{\partial x_a} \qquad \forall\ a \neq \mathrm{b} \tag{8.24}$$

a condition that is a generalization of Eq. [8.2]. Note that when no link interaction exists, condition [8.24] holds as well since all these derivatives will be zero.

Condition [8.24] can be stated as a requirement that the *Jacobian* matrix of the link-travel-time vector be symmetric. The Jacobian (of a vector of scalar functions) is formed by arranging the derivatives of all these functions, with respect to all the arguments, in matrix form. Letting $\mathbf{t}(\mathbf{x}) = (\dots, t_a(\mathbf{x}), \dots)$, the Jacobian of $\mathbf{t}(\mathbf{x})$, which is denoted by $\nabla\mathbf{t}(\mathbf{x})$ or $\nabla_{\mathbf{x}}\mathbf{t}$,† includes the partial derivatives of all the link travel times with respect to all link flows. In other words,

$$\nabla_{\mathbf{x}}\mathbf{t} = \begin{bmatrix} \dfrac{\partial t_1(\mathbf{x})}{\partial x_1} & \dfrac{\partial t_2(\mathbf{x})}{\partial x_1} & \cdots & \dfrac{\partial t_a(\mathbf{x})}{\partial x_1} & \cdots \\ \dfrac{\partial t_1(\mathbf{x})}{\partial x_2} & \dfrac{\partial t_2(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial t_a(\mathbf{x})}{\partial x_2} & \cdots \\ \vdots & \vdots & \ddots & & \\ \dfrac{\partial t_1(\mathbf{x})}{\partial x_a} & \dfrac{\partial t_2(\mathbf{x})}{\partial x_a} & & \dfrac{\partial t_a(\mathbf{x})}{\partial x_a} & \\ \vdots & \vdots & & & \ddots \end{bmatrix} \tag{8.25}$$

Condition [8.24] means that this Jacobian is symmetric.

As mentioned above, in cases in which the symmetry condition holds, an equivalent mathematical program, the solution of which satisfies the UE con-

†The operator "del" (∇) is used to denote the gradient vector, if it operates on a scalar function. In cases in which ∇ operates on a vector-valued function it represents the Jacobian of this function.

ditions, can be found. The objective function of this program is given by $z(\mathbf{x})$, defined by the line integral†

$$z(\mathbf{x}) = \int_0^{\mathbf{x}} \mathbf{t}(\boldsymbol{\omega})\, d\boldsymbol{\omega} \qquad [8.26]$$

The proof of this statement is given in the references mentioned in Section 8.4 (see also Problem 8.5). The Hessian of this objective function will be the Jacobian in [8.25], implying that this objective function will have a unique solution if the Jacobian is positive definite.

The conditions for having a unique equilibrium solution in the general case of link interactions can be stated as an extension of the uniqueness conditions for pairwise link interaction (see Eqs. [8.13]). These conditions are:

1. The travel time on each link is an increasing function of the flow on that link, that is,

$$\frac{\partial t_a(\mathbf{x})}{\partial x_a} > 0 \qquad \forall\, a \qquad [8.27a]$$

2. The main dependence of a link's travel time is on its own flow. In other words, the derivative of t_a with respect to x_a is much larger than the derivative of t_a with respect to the flow on any other link. This condition can be stated analytically as‡

$$\frac{\partial t_a(\mathbf{x})}{\partial x_a} \gg \frac{\partial t_a(\mathbf{x})}{\partial x_b} \qquad \forall\, b \neq a \qquad [8.27b]$$

Equations [8.27] can be looked upon as conditions for the positive definiteness of the link-travel-time Jacobian.

To illustrate the uniqueness issue, assume now that the performance functions of links 1 and 2 in the aforementioned example (see Figure 8.1) are given by

$$t_1(x_1, x_2) = 2 + 2x_1 + 2x_2 \qquad [8.28a]$$

$$t_2(x_1, x_2) = 4 + 2x_1 + x_2 \qquad [8.28b]$$

instead of [8.20a] and [8.20b]. All the other data remain unchanged. In this case, condition [8.27a] is met for both links, but condition [8.27b] is not satisfied since, for example,

$$\frac{\partial t_1(x_1, x_2)}{\partial x_1} = \frac{\partial t_1(x_1, x_2)}{\partial x_2} \qquad [8.29]$$

†A line integral can be converted to a standard integral by expressing the integrand as a function of a single parameter and then performing the integration along the path defined by that parameter. As it turns out, if the Jacobian is symmetric, the line integral is independent of the path of integration by Green's theorem.

‡The requirement, more precisely, is that $\partial t_a/\partial x_a$ be larger than $\sum_{b \neq a} [\partial t_a/\partial x_b]$.

Equation [8.29] implies that the Jacobian of $[t_1(\mathbf{x}), t_2(\mathbf{x})]$ may not be positive definite. The Jacobian is given by

$$\nabla_{\mathbf{x}}\mathbf{t} = \begin{bmatrix} 2 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad [8.30]$$

This matrix is not positive-definite since the determinant of the second minor along the diagonal is negative, that is,

$$\begin{vmatrix} 2 & 2 \\ 2 & 1 \end{vmatrix} = -2$$

The equilibrium flow pattern that solves this example is not unique. For example, the solution $x_1^* = 0$ and $x_2^* = 5$ is an equilibrium flow pattern since, $t_1^* = 12$ and $t_2^* = 9$; the travel time on the unused path is higher than the travel time on the used path, meaning that the UE conditions are satisfied. Similarly, $x_1^* = 5$ and $x_2^* = 0$ is another equilibrium solution with $t_1^* = 12$ and $t_2^* = 14$. Thus, there are at least two equilibrium flow patterns in this case. This example then demonstrates the existence of multiple equilibria in cases where the Jacobian is not positive definite (even though it may be symmetric).

As this example implies, there may be situations in which the equilibrium flow pattern is not unique. Such situations may require a reformulation of the model or a substitution of a model with lesser interactions. It should be noted, though, that the uniqueness conditions are frequently satisfied. Furthermore, even in cases where, for example, Eq. [8.27b] is violated, the violation is such that the solution may be unique anyway.

The next section examines cases in which the symmetry condition does not hold and consequently, no equivalent minimization program exists.

8.2 EQUILIBRIUM WITH ASYMMETRIC COST FUNCTIONS

This section addresses the question of finding the equilibrium flow pattern in a transportation network in which link performance is a function of the flows on several (and possibly all) links in the network. The problem addressed here is more general than the one discussed in the preceding section, in that the pattern of link interactions is not assumed to be symmetric. This is reflected in the link-travel-time Jacobian, which is assumed to be asymmetric. As mentioned before, in this case there is no known mathematical program the solution of which is the equilibrium flow pattern. The focus of this section is therefore on direct solution algorithms rather than on a mathematical programming formulation. These algorithms, however, are intimately related to the minimization algorithms used throughout this text. It should be noted that the following analysis still requires that the link-travel-time Jacobian be positive definite† (although not symmetric). If this requirement is not met, the problem may not have a unique equilibrium solution, as illustrated in the last part of Secton 8.1.

Algorithm

The equilibration algorithm presented here is based on solving a series of standard UE programs. Each iteration of this procedure requires the solution of one such program.

Assume that, at the nth iteration of the procedure, the values of all the link flow variables $(x_1^n, \ldots, x_A^n)$ are known. At this point, the following mathematical program can be formulated:

$$\min \tilde{z}^n(\mathbf{x}) = \sum_a \int_0^{x_a} t_a(x_1^n, \ldots, x_{a-1}^n, \omega, x_{a+1}^n, \ldots, x_A^n)\, d\omega \qquad [8.31a]$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\ k, r, s \qquad [8.31b]$$

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [8.31c]$$

This program is later referred to as the "subproblem." It is a standard UE equivalent minimization. The objective function includes only simple link performance functions in one argument each (i.e., t_a is a function of x_a only); all other flows that may affect link a are fixed at their values during the nth iteration. This formulation does *not* fix the flows themselves, only their cross-link effects. Accordingly, the subproblem is also known as the "diagonalized" problem (the Hessian of [8.31a] is diagonal since all cross-link effects have been fixed).

To simplify the notation used in the following discussion, let $\tilde{t}_a^n(x_a)$ denote the performance function for link a when all arguments of $t_a(\cdot)$, other than x_a, are fixed at their values during the nth iteration. In other words,

$$\tilde{t}_a^n(x_a) = t_a(x_1^n, \ldots, x_{a-1}^n, x_a, x_{a+1}^n, \ldots, x_A^n)$$

The general process of finding the equilibrium flows requires the following steps:

Step 0: *Initialization.* Find a feasible link-flow vector $\mathbf{x}^n$. Set $n := 0$.

Step 1: *Diagonalization.* Solve subproblem [8.31] by using any UE minimization method. This yields a link-flow vector $\mathbf{x}^{n+1}$.

Step 2: *Convergence Test.* If $\mathbf{x}^n \simeq \mathbf{x}^{n+1}$, stop. If not, set $n := n + 1$, and go to step 1.

†Note that positive definiteness cannot be proven for asymmetric matrices by using the leading minor determinant or the eigenvalues rules. These can be used only for symmetric matrices.

The convergence test required in step 2 of the algorithm can be based on the maximum difference in link flow between successive solutions or on any other criterion measuring the similarity of $\mathbf{x}^n$ and $\mathbf{x}^{n+1}$. For example, the algorithm can be terminated if

$$\frac{1}{A} \sum_a \frac{|x_a^{n+1} - x_a^n|}{x_a^{n+1}} \leq \kappa \qquad [8.32]$$

where A is the number of links in the network and κ is a (dimensionless) predetermined constant.

As the algorithm described above does not attempt to minimize any global objective function, it is not evident that it will result in the desired solution. The proof that this algorithm, at convergence, produces the equilibrium flow pattern is given below.

Proof of Solution†

The following arguments prove that, in the aforementioned procedure, $\mathbf{x}^n = \mathbf{x}^{n+1}$ *if and only if* $\mathbf{x}^n$ is a vector of flows that satisfies the UE conditions. The proof of any such statement has two parts. First, the "if" part (the sufficiency condition), which states that if $\mathbf{x}^n = \mathbf{x}^{n+1}$, then $\mathbf{x}^n$ is an equilibrium flow vector. The second part is the "only if" statement (the necessity condition), which means that $\mathbf{x}^n = \mathbf{x}^{n+1}$ only if $\mathbf{x}^n$ is an equilibrium flow pattern. Another way to state this condition is to treat it as a proof that if $\mathbf{x}^n$ is an equilibrium flow pattern, then $\mathbf{x}^n = \mathbf{x}^{n+1}$. Proving both parts of the statement will demonstrate that ($\mathbf{x}^n = \mathbf{x}^{n+1}$) is a necessary *and* sufficient condition for $\mathbf{x}^n$ to be an equilibrium flow vector.

To prove that if $\mathbf{x}^n = \mathbf{x}^{n+1}$, then $\mathbf{x}^n$ is an equilibrium solution, consider the first-order conditions of program [8.31] (the subproblem):

$$\left(\sum_a \tilde{t}_a^n(x_a^{n+1})\delta_{a,k}^{rs} - u_{rs}^n\right) f_k^{rs^{n+1}} = 0 \qquad \forall\ k, r, s \qquad [8.33a]$$

$$\sum_a \tilde{t}_a^n(x_a^{n+1})\delta_{a,k}^{rs} - u_{rs}^n \geq 0 \qquad \forall\ k, r, s \qquad [8.33b]$$

where $\tilde{t}_a^n(x_a^{n+1}) = t_a(x_1^n, \ldots, x_{a-1}^n, x_a^{n+1}, x_{a+1}^n, \ldots, x_A^n)$ and u_{rs}^n is the value of the dual variable associated with the flow conservation constraint [8.31b] at the solution of program [8.31]. Note that $\sum_a \tilde{t}_a^n(x_a^{n+1})\delta_{a,k}^{rs}$ is not equal to $c_k^{rs^{n+1}}$ [which equals $\sum_a t_a(\mathbf{x}^{n+1})\delta_{a,k}^{rs}$], nor does it equal $c_k^{rs^n}$ [which equals $\sum_a t_a(\mathbf{x}^n)\delta_{a,k}^{rs}$]. Equations [8.33] therefore do not represent any equilibrium condition.

If, however, the convergence criterion in step 2 above is met,

$$x_a^{n+1} = x_a^n \qquad \forall\ a \qquad [8.34]$$

†The reader may skip this proof without loss of continuity.

In this case x_a^n can be substituted for x_a^{n+1} in Eqs. [8.33] (and $f_k^{rs^n}$ can be substituted for $f_k^{rs^{n+1}}$) to give

$$\left(\sum_a \tilde{t}_a^n(x_a^n)\delta_{a,k}^{rs} - u_{rs}^n\right)f_k^{rs^n} = 0 \qquad \forall\ k, r, s \qquad [8.35a]$$

$$\sum_a \tilde{t}_a^n(x_a^n)\delta_{a,k}^{rs} - u_{rs}^n \geq 0 \qquad \forall\ k, r, s \qquad [8.35b]$$

Now, note that

$$\tilde{t}_a^n(x_a^n) = t_a(\mathbf{x}^n) = t_a^n$$

and

$$\sum_a t_a^n\,\delta_{a,k}^{rs} = c_k^{rs^n}$$

Equations [8.35a] and [8.35b] can thus be rewritten in terms of path-flow variables to state

$$(c_k^{rs^n} - u_{rs}^n)f_k^{rs^n} = 0 \qquad \forall\ k, r, s \qquad [8.36a]$$

$$c_k^{rs^n} - u_{rs}^n \geq 0 \qquad \forall\ k, r, s \qquad [8.36b]$$

Equations [8.36] imply that $\mathbf{x}^n$ is an equilibrium flow vector and that the sufficient condition is thus met; if $\mathbf{x}^n = \mathbf{x}^{n+1}$, then $\mathbf{x}^n$ is a UE flow vector.

To prove the necessity condition, assume that $\mathbf{x}^n$ is an equilibrium flow vector, that is, it satisfies Eqs. [8.36] (in the sense that the path travel times, $\{c_k^{rs^n}\}$, which are based on $\mathbf{x}^n$ satisfy these equations). This means that $\mathbf{x}^n$ also satisfies Eqs. [8.33]. This statement does not negate the fact that, when program [8.31] is solved in step 1 of the aforementioned algorithm, there is a unique set of link flows, namely $\mathbf{x}^{n+1}$, which minimizes the program. This set of flows, obviously, satisfies the first-order conditions given for this program in Eqs. [8.33]. Thus both $\mathbf{x}^n$ and $\mathbf{x}^{n+1}$ satisfy Eqs. [8.33], which have a unique solution (since the link-travel-time Jacobian was assumed to be positive definite). It therefore must be true that (if $\mathbf{x}^n$ is an equilibrium flow vector, then) $\mathbf{x}^n = \mathbf{x}^{n+1}$. This completes the proof. □

It should be noted that this is not a proof of convergence, but rather a proof that if the algorithm converges, its solution is the equilibrium flow pattern. The algorithm can be proven to converge in cases where the Jacobian of the link-travel-time function is positive definite. That proof, however, is beyond the scope of this text and is referred to in Section 8.4.

The algorithm is known as the diagonalization method, since the Jacobian of the subproblem [8.31] is made to be diagonal by fixing the cross-link interactions at a given flow level. The following example illustrates how the diagonalization procedure converges to the equilibrium solution.

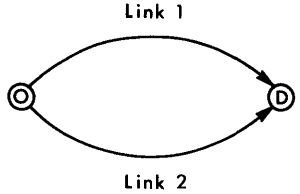


Figure 8.2 Network example with two links and one O–D pair.

Example

Consider the simple two-link network example depicted in Figure 8.2, with the following link performance functions:

$$t_1 = 2 + 4x_1 + x_2 \qquad [8.37a]$$

$$t_2 = 4 + 3x_2 + 2x_1 \qquad [8.37b]$$

The total O–D flow is 5 flow units, that is,

$$x_1 + x_2 = 5 \qquad [8.37c]$$

The equilibrium solution of this system can be determined analytically by solving Eqs. [8.37] directly. Assuming that both links are used at equilibrium, the equilibrium condition is

$$t_1(x_1, x_2) = t_2(x_1, x_2)$$

The resulting equilibrium solution is then $x_1^* = 3$ and $x_2^* = 2$, with $t_1^* = t_2^* = 16$. The equivalent mathematical programming approach to this problem cannot be used since the symmetry condition does not hold. The Jacobian of the link-travel-time function is the following asymmetric (though positive-definite) matrix

$$\begin{pmatrix} \dfrac{\partial t_1}{\partial x_1} & \dfrac{\partial t_1}{\partial x_2} \\ \dfrac{\partial t_2}{\partial x_1} & \dfrac{\partial t_2}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 4 & 1 \\ 2 & 3 \end{pmatrix} \qquad [8.38]$$

The problem can be solved, however, by using the diagonalization algorithm described in this section.

To apply this algorithm, an initial feasible solution is needed.† Assume that this solution is

$$x_1^0 = 0$$

$$x_2^0 = 5$$

Furthermore, assume that the convergence criterion is based on the relative change in the flow (i.e., if $\max_a \{|x_a^{n+1} - x_a^n|/x_a^n\} \leq \kappa$, stop) and that $\kappa = 0.01$

†An initial feasible solution may be obtained, for example, by applying an all-or-nothing assignment on the basis of free-flow travel times, as mentioned in connection with the standard UE case.

(or 1%). The algorithmic iterations can be described then as follows:

First iteration:

Step 1: *Diagonalization.* Solve the subproblem:

$$\min z(x_1, x_2) = \int_0^{x_1} (2 + 4\omega + 5)\, d\omega + \int_0^{x_2} (4 + 3\omega)\, d\omega$$

subject to

$$x_1 + x_2 = 5$$

$$x_1, x_2 \geq 0$$

This program can be solved by integrating the objective function, substituting the flow conservation constraint, differentiating, and finding the root of the resulting derivative. The solution of step 1 is‡

$$x_1^1 = 3.86$$

$$x_2^1 = 1.14$$

Step 2: *Convergence Test.* Compare

$$\begin{pmatrix} x_1^0 \\ x_2^0 \end{pmatrix} = \begin{pmatrix} 0 \\ 5 \end{pmatrix} \neq \begin{pmatrix} x_1^1 \\ x_2^1 \end{pmatrix} = \begin{pmatrix} 3.86 \\ 1.14 \end{pmatrix}$$

or

$$\max_a \left\{ \frac{x_a^1 - x_a^0}{x_a^0} \right\} = \frac{3.86 - 0}{0} = \infty$$

The convergence criterion is not met.

Second iteration:

Step 1: *Diagonalization.* Solve the subproblem

$$\min z(x_1, x_2) = \int_0^{x_1} (2 + 4\omega + 1.14)\, d\omega + \int_0^{x} (4 + 3\omega + 7.72)\, d\omega$$

subject to

$$x_1 + x_2 = 5$$

$$x_1, x_2 \geq 0$$

The solution is

$$x_1^2 = 3.37$$

$$x_2^2 = 1.63$$

‡More accurately, the solution shown below is $x_1^1 = 3.857143$.

TABLE 8.1 Iterations of the Diagonalization Procedure for the Network in Figure 8.2

Iteration	x_1^n	x_2^n	$\max\left\{\left\lvert\frac{x_a^n - x_a^{n-1}}{x_a^{n-1}}\right\rvert\right\} \times 100\%$	$\max\left\{\left\lvert\frac{x_a^n - x^*}{x^*}\right\rvert\right\} \times 100\%$
0	0	5	—	100.00
1	3.857	1.143	∞	42.85
2	3.367	1.633	42.87	18.35
3	3.157	1.843	12.86	7.85
4	3.067	1.933	4.88	3.35
5	3.029	1.971	1.97	1.45
6	3.012	1.988	0.86	0.60
7	3.005	1.995	0.35	0.25
8	3.002	1.998	0.15	0.10
9	3.001	1.999	0.05	0.05
10	3.000	2.000	0.03	0.00

Step 2: *Convergence Test.* Compare

$$\begin{pmatrix} x_1^1 \\ x_2^1 \end{pmatrix} = \begin{pmatrix} 3.86 \\ 1.14 \end{pmatrix} \neq \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix} = \begin{pmatrix} 3.37 \\ 1.63 \end{pmatrix}$$

or

$$\max_a \left\{\frac{x_a^2 - x_a^1}{x_a^1}\right\} = \frac{1.63 - 1.14}{1.14} = 0.43$$

The convergence criterion is not met.

A summary of 10 algorithmic iterations is shown in Table 8.1. This table displays the iteration number, the solution of the subproblem (i.e., the diagonalized UE program), and two convergence measures. The fourth column shows the convergence measure mentioned above, and the fifth column depicts the convergence relative to the *actual* equilibrium solution which was found analytically. In field applications the last measure is not available, of course, but in this example it shows the actual convergence rate of the procedure.

As the table indicates, the algorithm indeed converges to the correct equilibrium solution. Furthermore, this convergence is relatively rapid. The convergence criterion is met at the sixth iteration, where the convergence measure is 0.86%.

Convergence Considerations and a Streamlined Algorithm

Figure 8.3 depicts the convergence pattern as illustrated by the convergence measure, and the actual convergence itself. Both measures show a monotonic and asymptotic convergence rate. In other words, the convergence measure decreases with every iteration, and the marginal contribution of each successive iteration is decreasing. This convergence pattern is similar to the

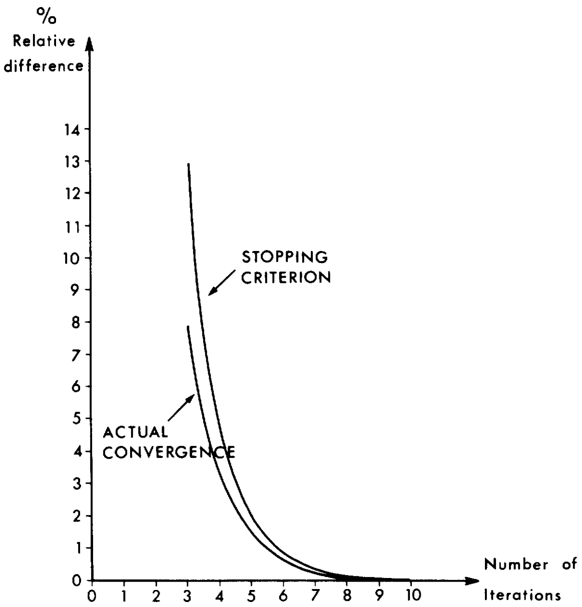


Figure 8.3 Convergence measure and actual convergence (with respect to the known solution) as a function of the number of iterations of the diagonalization algorithm.

convergence pattern of the convex combinations algorithm when it is applied to the solution of the UE minimization program.

The convergence pattern depicted in Figure 8.3 is typical of the diagonalization algorithm and it holds for larger networks as well. This algorithm requires, however, significant computational effort, since each iteration involves the solution of a full-scale UE program (the subproblem). In the example given above, these intermediate UE subproblems were solved analytically. When the diagonalization method is applied to larger networks, however, the subproblems have to be solved iteratively. A crucial question with regard to the applicability of the entire approach is the accuracy with which these subproblems have to be solved.

Before the applicability question is addressed, however, the question of validity should be raised. The proof that the algorithm generates an equilibrium solution assumes that the subproblems are solved accurately. This proof may not apply when the subproblems are solved only approximately. Both the validity and the applicability questions can be better addressed in the context of the following considerations:

1. The proof that the diagonalization algorithm converges to the equilibrium solution holds if the convergence criterion ($\mathbf{x}^{n+1} \simeq \mathbf{x}^n$) is met, regardless of how $\mathbf{x}^n$ was obtained. In fact, the aforementioned proof is based

only on the *last* iteration of the diagonalization algorithm (i.e., only $\mathbf{x}^{n+1}$ has to be obtained accurately).

2. All the subproblems are subject to the same set of constraints. This means that the optimal solution as well as every intermediate solution of any subproblem is *feasible* for any other subproblem. Thus $\mathbf{x}^n$ is a feasible solution for the $(n + 1)$th subproblem.
3. The convergence pattern of the subproblem solution procedure (when using the convex combination method) is similar to the convergence pattern of the solution procedure to the entire problem (when using the diagonalization algorithm). In fact, it was demonstrated in Chapter 5 that the convex combinations algorithm converges asymptotically and thus the criterion of similarity between consecutive flow patterns is a valid one for the solution of the UE program itself.

These points suggest that the algorithm will be valid even if the subproblems are solved only crudely at each iteration. Furthermore, such a crude solution may enhance the computational efficiency of the entire approach.

Consider a diagonalization algorithm in which each subproblem is solved only approximately by performing only a few iterations of the convex combinations algorithm. The resulting (approximate) solution of each subproblem can, however, be used to initialize the next subproblem (in light of consideration 2). Typically, it can be expected that the similarity between consecutive subproblems (and their approximate solutions) will increase as the algorithm proceeds. The use of the solution of each subproblem to initialize the next one would then cause the initial solution to each subproblem to improve as the algorithm proceeds. Thus, even if the subproblems are to be solved accurately, the computational effort required will decrease as the algorithm proceeds. This was illustrated by the sequence of solutions shown in Table 8.1 for the example given above. As the flow pattern stabilized, the consecutive subproblems have grown in similarity. If the same computational effort is used to solve each subproblem (e.g., a fixed number of convex combinations iterations), the accuracy of the solution to each subproblem will improve as the algorithm proceeds. The last subproblem will invariably be solved with high accuracy (since it is initialized with an accurate solution) and consequently the proof that the algorithm actually determines the equilibrium flow pattern holds (see consideration 1). Consideration 3 lays the ground for a streamlined diagonalization algorithm in which the subproblems are solved by using only one iteration of the convex combinations algorithm. In other words, the solution of each subproblem is obtained by using only a single all-or-nothing assignment. The streamlined algorithm includes the following general steps:

Step 0: *Initialization.* Find a feasible link-flow pattern vector, $\mathbf{x}^n$. Set $n := 0$.

Step 1: *Diagonalization.* Perform one iteration of the convex combinations algorithm on the diagonalized subproblem, using $\mathbf{x}^n$ as the initial solution. This yields a link-flow pattern $\mathbf{x}^{n+1}$.

Step 2: *Convergence test.* If $\mathbf{x}^{n+1} \simeq \mathbf{x}^n$, stop. If not, set $n := n + 1$ and go to step 1.

When the convergence criterion is met, it is met both for the last subproblem and for the diagonalization algorithm as a whole.

The streamlined algorithm can be written in detail by spelling out the diagonalization step. The algorithm is then given by:

Step 0: *Initialization.* Set $n := 0$. Find a feasible link-flow pattern vector $\mathbf{x}^n$.

Step 1: *Travel-time update.* Set $t_a^n = t_a(\mathbf{x}^n)$, $\forall\ a$.

Step 2: *Direction finding.* Assign the O–D flows, $\{q_{rs}\}$, to the network using the all-or-nothing approach based on $\{t_a^n\}$. This yields a link-flow pattern $\{y_a^n\}$.

Step 3: *Move-size determination.* Find a scalar, α_n, which solves the following program:

$$\min z(\alpha) = \sum_a \int^{x_a^n + \alpha(y_a^n - x_a^n)} t_a(x_1^n, \ldots, x_{a-1}^n, \omega, x_{a+1}^n, \ldots, x_A^n)\, d\omega$$

subject to

$$0 \leq \alpha \leq 1$$

Step 4: *Updating.* Set $x_a^{n+1} := x_a^n + \alpha_n(y_a^n - x_a^n)$, $\forall\ a$.

Step 5: *Convergence test.* If $x_a^{n+1} \simeq x_a^n$, $\forall\ a$, stop. The solution is $\mathbf{x}^{n+1}$ Otherwise, set $n := n + 1$ and go to step 1.

This algorithm is similar to the convex combinations algorithm used in Section 5.2 to solve the UE program. The only difference here is that the travel time on each link is updated based on the entire flow pattern, since $t_a = t_a(\mathbf{x})$ rather than $t_a = t_a(x_a)$. The diagonalization principle is reflected in the computation of the step size, α_n, where each link is treated as if its travel time varied only with its own flow. The flow on all other links is assumed fixed at $\mathbf{x}^n$ for this purpose.

The mechanics of this streamlined algorithm and its convergence pattern are illustrated below for the two-link example of Figure 8.2. This example is then used to compare the two algorithms to each other.

Example and Comparison of Algorithms

As before, the link performance functions for the network depicted in Figure 8.2 are

$$t_1 = 2 + 4x_1 + x_2$$
$$t_2 = 4 + 3x_2 + 2x_1$$

and the flow conservation condition is given by $x_1 + x_2 = 5$. Starting with the initial solution $x_1 = 0$ and $x_2 = 5$, the algorithmic iterations are as follows:

First iteration:

Step 1: $t_1^0 = 7; t_2^0 = 19.$

Step 2: Since $t_1^0 < t_2^0$, $y_1^0 = 5$ and $y_2^0 = 0$.

Step 3:†

$$\min_{0 \leqslant \alpha \leqslant 1} \int_0^{5\alpha} (7 + 4\omega)\, d\omega + \int_0^{5+\alpha(-5)} (4 + 3\omega)\, d\omega$$

The solution is $\alpha_0 = 0.343$.

Step 4: $x_1^1 = 1.7; x_2^1 = 3.3.$

Step 5: Go to step 1.

Second iteration:

Step 1: $t_1^1 = 12.1;\ t_2^1 = 17.3.$

Step 2: Since $t_1 < t_2$, $y_1^1 = 5$ and $y_2^1 = 0$.

Step 3:

$$\min_{0 \leqslant \alpha \leqslant 1} \int_0^{1.7+\alpha(5-1.7)} (5.3 + 4\omega)\, d\omega + \int_0^{3.3+\alpha(-3.3)} (7.4 + 3\omega)\, d\omega$$

resulting in $\alpha_1 = 0.224$

Step 4: $x_1^2 = 2.5;\ x_2^2 = 2.5.$

Step 5: Go to step 1.

A summary of 10 algorithmic iterations is shown in Table 8.2.

†Step 3 can be solved analytically by performing the integration and then differentiating and finding the root (with respect to α) of the resulting expression. In this case, after using the flow conservation equation, the result is (as the reader can check) $\alpha_n = (12 - 4x_1^n)/7(y_1^n - x_1^n)$.

TABLE 8.2 Summary of Iterations of the Streamlined Diagonalization Algorithm

Iteration	x_1^n	x_2^n	$\max_a \left\{\left\lvert\frac{x_a^{n+1} - x_a^n}{x_a^n}\right\rvert\right\} \times 100\%$	t_1^n	t_2^n	y_1^n	y_2^n	α_n	$\max_a \left\{\left\lvert\frac{x_a^n - x_a^*}{x_a^*}\right\rvert\right\} \times 100\%$
0	0.000	5.000	—	7.000	19.000	5.000	0.000	0.343	100.00
1	1.714	3.286	∞	12.143	17.287	5.000	0.000	0.224	64.30
2	2.449	2.551	42.88	14.347	16.551	5.000	0.000	0.123	27.55
3	2.764	2.236	12.86	15.292	16.236	5.000	0.000	0.060	11.80
4	2.899	2.101	4.88	15.696	16.101	5.000	0.000	0.028	5.05
5	2.957	2.043	2.00	15.870	16.043	5.000	0.000	0.012	2.15
6	2.981	2.019	0.81	15.944	16.019	5.000	0.000	0.005	0.95
7	2.992	2.008	0.37	15.976	16.008	5.000	0.000	0.002	0.40
8	2.997	2.003	0.15	15.990	16.003	5.000	0.000	0.001	0.17
9	2.999	2.001	0.07	15.994	16.002	5.000	0.000	0.000	0.07
10	2.999	2.001	0.03	15.998	16.001	5.000	0.000	0.000	0.03
	3.000	2.000	0.01						0.01

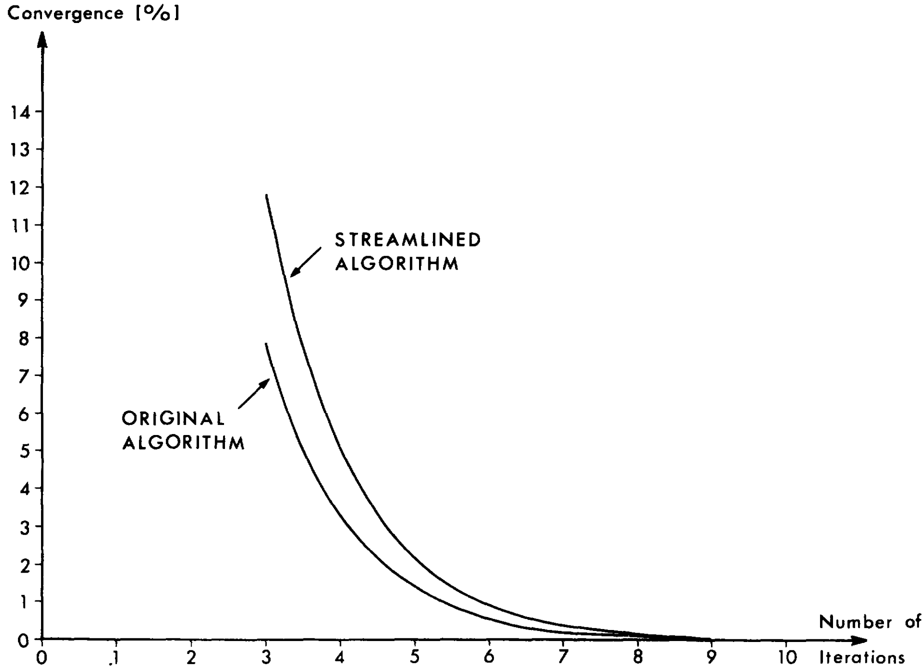


Figure 8.4 Convergence rates for the streamlined and the original diagonalization algorithm as a function of the number of iterations.

Figure 8.4 compares the actual convergence patterns of the original diagonalization procedure and the streamlined algorithm for the example given here. The convergence rate of both procedures seems to be similar in terms of the number of iterations required for convergence. It could be even concluded from this figure that the original algorithm is faster. This conclusion, however, would be erroneous since the relevant basis for comparison should be the *computational effort* required, not the total *number of iterations.* The original algorithm requires several all-or-nothing assignments at each iteration, whereas the streamlined algorithm requires only one such assignment at each iteration. The original algorithm would thus be significantly slower in terms of convergence per unit of computational effort. In large networks, the computational effort is directly related to the number of shortest-path computations, or the number of all-or-nothing assignments required. Figure 8.5 compares the convergence rate of both algorithms assuming that the original algorithm requires, on the average, three all-or-nothing assignments to solve each subproblem. The basis of comparison thus becomes the actual convergence after a given number of all-or-nothing assignments. The reader can appreciate, based on this figure, how much faster the streamlined algorithm is for this example. Limited experience suggests that the streamlined algorithm performs significantly better than the original diagonalization method in large networks as

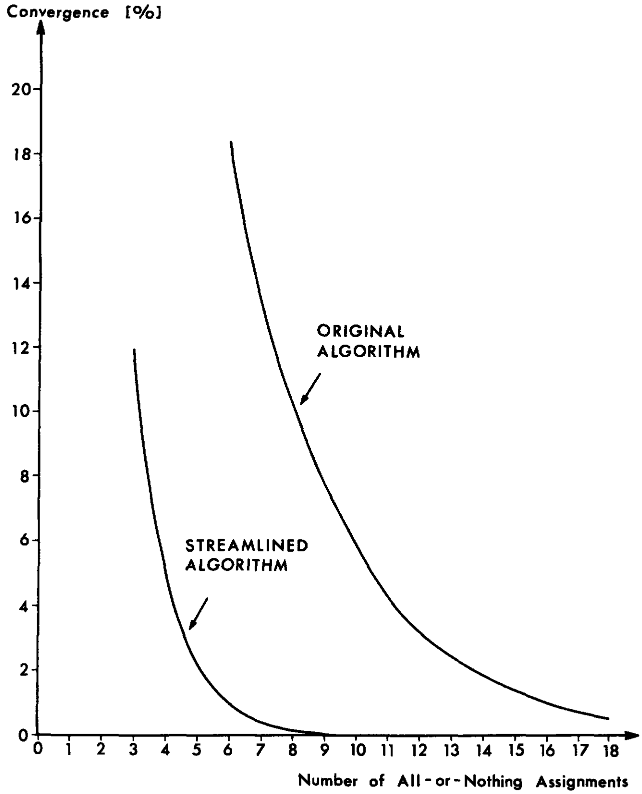


Figure 8.5 Convergence rates for the streamlined and the original diagonalization algorithm as a function of the computational effort. (The figure assumes that each subproblem requires an average of three all-or-nothing assignments to solve.)

well, implying that only one all-or-nothing iteration should be performed for each subproblem.

The streamlined diagonalization method is easy to implement. Existing codes for solving the standard UE program can easily be modified to perform the streamlined algorithms (given the appropriate performance functions).

8.3 SUMMARY

Chapter 8 has extended the framework of the equilibrium analysis to include cases in which travel time over a link depends on the flow on the link as well as on the flow on other links. The independence assumption used to develop the equivalent mathematical programming approach in Chapter 3 may not be justified in some instances encountered in everyday traffic conditions. For example, two-way streets, intersections, merging movements, and turning movements constitute instances in which the delay on a given link may be influenced by the flow on several other links. Thus, in general, the link-travel-time function is given by $t_a = t_a(\mathbf{x})$.

This chapter distinguishes two cases of link interdependence. In the first case, the link interaction pattern is assumed to be symmetric. In other words, the marginal effect of the flow along a given link, on the travel time of another is equal to the marginal effect of the flow through the latter link, on the travel time along the link under study. This condition can be expressed in terms of a requirement that the Jacobian matrix of the link-travel-time vector be symmetric. If the symmetry condition holds, the equilibration problem can be formulated as a mathematical program. Furthermore, if the Jacobian is positive definite, this program will have a unique solution. The equivalent mathematical program can be solved with any minimization algorithm, such as the convex combinations method.

The second case analyzed in this chapter is one in which the link interaction pattern is asymmetric. (The Jacobian is still required to be positive definite, however, to ensure uniqueness of the solution.) In this case, no equivalent minimization program can be formulated and no minimization algorithm can be applied to the problem. Section 8.2 suggests two algorithmic approaches to the solution of the equilibration problem under these circumstances.

The first approach is an algorithm based on an iterative diagonalization procedure, where each iteration requires the solution of a full-scale standard UE program. The algorithm converges when successive solutions are similar. The other approach is a streamlined version of the first one; it calls for a single iteration of the UE solution procedure to be performed at every iteration of the diagonalization procedure. Both algorithms are illustrated for a small network example. The streamlined version is the more efficient one and should be preferred.

8.4 ANNOTATED REFERENCES

The material in Chapter 8 follows a paper by Abdulaal and LeBlanc (1979). These authors developed a methodology for solving joint modal split/traffic assignment problems, which was adapted to the two-way traffic case analyzed in Section 8.1. Dafermos (1971) had suggested a model similar to the one presented in Section 8.1 for two-way traffic. Her approach starts from the general statement of the objective function (Eq. [8.26]) to develop specific models. This work includes a discussion of the existence, uniqueness, and stability of the equilibrium flow pattern.

For the case of asymmetric link interactions given in Section 8.2, this chapter again follows the algorithmic approach of Abdulaal and LeBlanc (1979). Instead of applying it to binary interactions between cars and buses, the approach is generalized to a full interaction pattern among the network links. The approach is then extended with the presentation of the streamlined algorithm in that section.

A large part of the recent work on the subject is centered around the formulation of the equilibrium conditions as variational inequalities. Using this formulation, Smith (1979) demonstrated the existence and uniqueness of traffic equilibrium when the Jacobian of the link-travel-time function is positive definite. Other proofs were suggested by Dafermos (1980), and Aashtiani and Magnanti (1981). Both these references include solution algorithms. The diagonalization algorithm was tested by Fisk and Nguyen (1982) in conjunction with the tests of several solution algorithms for this problem. These authors concluded that the diagonalization procedure outlined in Section 8.2 is the most efficient approach to the problem. This procedure has been embedded in the SATURN Transportation Planning Package developed by Bolland et al. (1979). A series of tests carried out by Nagurny (1983) was less conclusive. The diagonalization algorithm was found to be the most efficient only for some networks. All the tests mentioned above, however, did not use the streamlined version of the diagonalization method. A formal proof of convergence for the diagonalization algorithms was given by Dafermos (1982) and (for a somewhat restricted case) by Florian (1981). Some of these authors refer to this algorithm as the "relaxation" method.

8.5 PROBLEMS

8.1. Discuss the applicability of the assumption of independence among links in various cases, including freeway traffic, signalized and unsignalized intersections, traffic activated signals, two-way traffic, turning movements, and so on. In each case comment on the possibility of the interaction being symmetric.

8.2. Show that the standard UE equivalent minimization (Eqs. [3.3]) is not applicable when link interactions exist. Use the two-way-traffic case to illustrate your point.

8.3. The objective function of the equivalent minimization associated with the two-way interaction equilibrium problem includes the constant $\frac{1}{2}$ in the objective function (see Eq. [8.3a]). How will the program change if this constant is removed? Will it still be an equivalent minimization program for the equilibrium under consideration? Explain.

8.4. **(a)** Prove that the Hessian [8.16a] is positive definite if each block matrix [8.16b] is positive definite.

*__(b)__ Prove that the Jacobian [8.25] is positive definite if the main dependence characteristic [8.27b] holds.

8.5. **(a)** Show that the objective function for the pairwise interaction case (Eq. [8.3a]) can be derived as a special case of the line integral in Eq. [8.26].

*__(b)__ Show that if the link-travel-time Jacobian is symmetric, objective function [8.26] can be used to derive an equivalent minimization for the UE problem with link interactions.

8.6. Write out in detail the diagonalization algorithm given at the beginning of Section 8.2 for solving for equilibrium with asymmetric cost functions. Assume that the solution of each subproblem is based on the convex combinations algorithms.

8.7. Consider the example in Figure 8.1 and assume the following data:

$$t_1(x_1, x_2) = 3 + 2x_1^2 + 2x_2$$

$$t_2(x_1, x_2) = 15 + x_2^2 + 2x_1$$

$$t_3(x_3) = 20 + x_3^2$$

$$x_1 + x_2 = 5$$

(a) Derive the equilibrium solution by solving the UE conditions analytically.

(b) Show that the problem is symmetric. Formulate the associated equivalent minimization program. Solve this program and check that the solution is similar to the one you obtained in part (a).

(c) Does the problem have a unique solution? Analyze this point in detail.

8.8. Apply the convex combinations algorithm to the solution of the example given at the end of Section 8.1 (see Eq. [8.20] and the related description). Start with the initial solution $x_1 = 0$, $x_2 = 5$, $x_3 = 5$. Set a reasonable convergence criterion and explain it. Tabulate the intermediate solutions and all the intermediate quantities used in the iterative process.

8.9. **(a)** Suggest an intuitive explanation of why the streamlined algorithm is more efficient than the original diagonalization algorithm in solving for equilibrium with asymmetric travel-time functions. Based on your arguments, do you expect the relationships between these algorithms to hold true for large networks?

(b) In the original diagonalization, the equilibrium solution to one subproblem can be used to initialize the next one. As these subproblems grow in similarity it can be expected that the last subproblems would need very few iterations to converge while the first ones would need many all-or-nothing assignments. What is the effect of this phenomenon on the convergence rate of the original algorithm compared with the streamlined one?

8.10. Can the streamlined diagonalization algorithm be applied to an equilibrium problem with symmetric travel-time functions? Compare such an application to the application of the convex combinations algorithm to the solution of the appropriate equivalent minimization.

8.11. Describe the modifications to a standard UE convex combinations code that have to be implemented in order for it to perform:

(a) The diagonalization algorithm.

(b) The streamlined algorithm.

8.12. Consider a simple network comprising one O–D pair connected by two links. Let the data for this problem be

$$t_1 = 2.0\left[1 + 0.15\left(\frac{x_1}{12}\right)^4\right] + 0.30\left(\frac{x_2}{8}\right)^2$$

$$t_2 = 3.0\left[1 + 0.15\left(\frac{x_2}{8}\right)^4\right] + 0.20\left(\frac{x_1}{12}\right)^2$$

$$x_1 + x_2 = 20$$

Solve this program using the streamlined diagonalization algorithm. Use a convergence criterion of 0.10 and show the results of all intermediate iterations. Use $x = (20, 0)$ as an initial solution.

9

Supernetworks— Models of Joint Travel Choices

This chapter combines the various problems and algorithms presented in the preceding three chapters into a unified framework. It examines the four basic dimensions of travel choice modeled in the traditional urban transportation planning process: whether to take a trip, by which mode, to what destination, and by which route. These four dimensions can be modeled in a unified manner by formulating an equilibrium model with variable demand and joint modal split, trip distribution, and traffic assignment components.

The combined problems discussed in this chapter are formulated and solved using a modified network representation in a manner similar to the formulation of the modal split/traffic assignment problem and the distribution/assignment problem, in Chapters 6 and 7. In other words, the basic network is augmented with virtual (dummy) links to represent several choice dimensions. Such augmented networks are refered to in here as *supernetworks.* The modified networks used to convert the joint mode split/traffic assignment problems in Section 6.4 into equivalent UE programs, are, special cases of supernetworks according to this definition. Similarly, the networks used to convert distribution/assignment problems into equivalent UE problems in Section 7.2 are also special cases of supernetworks.

The supernetwork representation reduces the problem of solving several models jointly to that of finding the UE flow pattern over a single network. This latter problem can then be solved using the methods discussed in Chapter 5. In particular, the resulting equivalent UE problem (over the supernetwork) may include link interactions, in which case the methods explored in Chapter 8 should be used to find the equilibrium flow pattern. In this context, the supernetwork representation means that link interactions can be modeled among dummy links as well as among the basic network links. For example, the interaction between transit vehicles and the general traffic can be modeled and accounted for in the equilibration process.

The first part of this chapter discusses modal split/traffic assignment models, in which the interaction between transit vehicles and the general traffic is modeled explicitly. This treatment of the modal split/traffic assignment problem generalizes the limited exposition of the subject given in Section 6.4. The second part of the chapter describes the supernetwork concept discussed here in more detail and uses it to formulate and solve several travel dimensions simultaneously.

9.1 MODAL SPLIT IN MIXED TRAFFIC—EXTENSIONS

As an illustration of the methodology presented in Chapters 6 through 8, this section describes a joint modal split/traffic assignment problem which generalizes the problem originally investigated in Section 6.4. The problem formulation is expanded in three aspects: First, a full-scale transit network is included in the analysis and a UE condition is assumed to hold over the transit network (separately). Second, the link independence assumption is removed to account for interactions among automobiles and buses, all moving jointly in the traffic stream. Finally, the combined mode split/traffic assignment model is extended to include any type of transit assignment procedure (such as procedures that are not based on the UE assumptions within the transit network).

Mode Split with Full Transit Network

Transit trips between the various origins and destinations take place exclusively over the transit network. This network consists of nodes and links, where the nodes typically represent bus stops or transit stations, and the links represent line-haul movements. This section analyzes a problem of determining simultaneously the equilibrium flows over an automobile network and a transit network. At this point it is still assumed that the transit links and the automobile links are independent of each other (in terms of the effects of flow on travel time).

The notation used in this section for the transit network parallels the automobile network notation used throughout this text. Accordingly, let $\hat{x}_b$ and $\hat{t}_b$ denote the flow and travel time, respectively, on transit link b, which is part of a set of transit links $\hat{\mathscr{A}}$. (Note that the flow on the transit links is measured in terms of person trips, not vehicular trips.) Let $\hat{f}_l^{rs}$ and $\hat{c}_l^{rs}$ denote the flow and travel time, respectively, on transit route l connecting O–D pair r–s ($l \in \hat{\mathscr{L}}_{rs}$, where $\hat{\mathscr{L}}_{rs}$ is the set of transit routes connecting origin r and destination s). In this representation, the origin and destination centroid nodes are common to both the automobile and the transit networks (see the discussion of network representation in Section 1.2). In parallel with the auto links, $\hat{t}_b = \hat{t}_b(\hat{x}_b)$, $\forall\, b$, and the incidence relationships are assumed to hold over the transit network, that is,

$$\hat{x}_b = \sum_{rs} \sum_{l \in \mathscr{L}_{rs}} \hat{f}_l^{rs} \hat{\delta}_{b,l}^{rs} \qquad \forall\, b \in \hat{\mathscr{A}} \qquad [9.1a]$$

$$\hat{c}_l^{rs} = \sum_{b \in \hat{\mathscr{A}}} \hat{t}_b \hat{\delta}_{b,l}^{rs} \qquad \forall\, l \in \hat{\mathscr{L}}_{rs}, \forall\, r, s \qquad [9.1b]$$

where $\hat{\delta}_{b,l}^{rs} = 1$ if transit link b is on the lth transit route connecting origin r and destination s, and $\hat{\delta}_{b,l}^{rs} = 0$ otherwise. Finally, let q_{rs} denote the automobile O–D trip rate and let $\hat{q}_{rs}$ be the transit O–D trip rate between origin r and destination s. Both $\{\hat{q}_{rs}\}$ and $\{q_{rs}\}$ are variables in this problem. The total (auto + transit) O–D trip rate for O–D pair r–s, however, is assumed to be fixed and known. This total O–D trip rate is denoted by $\bar{q}_{rs}$. The following relationships then hold:†

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\, r, s \qquad [9.2a]$$

$$\sum_l \hat{f}_l^{rs} = \hat{q}_{rs} \qquad \forall\, r, s \qquad [9.2b]$$

$$q_{rs} + \hat{q}_{rs} = \bar{q}_{rs} \qquad \forall\, r, s \qquad [9.2c]$$

The type of equilibrium over the transit network depends on the assumed transit-route-choice behavior. One possibility is to assume that each transit rider chooses the shortest travel-time route from origin to destination. The equilibrium flows over the transit network will, in this case, satisfy a set of UE-type conditions. In many situations, however, this assumption may not be appropriate and other transit assignment models should be used, as shown later in this section. At this point it is assumed that the UE conditions hold throughout the transit network. Thus, at equilibrium, the flows and travel times over the transit network are such that

$$(\hat{c}_l^{rs} - \hat{u}_{rs})\hat{f}_l^{rs} = 0 \qquad \forall\, l, r, s \qquad [9.3a]$$

$$\hat{c}_l^{rs} - \hat{u}_{rs} \geq 0 \qquad \forall\, l, r, s \qquad [9.3b]$$

where $\hat{u}_{rs}$ is the minimum transit travel time between origin r and destination s. A similar set of conditions would hold over the automobile network, that is, at equilibrium

$$(c_k^{rs} - u_{rs})f_k^{rs} = 0 \qquad \forall\, k, r, s \qquad [9.4a]$$

$$(c_k^{rs} - u_{rs}) \geq 0 \qquad \forall\, k, r, s \qquad [9.4b]$$

Equations [9.3] and [9.4] define separate UE conditions for the transit

†As mentioned in Section 6.4, it is assumed that the auto occupancy factor is 1. for simplicity of presentation. If this is not the case, Eq. [9.2c] has to be modified to read $\psi q_{rs} + \hat{q}_{rs} = \bar{q}_{rs}$, where ψ is the average automobile occupancy.

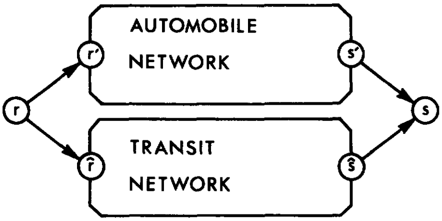


Figure 9.1 Supernetwork representing a joint modal split/traffic assignment problem, assuming a UE-type assignment within the transit network.

and automobile networks. The role of the mode split model is to allocate the O–D flows between these networks.

It would be simple to assume that the UE equilibrium equations govern the modal split as well as the assignment of trips within each mode. In this case the joint mode split/traffic assignment problem would be reduced to a UE problem over the supernetwork shown in Figure 9.1 (see Problem 9.1). In this supernetwork every origin centroid, r, is connected to a transit node, $\hat{r}$, and an automobile node, r', by dummy links. The destination node is split in a similar fashion. The dummy links $r \rightarrow r'$, $r \rightarrow \hat{r}$, $s' \rightarrow s$, and $\hat{s} \rightarrow s$ all have zero travel costs. When this supernetwork is in a user-equilibrium state, each of its components (i.e., the transit network and the auto network) is also in a user-equilibrium state. As explained in Section 6.4 (following program [6.25]), however, such an assumption is not reasonable and is not generally supported by empirical observations.

The mode split between each O–D pair is more appropriately based on an empirically calibrated modal split function. As in Chapter 6, the modal split function assumed here is based on the logit model. Accordingly, the automobile flow between origin r and destination s, at equilibrium, is given by

$$q_{rs} = \bar{q}_{rs} \frac{1}{1 + e^{\theta(u_{rs} - \hat{u}_{rs} - \Psi_{rs})}} \qquad \forall\ r, s \qquad [9.5]$$

where Ψ_{rs} and θ are (nonnegative valued, empirical) parameters, and the transit flow is given by $\hat{q}_{rs} = \bar{q}_{rs} - q_{rs}$, $\forall\ r, s$. Note that this logit formula is somewhat more general than the previous logit mode choice model introduced in Eq. [6.26]. Equation [9.5] includes a constant, Ψ_{rs}, which captures the effects of all factors other than the travel-time difference on the modal split. For example, a positive value of Ψ_{rs} can be interpreted as an "automobile preference" factor. In other words, $\Psi_{rs} > 0$ implies that the share of automobile trips between O–D pair r–s is greater than the transit share between this O–D pair even in cases in which $u_{rs} = \hat{u}_{rs}$.

Equivalent program. Following the logic used in Section 6.4 to formulate modal split/traffic assignment problems, Eq. [9.5] should be inverted to yield

$$u_{rs}(q_{rs}) = \frac{1}{\theta} \ln \frac{\bar{q}_{rs} - q_{rs}}{q_{rs}} + \hat{u}_{rs} + \Psi_{rs} \qquad [9.6a]$$

This equation expresses the O–D travel time over the auto network as a function of the automobile O–D flow. This is the inverse of Eq. [9.5] which gives the O–D auto flow as a function of the O–D travel time (as well as other factors). Equation [9.6a] can also be expressed in terms of $\hat{q}_{rs} = \bar{q}_{rs} - q_{rs}$, that is,

$$W_{rs}(\hat{q}_{rs}) = u_{rs}(q_{rs}) = \frac{1}{\theta} \ln \frac{\hat{q}_{rs}}{\bar{q}_{rs} - \hat{q}_{rs}} + \hat{u}_{rs} + \Psi_{rs} \qquad [9.6b]$$

The difference between the case analyzed here and the one analyzed in Section 6.4 (see program [6.28]) is that $\hat{u}_{rs}$ is not fixed and known here. It is the minimum travel time between r and s over the transit network at equilibrium. Consider, then, the following equivalent mathematical program:

$$\min z(\mathbf{x}, \hat{\mathbf{x}}, \hat{\mathbf{q}}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega + \sum_{rs} \int_0^{\hat{q}_{rs}} \left(\frac{1}{\theta} \ln \frac{\omega}{\bar{q}_{rs} - \omega} + \Psi_{rs} \right) d\omega + \sum_b \int_0^{\hat{x}_b} \hat{t}_b(\omega)\, d\omega \qquad [9.7a]$$

subject to

$$\sum_k f_k^{rs} = \bar{q}_{rs} - \hat{q}_{rs} \qquad \forall\, r, s \quad (u_{rs}) \qquad [9.7b]$$

$$\sum_l \hat{f}_l^{rs} = \hat{q}_{rs} \qquad \forall\, r, s \quad (\hat{u}_{rs}) \qquad [9.7c]$$

$$f_k^{rs}, \hat{f}_l^{rs} \geq 0 \qquad \forall\, k, l, r, s \qquad [9.7d]$$

The first-order conditions for this program include the UE conditions over the transit network and over the automobile network (i.e., Eqs. [9.3] and [9.4]). In addition, the following condition holds at the minimum point:

$$\frac{1}{\theta} \ln \frac{\hat{q}_{rs}}{\bar{q}_{rs} - \hat{q}_{rs}} - u_{rs} + \Psi_{rs} + \hat{u}_{rs} = 0 \qquad \forall\, r, s \qquad [9.8]$$

This equation can be rewritten as

$$\hat{q}_{rs} = \bar{q}_{rs} \frac{1}{1 + e^{\theta(\hat{u}_{rs} - u_{rs} + \Psi_{rs})}} \qquad \forall\, r, s$$

which is consistent with Eq. [9.5]. Thus, at equilibrium, the transit and automobile trip rates between each O–D pair are given by a logit equation. In addition, the flows over each mode's network are distributed in accordance with the UE conditions. Consequently, program [9.7] can be regarded as an equivalent mode split/traffic assignment minimization (see Problem 9.2a).

Note that program [9.7] could also have been formulated in terms of $\{q_{rs}\}$ rather than $\{\hat{q}_{rs}\}$ by using Eq. [9.6a] (rather than [9.6b]) in the objective function with the appropriate modifications to constraints [9.7b] and [9.7c] (see Problem 9.2b). Note also that due to the shape of the logit curve, the final solution will be such that $0 < \hat{q}_{rs} < \bar{q}_{rs}$ and $0 < q_{rs} < \bar{q}_{rs}$ for all O–D pairs served by the two modes.

Algorithmic solution procedures. Program [9.7] can be solved by using the convex combinations algorithm described in Section 5.2. The direction-finding step of this algorithm would involve, at the nth iteration, the solution of the following linear program:

$$\min z^n(\mathbf{g}, \hat{\mathbf{g}}, \hat{\mathbf{v}}) = \sum_{rs} \left[\sum_k c_k^{rsn} g_k^{rs} + \sum_l \hat{c}_l^{rsn} \hat{g}_l^{rs} + \left(\frac{1}{\theta} \ln \frac{\hat{q}_{rs}^n}{\bar{q}_{rs} - \hat{q}_{rs}^n} + \Psi_{rs} \right) \hat{v}_{rs} \right] \qquad [9.9a]$$

subject to

$$\sum_k g_k^{rs} = \bar{q}_{rs} - \hat{v}_{rs} \qquad \forall\, r, s \qquad [9.9b]$$

$$\sum_l \hat{g}_l^{rs} = \hat{v}_{rs} \qquad \forall\, r, s \qquad [9.9c]$$

$$g_k^{rs}, \hat{g}_l^{rs} \geq 0 \qquad \forall\, k, l, r, s \qquad [9.9d]$$

where g_k^{rs}, $\hat{g}_l^{rs}$, and $\hat{v}_{rs}$ denote the auxiliary variables associated with the flow variables f_k^{rs}, $\hat{f}_l^{rs}$, and $\hat{q}_{rs}$, respectively. The superscript n in program [9.9] refers to the values of the corresponding variables at the nth iteration of the algorithm, as in the preceding chapters. Program [9.9] also can be written in terms of the auxiliary path-flow variables alone by using constraint [9.9c]. The program then becomes

$$\min z^n(\mathbf{g}, \hat{\mathbf{g}}) = \sum_{rs} \left[\sum_k c_k^{rsn} g_k^{rs} + \sum_l \left(\hat{c}_l^{rsn} + \frac{1}{\theta} \ln \frac{\hat{q}_{rs}^n}{\bar{q}_{rs} - \hat{q}_{rs}^n} + \Psi_{rs} \right) \hat{g}_l^{rs} \right] \qquad [9.10a]$$

subject to

$$\sum_k g_k^{rs} + \sum_l \hat{g}_l^{rs} = \bar{q}_{rs} \qquad \forall\, r, s \qquad [9.10b]$$

$$g_k^{rs}, \hat{g}_l^{rs} \geq 0 \qquad \forall\, k, l, r, s \qquad [9.10c]$$

This program can be decomposed by O–D pair. For each pair r–s, the objective function will be minimized by assigning the entire O–D flow, $\bar{q}_{rs}$, to the (current) minimum-travel-time path. This path may traverse either the auto network (in which case its travel time is u_{rs}^n) or the transit network (in which case its travel time is $\hat{u}_{rs}^n$). The consideration as to which mode is to be assigned is as follows. If

$$u_{rs}^n < \hat{u}_{rs}^n + \frac{1}{\theta} \ln \frac{\hat{q}_{rs}^n}{\bar{q}_{rs} - \hat{q}_{rs}^n} + \Psi_{rs} \qquad [9.11]$$

then $\bar{q}_{rs}$ is assigned to the minimum path through the road network; otherwise, $\bar{q}_{rs}$ is assigned to the minimum path through the transit network. In case of equality, it may be assigned to either path. (If Eq. [9.11] holds as an equality [even approximately] for all O–D pairs, the current solution is likely to be the equilibrium flow pattern.) This process is repeated for all O–D pairs, yielding an auxiliary flow pattern $\{y_a^n\}$, $\{\hat{y}_a^n\}$, and $\{\hat{v}_{rs}^n\}$. These flows are then used to define the descent direction for the algorithm. The move size along this direction is determined by solving

$$\min_{0 \leqslant \alpha \leqslant 1} z(\alpha) = \sum_a \int_0^{x_a^n + \alpha(y_a^n - x_a^n)} t_a(\omega)\, d\omega + \sum_{rs} \int_0^{\hat{q}_{rs}^n + \alpha(\hat{v}_{rs}^n - \hat{q}_{rs}^n)} \left(\frac{1}{\theta} \ln \frac{\omega}{\bar{q}_{rs} - \omega} + \Psi_{rs} \right) d\omega$$
$$+ \sum_b \int_0^{\hat{x}_b^n + \alpha(\hat{y}_b^n - \hat{x}_b^n)} \hat{t}_b(\omega)\, d\omega \qquad [9.12]$$

The move size that minimizes this program, α_n, is used to update the flows for the next iteration.

When this algorithm is applied, the convergence criteria mentioned in previous chapters can be used to terminate the algorithm. The initialization procedure is also similar to those discussed earlier (see Section 6.4 for an example). As mentioned there, care must be taken not to initialize any of the O–D flows (automobile or transit) at zero, to avoid undefined values of the objective function.

Another algorithmic approach that can be adopted in this case is based on the double-stage algorithms explained in Chapter 7. In this case the direction-finding step would be executed as follows:

Direction finding:

(a) Find the current minimum paths $\{u_{rs}^n\}$ and $\{\hat{u}_{rs}^n\}$ over the automobile and transit networks.

(b) Find the auxiliary auto and transit O–D flows, $\{v_{rs}^n\}$ and $\{\hat{v}_{rs}^n\}$, based on the current travel times, that is,

$$\hat{v}_{rs}^n = \bar{q}_{rs}[(1 + \exp \theta(\hat{u}_{rs}^n - u_{rs}^n + \Psi_{rs})]^{-1} \quad \text{and} \quad v_{rs}^n = \bar{q}_{rs} - \hat{v}_{rs}^n \qquad \forall\, r, s$$

(c) Assign v_{rs}^n and $\hat{v}_{rs}^n$ to the minimum paths identified in (a). This yields a new flow pattern $\{y_a^n\}$ and $\{\hat{y}_a^n\}$.

The rest of the steps are identical to the ones described above in connection with the convex combinations algorithms.

The auxiliary flows used to define the descent direction of the double-stage algorithms can be interpreted as the solution of the program

$$\min z(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{v}}) = \sum_a t_a^n y_a + \sum_{rs} \int_0^{\hat{v}_{rs}} \left(\frac{1}{\theta} \ln \frac{\omega}{\bar{q}_{rs} - \omega} + \Psi_{rs} \right) d\omega + \sum_b \hat{t}_b^n \hat{y}_b \qquad [9.13]$$

subject to constraints [9.9b] through [9.9d]. The reader is asked (in Problem

9.3) to investigate this application of the double-stage algorithm. As it turns out, this algorithm is more efficient than the convex combinations method for solving program [9.9]. The considerations leading to this conclusion are similar to those presented in Section 7.2 in conjunction with distribution/assignment problems.

The logit-based modal split/traffic assignment problem discussed her˜ can be formulated as an equivalent UE supernetwork by introducing a set of dummy links associated with each O–D pair (see problem 9.4). The determination of equilibrium flows over this network may be prohibitively expensive since each O–D pair has to be assigned separately in the course of applying, say, the convex combinations method. If, however, direct O–D links are introduced (as in Section 6.4) a supernetwork can be constructed and solved efficiently. This possibility is explored in the last part of this section. First, however, the issues associated with modeling transit in mixed traffic are explained.

Mode Split with Interdependent Modes

The modal split/traffic assignment model discussed thus far assumes that no interactions exist between the transit links and the auto links. This assumption is not generally applicable, however, to bus transit. Buses move with the general traffic and experience the same congestion and delays as do automobiles. This section extends the analysis of the joint modal split/traffic assignment problem to cases in which the delay of either mode depends on flows of both modes. At equilibrium, it is assumed that the UE conditions are satisfied over both the auto and transit networks separately and that the mode split between each O–D pair is given by a logit model (such as the one in Eq. [9.5]).

To model this bus/auto modal split/traffic assignment problem, assume that each flow direction on a street is represented by two links: an auto link and a transit link. Assume further that the set of auto links, $\mathscr{A}$, and the set of transit links, $\hat{\mathscr{A}}$, have been ordered so that a and $\hat{a}$ represent auto and transit links corresponding to the same physical street (and the same direction of flow). Let $\hat{x}_a$ and x_a represent the transit and auto flows on link $\hat{a}$ (where $\hat{a} \in \hat{\mathscr{A}}$) and link a (where $a \in \mathscr{A}$), respectively. Similarly, let $\hat{t}_a$ and t_a represent the travel times on links $\hat{a}$ and a, respectively. The link performance functions are given by

$$t_a = t_a(x_a, \hat{x}_a) \qquad \forall\, a \in \mathscr{A} \quad \text{and} \quad \hat{a} \in \hat{\mathscr{A}} \tag{9.14a}$$

for the auto links, and

$$\hat{t}_a = \hat{t}_a(\hat{x}_a, x_a) \qquad \forall\, \hat{a} \in \hat{\mathscr{A}} \quad \text{and} \quad a \in \mathscr{A} \tag{9.14b}$$

for the bus links.

The main congestion effects stem from the influence of the automobile flows on the travel time of both automobiles and transit vehicles. In comparison, the effect of patrons' flow on transit delay is small. Furthermore, their effect on auto delay can be assumed to be zero as long as the flow of buses remains unchanged. The effect of the bus movements themselves on the traffic flow can be modeled as a constant, since bus routes and schedules can be assumed fixed in a short-run equilibrium analysis. When the analysis horizon is longer, it can be assumed that the bus operator changes bus routes and frequencies in response to changes in transit ridership. In this case the flow of bus riders can be assumed to have some influence on the traffic flow.† This text, however, deals with short-run equilibrium in which this effect is secondary. In any event, travel time and flow interactions between the transit and automobile modes cannot be assumed symmetric. In other words,

$$\frac{\partial t_a(x_a, \hat{x}_a)}{\partial \hat{x}_a} \neq \frac{\partial \hat{t}_a(\hat{x}_a, x_a)}{\partial x_a} \qquad \forall\, a$$

meaning that (as mentioned in Chapter 8) the mode split/traffic assignment problem under consideration cannot be formulated and solved as an equivalent minimization program. The diagonalization algorithm outlined in Section 8.2 can, however, be used to solve this problem.

The subproblem that has to be solved at the *n*th iteration of the diagonalization algorithm, when applied to this problem, is given by (see program [8.31] and the related discussion)

$$\min \tilde{z}^n(\mathbf{x}, \hat{\mathbf{x}}, \hat{\mathbf{q}}) = \sum_a \int_0^{x_a} t_a(\omega, \hat{x}_a^n)\, d\omega + \sum_{rs} \int_0^{\hat{q}_{rs}} \left(\frac{1}{\theta} \ln \frac{\omega}{\bar{q}_{rs} - \omega} + \Psi_{rs}\right) d\omega$$

$$+ \sum_a \int_0^{\hat{x}_a} \hat{t}_a(\omega, x_a^n)\, d\omega \qquad [9.15a]$$

subject to

$$\sum_k f_k^{rs} = \bar{q}_{rs} - \hat{q}_{rs} \qquad \forall\, r, s \qquad [9.15b]$$

$$\sum_l \hat{f}_l^{rs} = \hat{q}_{rs} \qquad \forall\, r, s \qquad [9.15c]$$

$$f_k^{rs}, \hat{f}_l^{rs} \geq 0 \qquad \forall\, k, l, r, s \qquad [9.15d]$$

This program parallels program [9.7] except for the appropriate diagonalization of the automobile and transit link performance functions.

The mechanics of the diagonalization algorithm were described in Section 8.2. Following a determination of an initial solution, the subproblem (program [9.15] here) is solved for a new flow pattern $(\mathbf{x}^{n+1}, \hat{\mathbf{x}}^{n+1})$. The solution is then compared to the previous one. If

$$(\mathbf{x}^{n+1}, \hat{\mathbf{x}}^{n+1}) \simeq (\mathbf{x}^n, \hat{\mathbf{x}}^n) \qquad [9.16]$$

†If the wait time for transit is included in the transit link performance function, and if the operator's response is included in the analysis, note that the wait time may go down with increasing volume (due to the increased frequency). This may result in nonconvex equivalent minimization programs, which are difficult to solve.

the algorithm terminates. If this condition is not met, the performance curves are updated and a new diagonalized subproblem is formulated and solved. Note that the interacting variables in this problem are the link flows. Consequently, the O–D flows, $\{q_{rs}\}$, are never (temporarily) fixed at any intermediate value in any of the subproblems solved in the course of applying the diagonalization algorithm. Furthermore, the convergence criterion can be based solely on the similarity of the link flows. Nevertheless, when the algorithm terminates, the solution $(\mathbf{x}, \hat{\mathbf{x}}, \hat{\mathbf{q}})$ satisfies all the equilibrium conditions. The proof of this statement follows the same logic as that used in Section 8.2 to prove the validity of the diagonalization algorithm in the case of the UE problem (see Problem 9.6). A convergence proof for this algorithm is given in the references discussed in Section 9.4. These references also discuss the conditions for uniqueness of the solution, which parallel the Jacobian condition discussed in Chapter 8.

In solving each subproblem, the direction-finding step is carried out in a manner analogous to the solution of program [9.7]. In other words, the network is decomposed by O–D pair and criterion [9.11] is used to decide whether $\bar{q}_{rs}$ is assigned to the minimum path through the transit network or the minimum path through the auto network.

The convergence considerations discussed in Section 8.2 apply to this algorithm as well. Consequently, instead of using the original version of the diagonalization algorithm, it is more efficient to use the streamlined approach discussed in that section. The use of this algorithm is demonstrated below.

Example. Consider the simple network shown in Figure 9.2. It consists of one origin–destination pair connected by one bus link and one automobile link. The performance functions are given by

$$t = 2 + 4x + \hat{x} \qquad [9.17a]$$

for the auto link, and

$$\hat{t} = 4 + 3\hat{x} + 2x \qquad [9.17b]$$

for the transit link. The O–D flow is

$$x + \hat{x} = 5 \qquad [9.17c]$$

This example is mathematically equivalent to the simple example considered in Section 8.2 (with x replacing x_1 and $\hat{x}$ replacing x_2—see Eqs. [8.37]). If travelers were assumed to choose the least travel time mode (i.e., the shortest

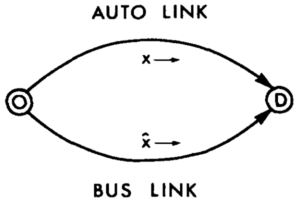


Figure 9.2 Modal split/traffic assignment supernetwork. Each network (automobile and transit) includes only one link. The data for this problem are given in Eqs. [9.17].

path) from the origin to the destination, the equilibrium flow pattern for this example would have been in accordance with a set of UE-type conditions. The resulting equilibrium flows would have then been $x^* = 3$ and $\hat{x}^* = 2$ (with $t = \hat{t} = 16$). Assume, however, that the equilibrium split between the two modes is given by a logit formula with parameters $\theta = 1.0$ and $\Psi = 0$, that is, at equilibrium

$$\frac{\hat{x}}{5} = \frac{1}{1 + e^{(\hat{t} - t)}} \qquad [9.17d]$$

This simple example can be solved analytically by substituting Eqs. [9.17a] and [9.17b] for t and $\hat{t}$ in [9.17d] and then substituting $x = 5 - \hat{x}$ to obtain the following equation in $\hat{x}$:

$$\frac{\hat{x}}{5} = \{1 + \exp[4 + 3\hat{x} + 2(5 - \hat{x}) - 2 - 4(5 - \hat{x}) - \hat{x}]\}^{-1}$$

The solution of this equation is $\hat{x}^* = 2.1$,† implying that $x^* = 2.9$. The equilibrium travel times are $t^* = 15.748$ and $\hat{t}^* = 16.083$.

This example can now be solved by applying the streamlined algorithm, starting from the initial solution $x^0 = 1.0$, $\hat{x}^0 = 4.0$. [The initial solution (0, 5) cannot be used here because of the logarithms in the objective function.] The first iteration of this algorithm is as follows (see the description of the streamlined algorithm in Section 8.2 and the description of the solution of program [9.7] given in Eqs. [9.9] through [9.12]):

Step 1: *Travel-time update*

$$t^0 = 2 + 4 \cdot 1 + 4 = 10$$

$$\hat{t}^0 + \frac{1}{\theta} \ln \frac{\hat{x}^0}{5 - \hat{x}^0} = 4 + 3 \cdot 4 + 2 \cdot 1 + \ln \frac{4}{5 - 4} = 19.386$$

Step 2: *Direction finding.* Since

$$t^0 < \hat{t}^0 + \frac{1}{\theta} \ln \frac{\hat{x}^0}{5 - \hat{x}^0} \qquad y^0 = 5 \quad \text{and} \quad \hat{y}^0 = 0$$

Step 3: *Move-size determination*

$$\min_{0 \leq \alpha \leq 1} \int_0^{1+4\alpha} (6 + 4\omega)\, d\omega + \int_0^{4-4\alpha} \ln \frac{\omega}{5 - \omega}\, d\omega + \int_0^{4-4\alpha} (6 + 3\omega)\, d\omega$$

The solution is,

$$\alpha = 0.295$$

Step 4: *Updating.* $\hat{x} = 2.82$, $x = 2.18$.

Step 5: *Convergence test.* The convergence criterion of flow similarity is not met. Go to step 1.

†The actual solution is $\hat{x}^* = 2.083984$.

TABLE 9.1 Algorithmic Iterations for the Network in Figure 9.2

Iteration	x^n	$\hat{x}^n$	$\max_a \left\{ \left\| \frac{x_a^{n+1} - x_a^n}{x_a^n} \right\| \right\}$	t^n	$\hat{t}^n + \ln \frac{\hat{x}^n}{5 - \hat{x}^n}$	y^n	$\hat{y}^n$	α_n	$\max_a \left\{ \left\| \frac{x_a^n - x_a^*}{x_a^*} \right\| \right\}$
0	1	4		10	19.386	5	0	0.2949	0.9194
1	2.1797	2.8203	1.1797	13.539	17.078	5	0	0.1608	0.3533
2	2.6332	2.3668	0.2081	14.900	16.260	5	0	0.0736	0.1357
3	2.8075	2.1925	0.0736	15.422	15.945	5	0	0.0305	0.0521
4	2.8744	2.1256	0.0305	15.623	15.824	5	0	0.0121	0.0200
5	2.9000	2.1000	0.0121	15.700	15.777	5	0	0.0047	0.0077
6	2.9099	20.0901	0.0047	15.730	15.759	5	0	0.0018	0.0029
7	2.9137	2.0863	0.0018	15.741	15.752	5	0	0.0007	0.0011
	2.9151	2.0849	0.0007						0.0004

The remaining algorithmic iterations are shown in Table 9.1. As this table shows, the convergence rate is asymptotic, as expected.

Non-UE Transit Assignment

The assumption made thus far in Section 9.1 is that the UE conditions can be applied to a transit network. These conditions result from the assumption that each transit traveler chooses the shortest-travel-time route (over the transit network) from origin to destination.

Empirical studies show, however, that travel time may not be the major determinant of transit route selection. Other factors, especially waiting time for the bus (or train)† and the need to make a transfer, may be very important in determining which route is chosen. Many of these factors cannot be easily modeled as link impedances and therefore cannot be analyzed in a framework which is based on selecting the minimum-impedance path. For example, consider a situation in which transit patrons can use several bus lines to get from a given origin to a given destination. If these patrons try to minimize their waiting time, some of them may take the first bus that comes along, regardless of the in-vehicle travel time. The implication of such behavior is that even without congestion phenomena, not all transit patrons between the origin and the destination will necessarily use the same single route. If the bus lines are scheduled with no consistent pattern relative to each other, and if patrons arrive randomly at the origin bus stop, the O–D flow assigned to each line will be directly proportional to its frequency. Such an assignment, or network loading rule, would not generate a UE-type situation over the transit network.

As a result, some transit assignment models do not use the UE criterion to assign the transit trips onto the network. For example, the Volvo Bus Transit Planning Model (see reference in Section 9.4) assumes that all passengers would use a no-transfer path from their origin to their destination if

†Empirical studies (some of which are referenced in Section 13.5) indicate that waiting time is two to three times more important than the in-vehicle time in determining mode choice.

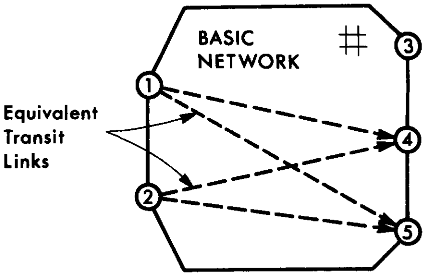


Figure 9.3 Supernetwork for a joint modal split/traffic assignment problem with non-UE transit assignment model.

available. Only if no such path is available would a one-transfer path be considered. If such a connection of two bus lines is not available, two-transfer paths are then used, and so on. When more than one option is available within a category (e.g., no transfer), the trips are distributed between the routes on the basis of the relative bus frequencies.

Transit assignment models such as Volvo's cannot be handled easily using equivalent mathematical programming methods and network representation. The problem addressed here is one of integrating these approaches with a UE model over the auto network, given that the transit travel times may be influenced by both transit and auto flows.

Such an integration can be accomplished by treating the transit network exogenously. The effects of flows in this network on the auto network will be represented by direct O–D equivalent transit links as shown in Figure 9.3. The flow on each of these links is $\hat{q}_{rs}$. This representation is similar in appearance to the supernetwork used to model the simplified mode split/traffic assignment problem in Section 6.4.

Assuming a logit-based modal split function, the "travel time" over each of the aforementioned equivalent transit links, $\hat{t}_{rs}$, is given by the "performance function"

$$\hat{t}_{rs}(\hat{q}_{rs}, \mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{\theta} \ln \frac{\hat{q}_{rs}}{\bar{q}_{rs} - \hat{q}_{rs}} + \Psi_{rs} + \hat{u}_{rs}(\mathbf{x}, \hat{\mathbf{x}}) \qquad \forall\, r, s \qquad [9.18]$$

where $\hat{u}_{rs}(\mathbf{x}, \hat{\mathbf{x}})$ is the minimum travel time over the transit network when the automobile flows are $\mathbf{x}$ and the transit flows are $\hat{\mathbf{x}}$.

Consider, now, the application of the diagonalization algorithm to the equilibrium over the above-mentioned supernetwork.† The (diagonalized) subproblem that has to be solved at every iteration of this algorithm is

$$\min z(\mathbf{x}, \hat{\mathbf{q}}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega + \sum_{rs} \int_0^{\hat{q}_{rs}} \left[\frac{1}{\theta} \ln \frac{\omega}{\bar{q}_{rs} - \omega} + \Psi_{rs} + \hat{u}_{rs}(\mathbf{x}^n, \hat{\mathbf{x}}^n) \right] d\omega \qquad [9.19a]$$

†The following discussion assumes, for clarity of presentation, that automobile travel times are not affected by transit flows. As mentioned before, this is a reasonable approximation. (It is still assumed that the transit travel times are affected by the automobile flows.)

subject to

$$\sum_k f_k^{rs} + \hat{q}_{rs} = \bar{q}_{rs} \qquad \forall\ r, s \qquad [9.19b]$$

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [9.19c]$$

$$0 < \hat{q}_{rs} < \bar{q}_{rs} \qquad \forall\ r, s \qquad [9.19d]$$

where the diagonalization principle is reflected in $\hat{u}_{rs}(\mathbf{x}^n, \hat{\mathbf{x}}^n)$ being a constant in program [9.19]†. When the streamlined version of the diagonalization algorithm is applied to the problem at hand, the transit link flows may be obtained by *any* transit assignment model. The following steps have to be executed in order to apply this algorithm to the supernetwork under consideration‡:

Step 0: *Initialization.* Obtain an initial set of link flows over the supernetwork. These include the auto flows $\{x_a^0\}$, and the flow over the dummy transit links, $\{\hat{q}_{rs}^0\}$. Also obtain an initial transit link flow, $\{\hat{x}_b^0\}$.§ Set $n := 0$.

Step 1: *Travel-time update.* Set

$$t_a^n = t_a(x_a^n) \qquad \forall\ a \in \mathscr{A}$$

$$\hat{t}_b^n = \hat{t}_b(\hat{\mathbf{x}}^n, \mathbf{x}^n) \qquad \forall\ b \in \hat{\mathscr{A}}$$

Compute

$$\hat{u}_{rs}^n = \hat{u}_{rs}(\mathbf{x}^n, \hat{\mathbf{x}}^n) \qquad \forall\ r, s$$

Step 2: *Network assignment.* Perform an all-or-nothing assignment over the modified network (i.e., assign $\bar{q}_{rs}$ to this supernetwork).|| This yields a new link-flow pattern, $\{y_a^n\}$ over the auto network and $\{\hat{v}_{rs}^n\}$ over the direct O–D links representing transit.

Step 3: *Move-size determination.* Find an α_n that solves

$$\min_{0 \leqslant \alpha \leqslant 1} z(\alpha) = \sum_a \int_0^{x_a^n + \alpha(y_a^n - x_a^n)} t_a(\omega)\, d\omega$$
$$+ \sum_{rs} \int_0^{\hat{q}_{rs}^n + \alpha(\hat{v}_{rs}^n - \hat{q}_{rs}^n)} \left(\frac{1}{\theta} \ln \frac{\omega}{\bar{q}_{rs} - \omega} + \Psi_{rs} + \hat{u}_{rs}^n\right) d\omega$$

†Note that constraint [9.19d] is automatically satisfied at the solution of program [9.19] (see program [6.28] and the related discussion).

‡The notations used in this description parallel the ones introduced in conjunction with the discussion of mode split with full transit networks (see Eqs. [9.1] and [9.2] and the related discussion).

§This is calculated exogeneously using an appropriate transit assignment procedure.

||The equivalent travel times over the dummy transit links are given by Eq. [9.18].

Step 4: *Flow update.* Set

$$x_a^{n+1} = x_a^n + \alpha_n(y_a^n - x_a^n) \qquad \forall\ a$$

$$\hat{q}_{rs}^{n+1} = \hat{q}_{rs}^n + \alpha_n(\hat{v}_{rs}^n - \hat{q}_{rs}^n) \qquad \forall\ r, s$$

Assign $\hat{q}_{rs}^{n+1}$ to the transit network using the transit assignment rule. This yields $\hat{\mathbf{x}}^{n+1}$.

Step 5: *Convergence criterion.* If $(\mathbf{x}^{n+1}, \hat{\mathbf{q}}^{n+1}) \simeq (\mathbf{x}^n, \hat{\mathbf{q}}^n)$, stop. Otherwise, set $n := n + 1$ and go to step 1.

Note that the term $u_{rs}^n(\cdot, \cdot)$ is not part of the line search in step 3 (i.e., it is not a function of any of the flow variables) since it is fixed by the diagonalization process.

This program will have a unique solution if the appropriate conditions mentioned in Section 8.2 hold for the supernetwork under consideration.† The first of these conditions requires each link performance function to be an increasing function of its flow. This requirement naturally holds for the basic (auto) links and should hold for the dummy transit links as well, regardless of the transit network assignment rule used. The second part of the uniqueness conditions requires that each link be influenced primarily by its own flow. This condition can be expected to hold over the basic (auto) network due to the characteristics of the link interactions mentioned in Section 8.2. Regarding the dummy transit links, this condition may not always hold since the marginal effect of an automobile trip may be higher than the marginal effect of a transit trip on the transit O–D equivalent travel times. In other words, it may be the case that the requirement that

$$\frac{\partial \hat{t}_{rs}(\hat{q}_{rs}, \mathbf{x}, \hat{\mathbf{x}})}{\partial \hat{q}_{rs}} \gg \frac{\partial \hat{t}_{rs}(\hat{q}_{rs}, \mathbf{x}, \hat{\mathbf{x}})}{\partial x_a} \qquad \forall\ r, s, a \tag{9.20}$$

does not hold.‡ In such cases the equilibrium may not be unique.

When the streamlined diagonalization algorithm is executed as described above, it will converge to an equilibrium flow pattern. At that point, the travel times and flows over the basic network would satisfy the UE conditions, with the relative transit and automobile flows determined by a logit model, and the flows over the transit network derived from the model used for transit assignment.§

†As mentioned in that section, the uniqueness condition can be summarized as a requirement on the positive definiteness of the link-travel-time Jacobian. In this case the Jacobian is defined for the modified network.

‡Note that if these uniqueness conditions do not hold, it does not necessarily mean that the problem does not have a unique solution. It only means that uniqueness cannot be guaranteed a priori.

§The proof of this statement follows the lines of the proof given in Section 8.2 regarding the validity of the diagonalization algorithm. The only difference is that the proof, in this case, applies to the modified network (supernetwork), with program [9.19] being solved as the subproblem.

9.2 JOINT TRAVEL DECISIONS AND SUPERNETWORKS

This section brings together many of the ideas presented in Parts II and III. It outlines a framework for jointly solving several travel decision models, accounting for equilibrium effects. The travel decisions that can be included in this framework are the decisions of whether to travel (during the design period), by which mode, where to go, and what route to take. As mentioned in the introduction to this chapter, these decisions correspond to the four steps of the conventional urban transportation planning process, which, for the most part, have been modeled separately in practice. The aggregation of all individual decisions associated with taking a trip is usually captured in "trip production" or "trip generation" models. Such models relate the general level of accessibility of each origin (zone or centroid) to the number of trips originating there. The variable-demand models discussed in Chapter 6 can be viewed as a special version of trip generation models. The mode choice and destination choice problems have also been traditionally modeled independently of each other and independently of the route-choice decisions. Sections 6.4 and 9.1 show how mode choice can be modeled simultaneously with the traffic assignment, and Chapter 7 presented joint distribution/assignment models. This section demonstrates how all these models can be integrated in a joint formulation and solved simultaneously using the supernetwork concept.

Joint Modal Split/Distribution/Assignment Model

This part of Section 9.2 describes a combined modal split/trip distribution/traffic assignment model. The model assumes that at equilibrium, flows and travel times over the basic network will satisfy the UE equations, and that both trip distribution and modal split will be given by logit-based demand functions.† The transit network is assumed to be independent of the automobile network and the transit O–D travel times are assumed fixed (i.e., not flow dependent) for the purposes of this exposition. In other words, at equilibrium, the flow of automobile users between origin r and destination s, q_{rs}, will be given by (see Eq. [9.5])

$$q_{rs} = \bar{q}_{rs} \frac{1}{1 + e^{\theta(u_{rs} - \hat{u}_{rs} - \Psi_{rs})}} \qquad [9.21a]$$

where θ and Ψ_{rs} are parameters of this logit model, and $\bar{q}_{rs}$ is the total (automobile + transit) flow between r and s. The transit flow is $\hat{q}_{rs}$, where $\hat{q}_{rs} = \bar{q}_{rs} - q_{rs}$.

†Other models may also be used in this context. Note only that the trip distribution model has to satisfy the trip production constraint (see Section 7.1) and a share model is therefore convenient to use for this purpose. The logit formula was chosen to exemplify the concepts given here due to its wide-spread use and flexibility (e.g., any share model can be represented as a generalized logit equation).

Unlike the model discussed in Section 9.1, the total O–D flow, $\bar{q}_{rs}$, is not constant in this particular model. This flow is given by a logit-based trip distribution model, that is,

$$\bar{q}_{rs} = O_r \frac{e^{-\gamma(u_{rs} - M_{rs})}}{\sum_m e^{-\gamma(u_{rm} - M_{rm})}} \qquad [9.21b]$$

where γ is a parameter, M_{rs} measures the attraction of destination s to trips from r,† and O_r is the total flow originating from node r.

Equation [9.21b] implies that the distribution of flow to the various destinations is determined by the automobile travel time to these destinations, as well as some fixed parameters. Since the transit travel times, $\{\hat{u}_{rs}\}$, are assumed to be flow independent, these times can be treated as constants in the distribution model and be included in the attraction measures, $\{M_{rs}\}$. The distribution model will then be

$$\bar{q}_{rs} = O_r \frac{e^{-\gamma(u_{rs} + \hat{u}_{rs} - M'_{rs})}}{\sum_m e^{-\gamma(u_{rm} + \hat{u}_{rm} - M'_{rm})}} \qquad \forall\ r, s \qquad [9.22]$$

where $M'_{rm} = M_{rm} + \hat{u}_{rm}\ \forall\ r, m$ (see Eq. [9.21b]). It is not clear, however, that trip distribution should depend on the sum of the transit and automobile travel times. If the effect of these times on trip distribution is not identical, they may be scaled by different parameters, or a different functional form may be used in the trip distribution equations.‡ For simplicity of presentation, the following discussion assumes that the trip distribution model is given by Eq. [9.21b]. Similar models can be developed for other specifications of the trip distribution equation (see Problem 9.14).

The supernetwork that can be used to solve this modal split/trip distribution/traffic assignment problem includes two types of links in addition to those of the basic network. First, all destinations are connected to a virtual (dummy), origin–specific destination node, r', by dummy links. This representation parallels the augmented network illustrated in Figure 7.2. The flow on the dummy link connecting destination s to the dummy destination r' is $\bar{q}_{rs}$. The equivalent performance functions on these links are given by (see Eq. [7.22])

$$t_{sr'} = \frac{1}{\gamma} \ln \bar{q}_{rs} - M_{rs} \qquad \forall\ r, s \qquad [9.23a]$$

†If the attraction at destination s is not related to origin characteristics, the attraction measure, M_{rs}, can be indexed by s only, as in Section 7.2.

‡For example, given that the mode split is given by a logit formula such as Eq. [9.21a], the terms $(u_{rm} + \hat{u}_{rm})$ in Eq. [9.22] could be replaced by the function $-(1/\theta) \ln (e^{-\theta(\hat{u}_{rm} + \Psi_{rm})} + e^{-\theta u_{rm}})$. This nonlinear function of u_{rm} and $\hat{u}_{rm}$ captures the combined level of service (of transit and automobile) between r and m under these circumstances. The rationale behind this particular specification is beyond the material presented thus far with regard to logit models. Chapter 10 explains the concept of satisfaction, which bears directly on this formulation.

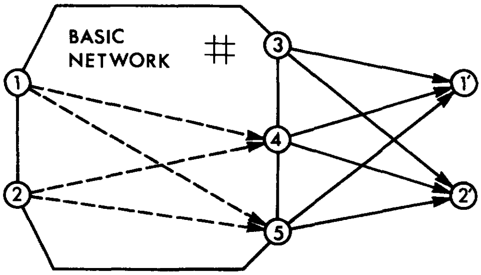


Figure 9.4 Supernetwork for a joint mode split/trip distribution/traffic assignment problem.

The transit mode is represented, in this supernetwork, by direct origin-to-destination links as in the last part of Section 9.1. The flow on the transit link connecting O–D pair r–s is $\hat{q}_{rs}$, and the equivalent performance function on each of these links is given by (see Eq. [9.18])†

$$\hat{t}_{rs}(\hat{q}_{rs}) = \frac{1}{\theta} \ln \frac{\hat{q}_{rs}}{\bar{q}_{rs} - \hat{q}_{rs}} + \Psi_{rs} + \hat{u}_{rs} \qquad \forall\ r, s \qquad [9.23b]$$

The supernetwork corresponding to this representation is shown in Figure 9.4. In this example, the network includes two origins and three destinations. Origin nodes 1 and 2 are connected to destination nodes 4 and 5 by direct transit links, shown as dashed lines in the figure. In addition, the destinations are connected to origin-specific dummy destination nodes.

The formulation of the modal split/distribution/assignment equilibrium problem as an equivalent mathematical program is not straightforward. The reason is that the equivalent performance function on the dummy transit links (given in Eq. [9.23b]) includes two types of flow variables. The first one is the transit flow, $\hat{q}_{rs}$, and the second one is the total O–D flow, $\bar{q}_{rs}$, which is a *variable* in this problem. Thus, even if transit level of service is assumed to be independent of automobile flow, the equivalent transit travel time, $\hat{t}_{rs}$, is a function of the flow over the dummy link sr' in addition to the transit flow (i.e., it is a function of both $\hat{q}_{rs}$ and $\bar{q}_{rs}$). Consequently, link independence cannot be assumed. Furthermore, this interaction is not symmetric, as is evident from Eqs. [9.23a] and [9.23b], and this problem cannot, therefore, be formulated as a minimization program.

The equilibrium flows can be determined, however, by using the diagonalization algorithm. This algorithm requires iterative solutions of a mathematical programming subproblem, formulated by fixing the effects of "foreign flows" on the travel time (or equivalent travel time) of each link. The iterations continue until the solutions of two consecutive subproblems are judged to be sufficiently similar. When the supernetwork under consideration is diagonalized, the subproblem that has to be solved at the nth iteration of the

†For simplicity of presentation, this formulation assumes that transit travel times are independent of both transit and automobile flows. The model can be generalized to include these interactions.

algorithm is the following:

$$\min z(\mathbf{x}, \hat{\mathbf{q}}, \bar{\mathbf{q}}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega + \sum_{rs} \int_0^{\hat{q}_{rs}} \left(\frac{1}{\theta} \ln \frac{\omega}{\bar{q}_{rs}^n - \omega} + \Psi_{rs} + \hat{u}_{rs} \right) d\omega$$

$$+ \sum_{rs} \int_0^{\bar{q}_{rs}} \left(\frac{1}{\gamma} \ln \omega - M_{rs} \right) d\omega \qquad [9.24a]$$

subject to

$$\sum_k f_k^{rs} = \bar{q}_{rs} - \hat{q}_{rs} \qquad \forall\, r, s \quad (u_{rs}) \qquad [9.24b]$$

$$\sum_s \bar{q}_{rs} = O_r \qquad \forall\, r \quad (\lambda_r) \qquad [9.24c]$$

$$f_k^{rs} \geq 0 \qquad \forall\, k, r, s \qquad [9.24d]$$

$$0 < \hat{q}_{rs} < \bar{q}_{rs} \qquad \forall\, r, s \qquad [9.24e]$$

This program is formulated in terms of the total O–D flows, $\bar{\mathbf{q}} = (\ldots, \bar{q}_{rs}, \ldots)$, and the transit flows, $\hat{\mathbf{q}} = (\ldots, \hat{q}_{rs}, \ldots)$, in addition to the link flows over the basic network. It can also be expressed explicitly in terms of **x**, $\hat{\mathbf{q}}$, and **q**. The diagonalization principle is reflected in this formulation by fixing $\bar{q}_{rs}$ at its level at the *n*th iteration, $\bar{q}_{rs}^n$, in the second sum in Eq. [9.24a].

Program [9.24] is a straightforward program, the first-order conditions of which are†

$$(c_k^{rs} - u_{rs}) f_k^{rs} = 0 \qquad \forall\, k, r, s \qquad [9.25a]$$

$$c_k^{rs} - u_{rs} \geq 0 \qquad \forall\, k, r, s \qquad [9.25b]$$

$$\frac{1}{\theta} \ln \frac{\hat{q}_{rs}}{\bar{q}_{rs}^n - \hat{q}_{rs}} + \Psi_{rs} + \hat{u}_{rs} - u_{rs} = 0 \qquad \forall\, r, s \qquad [9.25c]$$

$$\frac{1}{\gamma} \ln \bar{q}_{rs} - M_{rs} + u_{rs} - \lambda_r = 0 \qquad \forall\, r, s \qquad [9.25d]$$

and the original constraints [9.24b] through [9.24e].

The variables in this equation should be superscripted by $n + 1$ to indicate that they stand for the solution of the diagonalized subproblem [9.24]. Recall, however, that $\bar{q}_{rs}$ in Eq. [9.25c] is superscripted by n due to the diagonalization. When the diagonalization algorithm converges, $\bar{q}_{rs}^n \simeq \bar{q}_{rs}^{n+1}$, indicating an (approximate) equilibrium solution.

Equations [9.25a] and [9.25b] specify the UE conditions over the basic network. Equation [9.25c] can be rewritten (given that $\bar{q}_{rs}^{n+1} \simeq \bar{q}_{rs}^n$) as

$$\hat{q}_{rs} = \bar{q}_{rs} \frac{1}{1 + e^{\theta(\hat{u}_{rs} - u_{rs} + \Psi_{rs})}} \qquad [9.26]$$

†As mentioned before (see program [9.19] and the related footnote), constraint [9.24e] can be ignored in the derivation of the first-order conditions since it will be satisfied automatically at the minimum (see Eq. [9.26a]).

where the superscript denoting the iteration number is omitted for clarity. This equation means that the transit share of flow is determined by a logit model, as required by Eq. [9.21a]. Condition [9.25d] can be rewritten as

$$\bar{q}_{rs} = e^{\gamma \lambda_r} e^{-\gamma(u_{rs} - M_{rs})} \qquad \forall\ r, s \tag{9.27a}$$

The trip production constraints [9.24c] should also hold and thus

$$\sum_m \bar{q}_{rm} = e^{\gamma \lambda_r} \sum_m e^{-\gamma(u_{rm} - M_{rm})} \tag{9.27b}$$

Equation [9.27a] can be divided by Eq. [9.27b] to give

$$\frac{\bar{q}_{rs}}{\sum_m \bar{q}_{rm}} = \frac{e^{-\gamma(u_{rs} - M_{rs})}}{\sum_m e^{-\gamma(u_{rm} - M_{rm})}} \qquad \forall\ r, s \tag{9.28}$$

which is a multinomial logit model with parameter γ.

The dual variable λ_r plays the same role in this model that u_{rs} plays in the standard UE model of a basic network with fixed demand, analyzed in Section 5.2. It represents the travel time on the minimum path (through the supernetwork) connecting origin r to the dummy destination node r'. In other words, at equilibrium the equivalent travel time between node r and node r' is the same on all routes used between these two nodes. To see this, note that there are two types of paths connecting node r and r' in the supernetwork under consideration. The first one goes through the basic network and a dummy link sr'. The travel time over this path is

$$u_{rs} + \left(\frac{1}{\gamma} \ln \bar{q}_{rs} - M_{rs}\right) \tag{9.29a}$$

At equilibrium, the equivalent travel time over this path equals λ_r, as indicated by Eq. [9.25d].

The other type of path from r to r' goes through the transit link from r to some destination node s, and then through a dummy link from s to r'. The travel time on this path is

$$\left(\frac{1}{\theta} \ln \frac{\hat{q}_{rs}}{\bar{q}_{rs} - \hat{q}_{rs}} + \Psi_{rs} + \hat{u}_{rs}\right) + \left(\frac{1}{\gamma} \ln \bar{q}_{rs} - M_{rs}\right) \tag{9.29b}$$

This expression also equals λ_r, as can be verified by simply adding equations [9.25c] and [9.25d]. Thus λ_r is the minimum "travel time" between origin, r, and the virtual destination, r', through the supernetwork under consideration.

The diagonalization algorithm can be applied iteratively to this supernetwork, solving subproblem [9.24] at each iteration. Each subproblem can be solved by using the convex combinations algorithm. Such iterative applications would converge to the equilibrium solution of the modal split/trip distribution/traffic assignment problem. The proof that the point of convergence corresponds to the equilibrium flow pattern could follow the same lines as the proof given in Section 8.2. In view of the supernetwork representation, however, this effort may be superfluous since a proof that applies to a simple basic network with fixed demand also applies to a supernetwork. Furthermore, the streamlined diagonalization algorithm described in Section 8.2 can be used to solve this problem instead of the original version of this algorithm, for faster convergence.

When the streamlined diagonalization algorithm is applied to this problem, each iteration involves the determination of the minimum (equivalent) travel-time path through the supernetwork from each origin, r, to each virtual destination, r'. The problem can, therefore, be decomposed by origin. When each origin is considered, the aforementioned minimum path is assigned O_r flow units. When all origins have been assigned, the resulting intermediate flow pattern ($\mathbf{y}^n$, $\hat{\mathbf{v}}^n$, $\bar{\mathbf{v}}^n$) is used to define a descent direction and compute the flow pattern for the next iteration.

To expedite convergence, it may be more efficient to use a multistage algorithm (similar to the double-stage algorithm discussed in Chapter 7) to solve subproblem [9.24]. Instead of finding the search direction by loading O_r flow units to the shortest path between r and r', it may be more efficient to compute the minimum-travel-time path over the automobile and transit networks separately, apply the distribution models to find $\bar{v}_{rs}$ for every O–D pair r–s, apply the modal split model to find v_{rs} and $\hat{v}_{rs}$, and then assign $\{v_{rs}\}$ to the automobile network to find the set of intermediate flows, $\{y_a^n\}$. The remaining steps of the algorithm would be identical to those of the convex combinations method.

This multistage approach can also be used in conjunction with the streamlined diagonalization algorithm. In this case each iteration will be executed as a single multistage step and the flows will be updated after every iteration.

The diagonalization approach can also be used with a doubly constrained trip distribution model (see Section 7.3). Such a model would include the additional constraints

$$\sum_r \bar{q}_{rs} = D_s \qquad \forall\ s$$

in program [9.24]. This implies that if the convex combinations algorithm is used to solve the subproblems, a Hitchcock transportation problem would have to be solved at every iteration of this algorithm (see Section 7.4).

Joint Modal Split/Distribution/Assignment Model with Variable Demand

As mentioned in the introduction to Section 9.2, the framework developed in this text can be used to solve problems which include trip generation, modal split, trip distribution, and traffic assignment. Such a problem would be similar to the one addressed above (see Eqs. [9.21]) except that O_r, the total number of trips originating from each node r, would not be given exogenously. Instead, it is replaced by a special demand functon that relates total trip production to the level of service that travelers face over the network. The trip generation (or trip production) model for origin r is

$$O_r = D_r(\lambda_r) \qquad \forall\ r \tag{9.30}$$

The argument of the trip generation function,† λ_r, is the equivalent travel time between origin r and the dummy destination node, r', in the supernetwork representation of the modal split/distribution/assignment problem (see Eqs. [9.29]).

Combining the concept described in Sections 6.2 and 6.3 for user equilibrium problems with variable demand along with the diagonalization approach, the joint trip generation/modal split/trip distribution/traffic assignment problem can be solved by adding one term to the objective function of program [9.24]. This term is (see Eq. [6.1a])

$$-\sum_{rs} \int_0^{O_r} D_r^{-1}(\omega)\, d\omega \tag{9.31}$$

The resulting subproblem can, again, be solved by an application of the convex combinations method. Note that this requires that an upper bound constraint be added to the set of constraints of program [9.24]. The constraint is (see Eq. [6.10c] and the related discussion)

$$O_r \leq \bar{O}_r \qquad \forall\ r \tag{9.32}$$

where $\bar{O}_r$ is a (large enough) constant. The application of the algorithm to each subproblem parallels the solution of the variable-demand problem presented in Section 6.2.

In a fashion similar to network representation associated with the excess demand formulation described in Section 6.3, the supernetwork concept can be extended to include the trip generation model as well. To do this, the supernetwork depicted in Figure 9.4 should be modified by adding dummy links leading from each origin, r, to the associated dummy destination node, r', as shown in Figure 9.5. Letting $e_r = \bar{O}_r - O_r$, the performance function for these dummy links, $W_r(\cdot)$, will be given by (see Eqs. [6.23])

$$W_r(e_r) = D_r^{-1}(\bar{O}_r - e_r) \tag{9.33}$$

In this formulation, $\bar{O}_r$ can be interpreted as the total potential flow out of node r. It includes both travelers and nontravelers and may equal, for example,

†The demand function shown in Eq. [9.30], $D_r(\cdot)$, can be calibrated from observations of O_r and λ_r over an existing urban network. Calibration and model estimation issues are presented in Chapter 13. At this point, it is only important to realize that λ_r can be measured, given that the trip distribution and modal split models are known, by using Eqs. [9.29]. Note that a trip production function such as the one shown in Eq. [9.30] may include many other variables which characterize origin nodes, such as size, population, auto-ownership level, and so on. These variables, however, are exogenous to the equilibrium model. In addition, λ_r implicitly includes the variables that appear in the modal split and trip distribution models (see Eqs. [9.29] and the related discussion).

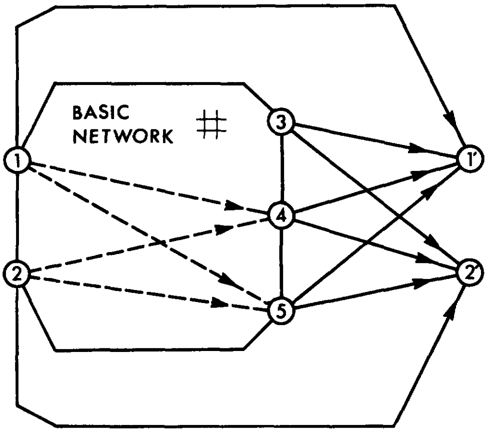


Figure 9.5 Supernetwork for a joint trip generation/mode split/trip distribution/traffic assignment program.

the entire population of the traffic zone represented by node r. The argument of $W_r(\cdot)$, e_r, is the flow over the excess flow link $r \rightarrow r'$; it is, in fact, the "flow of nontravelers" out of node r.

Analysis of Supernetworks

The supernetworks presented earlier in this section are two examples of joint analysis of several travel choice dimensions. Actually, most of the network equilibrium models addressed in this book and elsewhere could be solved using supernetworks of various complexity.

The simplest "supernetwork" includes only the basic automobile network. This standard UE problem, analyzed in Chapter 5, requires the specification of a complete O–D trip matrix. Such a specification assumes that the number of auto trips between each O–D pair is fixed (and known) and is not influenced by the transportation level of service. This basic framework can be extended in several directions, three of which are described in this text. These include models of equilibrium assignment with variable demand, modal split, or trip distribution. The variable-demand formulation presented in Chapter 6 assumes that the O–D flows are given by a function that depends on the O–D level of service. The problem can be solved with the excess-demand formulation, which leads to a supernetwork representation. Instead of specifying the number of O–D trips, the variable-demand model requires the specification of the O–D demand function. In this model the total potential O–D trips is either assigned to the network, or is not assigned at all if the O–D travel time is too high.

The combined modal split/traffic assignment models discussed in Sections 6.4 and 9.1 assume that the total flow between each origin and each destination is fixed and known. The O–D flows on the basic network, however, depend on the split between auto and transit trips, as given by a special modal split function. Similarly to the variable-demand model, the number of trips between each O–D pair (and also the modal split) depends only on the level of service between the O–D under consideration, not on the level of service between the origin and other destinations. The solution methodology can be based on various supernetwork representations, depending on the transit assignment model used.

The "competition" between O–D pairs is modeled explicitly in the joint distribution/assignment model discussed in Chapter 7. In that chapter the focus is exclusively on the automobile network. The required inputs include the total number of vehicular trips originating at each node. The O–D trip rates are determined by the model, based on the level of service between each origin and all possible destinations. Again, this problem can be solved by using a supernetwork representation, as shown in Chapter 7.

In a joint modal split/trip distribution/traffic assignment model, the input constants include only the total number of trips originating at each node. The distribution of these trips between the various destinations and the split between transit and automobile trips are all determined endogenously by the model. These variables appear as flows on various components of an appropriate supernetwork. This model requires, however, that modal split and trip distribution functions be specified.

Unlike the aforementioned models, the trip generation/modal split/trip distribution/traffic assignment supernetwork model does not require any constant trip rates as input. Instead, it requires the specification of a trip generation model in addition to a mode split model and a trip distribution model.† All these models are then formulated as performance functions over dummy links of an appropriate supernetwork and the equilibrium problem is solved using the diagonalizaton algorithm. Each iteration of this algorithm requires the solution of an equivalent UE mathematical program over the supernetwork. The output of this analysis includes the flow of trips generated at each origin node, the flow between each origin and each destination, by mode, as well as the flow on each link of both the transit and auto networks. The output also includes the travel times on all the supernetwork links.

The principal advantage of a joint model of the travel decisions included in the supernetwork framework is that it may contribute to a more consistent analysis of the impact of transportation policies and designs. For example, if a policy of lowering transit fares is contemplated, its consequences may include higher transit volumes along with an increase in the flow of people going to destinations served by transit. Since these travelers will not be using their automobiles, the flow pattern over the basic network will change, thereby influencing the automobile level of service. These changes may, in turn, induce travel between some O–D pairs and reduce travel between others, possibly affecting the whole activity system in the area under study. The problem formulation given in this chapter can be used either as a conceptual guide for looking at the problem as a whole or as an actual problem solving tool. In

†In addition, all the models reviewed here require the specification of the (transit and auto) networks, including all the link performance curves.

either case it may contribute to a consistent prediction of the consequence of many transportation-related projects in urban areas.

The possible drawbacks associated with the supernetwork approach include the following:

1. The computational effort associated with such an analysis is higher than that required for nonequilibrium analysis, in which trip generation, modal split, and trip distribution are assumed to be independent of the transportation level of service (or the level of service is assumed independent of the flow).
2. If the supernetwork is used as an actual problem solving tool, the network structure implies certain restrictions on the demand functions used to model trip generation, modal split, and trip distribution. When these steps are modeled independently (not within an equilibrium framework), these demand functions can be more elaborate.

Many of the consideratons mentioned here can be reduced to a common yet difficult trade-off between accuracy and added realism, on the one hand, and the added costs associated with a more elaborate model, on the other. These trade-offs naturally come into play in almost every aspect of the analysis (including the level of detail of the network representation, the form of the demand and performance functions, the convergence criteria, the complexity as well as the inclusion or exclusion of submodels, and so on). The analysis offered in Part III sheds some light on what kind of models can be formulated and how such formulations can be solved. As noted throughout the book, there are many cases in which the resulting model may not be solvable, or it may not have a unique solution. In such cases a model may have to be modified or approximated by considering, again, all the above-mentioned trade-offs.

In conclusion, it is important to remember, that in most cases, the major component of the costs of studying urban transportation networks is not the analytical efforts, model building, or computer time, but the costs of obtaining the relevant data. In fact, in most cases data considerations are the driving force and the dominant factor in model selection and analysis style.

9.3 SUMMARY

The analysis framework described in Parts II and III is extended and generalized in this chapter to the joint treatment of several travel-choice dimensions. These dimensions include decisions of whether to travel or not, which mode to use, where to go, and what route to take. The flow pattern resulting from travelers' choices along all these dimensions can be analyzed as an equilibrium assignment problem over a supernetwork. The supernetwork includes the links and nodes representing physical facilities such as streets and intersections, as well as nodes and links representing the various travel choices. The performance curves on each of the supernetwork links may be interdependent. Consequently, the supernetwork should be solved using the (streamlined) diagonalization approach of Section 8.2.

Section 9.1 concentrates on a combined modal split/traffic assignment problem. It extends the results of Section 6.4 to include an explicit transit network representation and to mode split with link interactions. This case is especially important when the transit mode includes buses which move and interact with the general traffic. The last part of Section 9.1 demonstrates how this equilibrium framework can accommodate a transit assignment model that is not necessarily based on travel-time minimization.

The concept of a supernetwork is fully illustrated in Section 9.2, in which a supernetwork corresponding to a joint trip generation/mode split/trip distribution/traffic assignment problem is formulated. The trip generation, modal split, and the trip distribution models are given, at equilibrium, by appropriate demand functions, whereas flows and travel times over the automobile network satisfy the UE criteria.

9.4 ANNOTATED REFERENCES

Chapter 9 includes, for the most part, extensions of works and references mentioned previously. A proof of convergence for the mode split/traffic assignment diagonalization algorithm is given by Dafermos (1982). A combined model of trip mode choice, trip distribution, and traffic assignment was studied by Florian and Nguyen (1978) using the entropy concept. The concept of supernetwork and its uses in combining travel demand and network equilibrium models was studied extensively by Sheffi and Daganzo (1979, 1980) in their hypernetwork models. These authors use probit (rather than logit) models in their work. Probit models are discussed in Chapter 10, while the hypernetwork concept and algorithm are mentioned only briefly in Section 12.3.

As mentioned in Section 9.1, the supernetwork framework can accommodate transit assignment models that are not based on travel-time minimization. The Volvo Transit Assignment Model mentioned in Section 9.1 is described by Andersen (1977). Other approaches to transit network assignment are described by Dial and Bunyan (1968) and Han and Wilson (1982).

Section 9.2 refers to the four-step urban transportation planning process. This process is described in many references including the books by Hutchinson (1974), and Stoper and Meyburg (1975).

9.5 PROBLEMS

9.1. Show that the following program describes UE conditions over a network of transit and automobile links (with no flow interaction):

$$\min z(\mathbf{x}, \hat{\mathbf{x}}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega + \sum_b \int_0^{\hat{x}_b} \hat{t}_b(\omega)\, d\omega$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\, r, s \quad (u_{rs})$$

$$\sum_l \hat{f}_l^{rs} = \hat{q}_{rs} \qquad \forall\, r, s \quad (\hat{u}_{rs})$$

$$q_{rs} + \hat{q}_{rs} = \bar{q}_{rs} \qquad \forall\, r, s \quad (\tilde{u}_{rs})$$

$$f_k^{rs}, \hat{f}_l^{rs} \geq 0 \qquad \forall\, l, k, r, s$$

and the incidence relationships

$$x_a = \sum_{rs} \sum_k f_k^{rs} \delta_{a,k}^{rs}, \qquad c_k^{rs} = \sum_a t_a \delta_{a,k}^{rs}$$

$$\hat{x}_b = \sum_{rs} \sum_l \hat{f}_l^{rs} \hat{\delta}_{b,l}^{rs}, \qquad \hat{c}_k^{rs} = \sum_b \hat{t}_b \hat{\delta}_{b,k}^{rs}$$

Interpret each of the first-order conditions and the dual variables. Sketch the associated supernetwork.

9.2. (a) Show that the first-order conditions of program [9.7] specify the UE conditions over each mode's network and the logit mode split model between the modes.

(b) Investigate the first-order conditions of program [9.7] if the constraints are specified as in Problem 9.1. Show that both formulations have the same first-order conditions.

9.3. Consider the double-stage algorithm described in conjunction with the joint mode split/traffic assignment program (the discussion following Eq. [9.12]).

(a) Explain intuitively why this algorithm may be somewhat more efficient than the convex combinations algorithm for this problem.

(b) Show that the direction generated by this algorithm is a descent direction in terms of the appropriate equivalent minimization program (Eqs. [9.7]).

9.4. Formulate a supernetwork for the joint mode choice/traffic assignment problem given by program [9.7]. What are the computational difficulties associated with finding the UE flow pattern for this supernetwork?

9.5. Assume that the marginal effects of bus trips flow on automobile travel time equals the marginal effect of the automobile flow on the bus travel time [i.e., $\partial t_a(x_a, \hat{x}_a)/\partial \hat{x}_a = \partial \hat{t}_a(\hat{x}_a, x_a)/\partial x_a$]. Formulate the equivalent mathematical program and suggest an algorithmic solution procedure for the combined bus/automobile modal split/traffic assignment problem.

9.6. Prove that the diagonalization algorithm applied to subproblem [9.15] with convergence criterion [9.16] solves the joint problem of equilibrium assignment and logit mode choice with link interaction problem. (*Hint:* The proof is very similar to the one given in Section 8.2.)

9.7. Apply the streamlined diagonalization algorithm to a simple network consisting of an origin and a destination connected by an automobile and transit link (similar to Figure 9.2). The performance functions are given by

$$\hat{t} = 2 + \hat{x}^2 + x$$

$$t = 3 + 2x^2 + 2\hat{x}$$

and the O–D volume is $q = x + \hat{x} = 6$. The auto/transit modal split at equilibrium is given by the following logit function:

$$\frac{\hat{x}}{\hat{x} + x} = \frac{1}{1 + e^{0.5(\hat{t} - t)}}$$

Show that the iterations are converging toward the correct equilibrium point.

9.8. **(a)** Investigate the condition under which program [9.15] has a unique solution.
(b) Repeat part (a) for program [9.19].

9.9. Consider the network shown in Figure 9.2 with the following volume–delay function:

$$t_1 = 2 + x_1^2$$

$$t_2 = 1 + x_2 + 2x_2^2$$

Assume now that a bus service operates along the second link. The travel time on the bus is

$$t_b = 3 + x_b + x_2$$

where x_b is the flow of bus passengers. The total volume from r to s is $\bar{q}_{rs} = 6$ units of flow. The mode share between transit and automobile is given by the following logit model:

$$\frac{\hat{q}_{rs}}{\bar{q}_{rs}} = \frac{1}{1 + e^{1 + t_b - u_{rs}}}$$

where u_{rs} is the (auto) origin–destination travel time at equilibrium. The questions of interest involve the link (automobile and transit) flows at equilibrium.

(a) Does this problem have a unique solution? Explain.
(b) Can this problem be formulated as an equivalent minimization program? Explain.
(c) Apply an algorithmic procedure and solve for the equilibrium flows. Show the results of all the intermediate iterations. Check that the final solution complies with the equilibrium equations.

9.10. Explain the features of transit route selection which make it difficult to represent this process as a minimum-path choice.

9.11. Consider the problem of a joint mode split/traffic assignment problem with non-UE transit assignment. The transit travel times are assumed to be a function of both transit and automobile flows.

(a) How would you model the problem if the automobile travel times are a function of the flow of transit patrons (as well as the automobile flow)?
(b) How would the problem be solved if the automobile links interact with each other in addition to the transit–automobile interactions?

9.12. Program [9.19] can be solved with a double-stage algorithm (similar to the one explained in Chapter 7). Describe the application of such an algorithm in the context of a diagonalization algorithm for solving joint mode split/traffic assignment problems with non-UE transit assignment.

9.13. Consider the network discussed in Problem 9.7 (shown in Figure 9.2). Assume, however, that the O–D volume is not given but rather is determined by the function

$$q = D(t) = \frac{100}{t} - 1$$

where t is the automobile travel time. How would you find the equilibrium flows in such a network? Apply the diagonalization algorithm and show the intermediate results.

9.14. Formulate the joint mode split/trip distribution/traffic assignment problem in the case where the transit travel time is a function of the automobile traffic (as well as the transit flow), assuming that the modal interaction is asymmetric. Use the following trip distribution model:

$$\bar{q}_{rs} = \frac{O_r e^{-\gamma(u_{rs} + \beta \hat{u}_{rs} - M'_{rs})}}{\sum_m e^{-\gamma(u_{rs} + \beta \hat{u}_{rs} - M'_{rs})}}$$

(compare this model to the one given in Eq. [9.22]). Note that in this case both $\{u_{rs}\}$ and $\{\hat{u}_{rs}\}$ are a function of $\mathbf{x}$ [i.e., $u_{rs} = u_{rs}(\mathbf{x})$ and $\hat{u}_{rs} = \hat{u}_{rs}(\mathbf{x}, \hat{\mathbf{x}})$, $\forall\ r, s$].

9.15. **(a)** Formulate the mathematical program which has to be solved at the nth iteration, if the diagonalization approach is applied to the problem of joint modal split, trip distribution, and traffic assignment with variable demand. Use logit modal split and trip distribution formulas and a general term for the trip generation model, as done in Section 9.2.

(b) Describe the application of the convex combinations algorithm to the solution of this program.

(c) Describe the application of a multistage approach to the calculation of the descent direction for the problem in part (a).

9.16. The traditional "four steps" transportation planning process includes an independent analysis of trip generation, trip distribution, mode split, and traffic assignment. Each step includes its own methodology and its output is used to conduct the other analyses in a sequential manner. Compare this planning process to the supernetwork approach described in Section 9.2. Use examples of planning problems in your discussion.

IV

Stochastic User Equilibrium

The behavioral mechanism underlying the traffic assignment model is a choice, or decision-making process. Motorists traveling between origins and destinations choose the path on which to travel.

The deterministic approach to traffic assignment assumes that travelers choose the least cost, shortest, or minimum-travel-time path from their origin to their destination. This behavioral assumption is the core of the user-equilibrium (UE) conditions, which characterize the solution to all the traffic assignment problems described in Parts II and III. Although this assumption may not seem unreasonable, it presumes that all travelers have perfect information regarding travel time over the entire network, that they make consistently correct decisions, and that they all behave in identical fashion.

Part IV relaxes some of these presumptions by including a random component in travelers' perception of travel time. Route selection is then analyzed by applying discrete choice models to this process. These models are based on the concepts of utility maximization and random utility, and are used in many disciplines to model individuals' choices among a finite set of discrete options. The general form of these models is introduced in Chapter 10, which also applies the models to the route choice problem. Chapter 11 introduces two specific route-choice model forms and discusses algorithms for implementing these models. The stochastic nature of the travel times and the dependence of these times on the vehicular flows are combined in Chapter 12 in a stochastic equilibrium model.

10

Discrete Choice Models and Traffic Assignment

This chapter formulates the route-choice as a process of selection among alternative paths, on which the travel time is random. The formulation is based on the theory of discrete choice models, which describe individuals' choices between competing alternatives. The focus in this book is on travel alternatives such as the available paths from an origin to a destination, alternative modes of transportation, or alternative destinations which could be visited.

The relevant underlying theory and the two most commonly used discrete choice models are introduced in Section 10.1. This section concentrates on the formulation of the models and their relationship to the behavioral concept of utility maximization. Section 10.2 shows how these models can be applied to calculate the flow assigned to each path connecting an origin and a destination in a transportation network.

10.1 REVIEW OF DISCRETE CHOICE MODELS

The hypothesis underlying discrete choice models is that when faced with a choice situation, an individual's preferences toward each alternative can be described by an "attractiveness" or "utility" measure associated with each alternative. This utility is a function of the attributes of the alternatives as well as the decision maker's characteristics. The decision maker is assumed to choose the alternative that yields the highest utility. Utilities, however, cannot be observed or measured directly. Furthermore, many of the attributes that influence individuals' utilities cannot be observed and must therefore be treated as random. Consequently, the utilities themselves are modeled as random, meaning that choice models can give only the *probability* with which alternatives are chosen, not the choice itself.

This section presents some of the basic concepts associated with discrete choice models (which are also known as random utility models). It deals with the notions of a choice function, a satisfaction function, and the application of discrete choice models to populations with heterogeneous characteristics. These issues are discussed in general as well as in relation to two specific discrete choice model forms: the multinomial logit and the multinomial probit.

Choice Function

Let $\mathbf{U} = (U_1, \ldots, U_K)$ denote the vector of utilities associated with a given set of alternatives, $\mathscr{K}$. This set includes K alternatives numbered $1, 2, \ldots, K$. The utility of each alternative to a specific decision maker can be expressed as a function of the observed attributes of the alternatives and the observed characteristics of this decision maker. Let $\mathbf{a}$ denote the vector of variables which include these characteristics and attributes. Thus $U_k = U_k(\mathbf{a})$. To incorporate the effects of unobserved attributes and characteristics, the utility of each alternative is expressed as a random variable consisting of a systematic (deterministic) component, $V_k(\mathbf{a})$, and an additive random "error term", $\xi_k(\mathbf{a})$, that is,

$$U_k(\mathbf{a}) = V_k(\mathbf{a}) + \xi_k(\mathbf{a}) \qquad \forall\ k \in \mathscr{K} \qquad [10.1]$$

The random component of the utility satisfies $E[\xi_k(\mathbf{a})] = 0$, meaning that $E[U_k(\mathbf{a})] = V_k(\mathbf{a})$. In this context, $U_k(\mathbf{a})$ is sometimes referred to as the "perceived utility" and $V_k(\mathbf{a})$ as the "measured utility."†

The utility functions usually include a set of parameters that are statistically estimated from individuals' observed choices, as explained in Chapter 13. For now, assume that the values of these parameters are known and thus U_k changes only with $\mathbf{a}$.

Given the distribution of possible utility values, the probability that a particular alternative will be chosen by a decision maker, who is randomly selected from a given population, can be calculated. The distribution of the utilities is a function of the attribute vector, $\mathbf{a}$. Therefore, the probability that alternative k (where $k \in \mathscr{K}$) will be chosen, P_k, can be related to $\mathbf{a}$. The function relating P_k to $\mathbf{a}$ is known as the choice function and is denoted by $P_k(\mathbf{a})$. The probability that alternative k is chosen, given $\mathbf{a}$, should be understood as the fraction of individuals in a large population, all members of which are characterized by $\mathbf{a}$, who choose alternative k. The choice probability is

†U_k can be viewed as the utility of the kth alternative as perceived by the decision maker while V_k can be viewed as the utility of this alternative as measured by the analyst.

then the probability that $U_k(\mathbf{a})$ is higher than the utility of any other alternative (given $\mathbf{a}$) and

$$P_k(\mathbf{a}) = \Pr\,[U_k(\mathbf{a}) \geq U_l(\mathbf{a}), \forall\, l \in \mathscr{K}] \qquad \forall\, k \in \mathscr{K} \qquad [10.2]$$

The choice function, $P_k(\mathbf{a})$, has all the properties of an element of a probability mass function, that is,

$$0 \leq P_k(\mathbf{a}) \leq 1 \qquad \forall\, k \in \mathscr{K} \qquad [10.3a]$$

$$\sum_{k=1}^{K} P_k(\mathbf{a}) = 1 \qquad [10.3b]$$

Once the distribution of the error term, ξ_k, is specified, the distribution of the utilities can be determined, and the choice function can be calculated explicitly. Consider, for example, a choice situation in which only two alternatives are available. The utility of the first alternative is $U_1 = 3 + \xi$ and the utility of the second alternative is $U_2 = 2$. In this case all parameters and characteristics are known and fixed. The probability that the first alternative will be chosen, P_1, is

$$\begin{aligned} P_1 = \Pr\,(U_1 \geq U_2) &= \Pr\,(3 + \xi \geq 2) \\ &= \Pr\,(\xi \geq -1) \end{aligned} \qquad [10.4]$$

Assume now that the density of ξ is uniform between -2 and 2, that is,

$$\Pr\,(\omega \leq \xi \leq \omega + d\omega) = \begin{cases} \frac{1}{4} & \text{for } -2 \leq \omega \leq 2 \\ 0 & \text{otherwise} \end{cases} \qquad [10.5]$$

where ω is any real number. Given this distribution, the choice probability expressed by Eq. [10.4] is $P_1 = 0.75$, implying that $P_2 = 0.25$. This example demonstrates the nature of the choice probability; 75% of all individuals for whom the utility functions are specified as above will choose alternative 1, according to this model. If, for some other population of individuals, U_1 remains unchanged and $U_3 = 3$ (instead of 2), the choice probability of the first alternative, P_1, will be 0.5 (assuming that the distribution of ξ does not change). In other words, half of the new population will choose alternative 1. The choice probability thus depends on the characteristics (captured by the vector $\mathbf{a}$) of the utility functions, as well as on the distribution of the random component.

The following two sections discuss two specific discrete choice models, arising from two different specifications of the error term. These are the logit and the probit models.

Multinomial Logit

One of the most widely used discrete choice models is the logit model. The logit formula was presented in Chapters 6, 7, and 9 both as a modal split and as a trip distribution model. In those contexts it was presented as an aggregate share formula giving the total number of individuals selecting a particular travel alternative. This model, however, can be derived from the concepts of random utility and utility maximization by assuming that the random terms of each utility function are independently and identically distributed Gumbel variates.† The choice probability is then given by

$$P_k = \frac{e^{V_k}}{\sum_{l=1}^{K} e^{V_l}} \qquad \forall\ k \in \mathscr{K} \qquad [10.6a]$$

where the dependence of P_k and V_k on $\mathbf{a}$ is omitted from the notation for clarity of presentation. The choice probability of a given alternative can be expressed as a function of the difference between the measured utilities of that and all other alternatives. To see this, divide the right-hand side of Eq. [10.6a] by e^{V_k} to obtain

$$P_k = \frac{1}{1 + \sum_{l \neq k} e^{V_l - V_k}} \qquad [10.6b]$$

In the case of binary logit, that is, when there are only two alternatives (indexed $k = 1$ and $k = 2$), the model can be written as

$$P_1 = \frac{e^{V_1}}{e^{V_1} + e^{V_2}} = \frac{1}{1 + e^{V_2 - V_1}} \qquad [10.7a]$$

and, of course,

$$P_2 = 1 - P_1 = \frac{1}{1 + e^{V_1 - V_2}} \qquad [10.7b]$$

Consider, for example, a bus/automobile logit mode choice model, where the utilities of the two alternative modes have been specified as follows:

$$U_{\text{bus}} = V_{\text{bus}} + \xi \qquad [10.8a]$$

where $V_{\text{bus}}(\hat{t}) = -2\hat{t}$, and

$$U_{\text{auto}} = V_{\text{auto}} + \xi \qquad [10.8b]$$

where $V_{\text{auto}}(t) = 1 - 2t$. The variables of interest in this example are $\hat{t}$ and t, the transit and automobile travel time, respectively. The probability of choosing the automobile mode (see Eq. [10.7a]),

$$P_{\text{auto}}(t, \hat{t}) = \frac{1}{1 + e^{V_{\text{bus}} - V_{\text{auto}}}} = \frac{1}{1 + e^{-1-2(\hat{t}-t)}} \qquad [10.9]$$

†The cumulative Gumbel distribution function is given by

$$F(\omega) = \Pr\ (\xi_k \leq \omega) = e^{-e^{-\omega+E}}$$

where ω is any real number and E is Euler's constant (i.e., $E = 0.5708\ldots$).

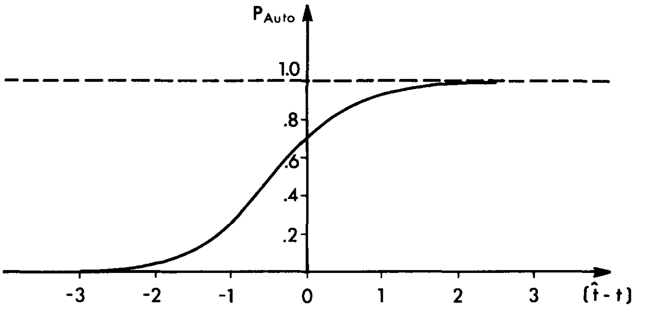


Figure 10.1 Logit-based automobile choice function (see Eq. [10.9]).

Figure 10.1 illustrates the automobile choice function, $P_{\text{auto}}(t, \hat{t})$, versus the difference $(\hat{t} - t)$. Given t and $\hat{t}$ (or merely their difference), the model gives the share of all individuals in the population under study, choosing the automobile alternative. If the size of this population is known, the number of individuals choosing each mode can be determined.

The logit formula is easy to use in the multinomial case (see Eq. [10.6a]) as well. This, in fact, is the main advantage of the logit as compared with other, more sophisticated discrete choice models.

Multinomial Probit

Following many other statistical models, it may be natural to assume that the random error term of each utility is normally distributed. This is the underlying assumption of the probit model which is based on the postulate that the joint density function of these error terms is the multivariate normal (MVN) function.

The MVN distribution is the multinomial extension of the well-known normal density function. It describes the distribution of a random vector, $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_K)$. This distribution is characterized by a (K-length) vector of means, $\boldsymbol{\mu}$, and a ($K \times K$) covariance matrix, $\boldsymbol{\Sigma}$. The notation $\boldsymbol{\xi} \sim \text{MVN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ indicates that the vector $\boldsymbol{\xi}$ is MVN distributed with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The covariance matrix includes the variances of the components of the random vector, and the covariances between these components, that is,

$$(\boldsymbol{\Sigma})_{kk} = \text{var}(\xi_k) \quad \forall\, k \quad \text{and} \quad (\boldsymbol{\Sigma})_{kl} = \text{cov}(\xi_k, \xi_l) \quad \forall\, k \neq l \qquad [10.10]$$

The appendix describes some of the properties of MVN distributions which are important for probit analysis. As shown there, this distribution conserves its shape under linear transformation, meaning, for example, that the sum (as well as the difference) of two normal variates is normally distributed. Given a covariance matrix (associated with the randomness in the perception of a set of alternatives) and a vector of known alternatives' attributes, the distribution of the utility vector, $\mathbf{U}(\mathbf{a}) = [U_1(\mathbf{a}), \ldots, U_K(\mathbf{a})]$, can be modeled as multivariate normal; in other words,†

$$\mathbf{U}(\mathbf{a}) \sim \text{MVN}\,[\mathbf{V}(\mathbf{a}), \boldsymbol{\Sigma}] \qquad [10.11]$$

As with any discrete choice model, the probability that a particular alternative will be chosen is given by the probability that its utility is the highest in the choice set (see Eq. [10.2]). With probit models, however, this choice probability cannot be expressed analytically since the cumulative normal distribution function cannot be evaluated in closed form. In the case of a two-alternative (i.e., binary) model, however, the choice probabilities can be computed by referring to cumulative normal tables, as shown below.

To demonstrate the computation of a choice probability in the binary case, assume that $\mathbf{U} = (U_1, U_2)$ where $U_1 = V_1 + \xi_1$ and $U_2 = V_2 + \xi_2$. The distribution of the error term vector is given by

$$\boldsymbol{\xi} \sim \text{MVN}\,(\mathbf{0}, \boldsymbol{\Sigma})$$

where $\boldsymbol{\xi} = (\xi_1, \xi_2)$, $\mathbf{0}$ is a vector of zeros, (0, 0), and

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{pmatrix}$$

Now (from Eq. [10.2]), the probability of choosing the first alternative is given by

$$\begin{aligned} P_1 &= \Pr\,[(U_1 \geq U_2) = \Pr\,[(V_1 + \xi_1) \geq (V_2 + \xi_2)] \\ &= \Pr\,[(\xi_2 - \xi_1) \leq (V_1 - V_2)] \qquad [10.12a] \end{aligned}$$

Due to the shape-conserving characteristics of the MVN distribution, $(\xi_2 - \xi_1)$ is a normally distributed random variable, with mean zero and variance $\sigma^2 = \sigma_1^2 + \sigma_2^2 - 2\sigma_{12}$. Using the standard normal transformation, Eq. [10.12a] implies that

$$P_1 = \Phi\left(\frac{V_2 - V_1}{\sqrt{\sigma^2}}\right) \qquad [10.12b]$$

where $\Phi(\cdot)$ is the standard cumulative normal curve. Once the values of V_1, V_2, and σ are given, the value of P_1 can be found by referring to a standard normal distribution table.

The calculation of the probit choice probability, in the presence of more than two alternatives is not straightforward. The choice probability can be estimated by using either an analytical approximation or a Monte Carlo simulation. These two approaches are outlined below.

The literature suggests several analytical approximation methods for computing the probit choice probabilities. These include numerical integration

†In general, the elements of the covariance matrix can be a function of **a** [in which case $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(\mathbf{a})$]. This book, however, deals only with probit models in which this is not the case.

algorithms as well as a successive approximations method. The numerical integration methods try to solve directly for the choice probability. This leads to a multiple integral expressing the cumulative distribution function of the utilities. This integral is then solved numerically.

The successive approximations method is based on the approximation of the distribution of the *maximum* of two normally distributed random variables as another normal variable. Given a set of normally distributed utilities, this approximation can be applied iteratively. Consider, for example, the computation of the choice probability of alternative k (this equals the probability that $U_k \geq U_l$, $\forall\ l$). At each iteration, two random utilities are considered and an auxiliary random variable, equivalent to the maximum of these two, is created. The next iteration consists of calculating a new auxiliary variable which is the maximum of the former auxiliary variable and another utility variable. The new auxiliary variable replaces the old one and the process continues. U_k is considered last. At that point the auxiliary variable approximates the distribution of the maximum of all the alternatives' utilities (excluding k). The choice probability is, then,

$$P_k = \Pr\left(U_k \geq \max_{\forall\, l \neq k} \{U_l\}\right) \qquad [10.13]$$

which is equivalent to the probability that U_k is greater than or equal to all U_l (for $l \neq k$). Equation [10.13] includes only two random variables (U_k and $\max_{\forall\, l \neq k} \{U_l\}$) and P_k can, therefore, be computed as before (see Eq. [10.12b]). This approximation (known as Clark's method) is described in detail in the Appendix.

None of the analytical approximation methods can be practically applied to problems involving a large number of alternative choices. The numerical integration method cannot be applied to problems involving more than four or five alternatives due to prohibitive computational costs. The successive approximations can be applied to somewhat larger problems but, again, it would involve prohibitive computational costs for problems of the size encountered in network analysis. Furthermore, the accuracy of this method deteriorates as the number of alternatives increases.

Another approach to the computation of the probit choice probabilities is based on a Monte Carlo simulation procedure. The simulation method can be applied to the computation of the choice function of any discrete choice model as shown here. To describe the technique, consider a set of utility functions: $U_k = V_k + \xi_k$, $\forall\ k \in \mathscr{K}$. Given the values of $\mathbf{V} = (V_1, \ldots, V_K)$, the simulation works iteratively as follows: A vector comprising K random variables is drawn at every iteration from the density function of $\boldsymbol{\xi}$. Denote the drawings generated in the nth iteration by $\xi_1^n, \ldots, \xi_K^n$. The perceived utility of each alternative can now be computed by adding the systematic utility, V_k, to the (drawn) random term ξ_k^n, that is, $U_k^n = V_k + \xi_k^n$, $\forall\ k$. Next, the maximum utility alternative (i.e., the alternative for which $U_k^n \geq U_l^n$, $\forall\ l$) is recorded. This process is repeated N times. Denoting the number of times that each alternative k was recorded (as the maximum utility one) by N_k, the choice probability of the kth alternative, P_k, is given by

$$P_k \simeq \frac{N_k}{N} \qquad [10.14]$$

where, at the limit (as $N \to \infty$), $P_k = N_k/N$. This technique is particularly useful for network analysis, as shown in Chapter 11.

The Satisfaction Function

The focus of the first part of this section was on the choice function and the computation of the choice probability. Another important quantity related to discrete choice models is the *satisfaction*, $\tilde{S}$, and the *satisfaction function*, $\tilde{S}(\mathbf{a})$. The satisfaction is a scalar, which captures the expected utility that an individual (decision-maker) derives from a *set* of alternatives, $\mathscr{K}$, each with utility U_k. Since each individual is assumed to select the alternative with the maximum utility, the satisfaction is simply the expectation of the maximum utility alternative. In other words,

$$\tilde{S} = E\left[\max_{\forall k} \{U_k\}\right] \qquad [10.15a]$$

where the expectation is taken with respect to the distribution of $\mathbf{U}$ (which is, of course, derived from the distribution of $\boldsymbol{\xi}$). The satisfaction can be expressed as a function of the vector $\mathbf{a}$ and the quantity $\tilde{S}(\mathbf{a})$ is known as the satisfaction function. Given that $\mathbf{V} = \mathbf{V}(\mathbf{a})$ and given the distribution of $\boldsymbol{\xi}$, the satisfaction can also be expressed as a function of the vector of measured utilities, $\mathbf{V}$, that is,†

$$\tilde{S}(\mathbf{V}) = E\left[\max_{\forall k} \{V_k + \xi_k\}\right] \qquad [10.15b]$$

The satisfaction function, $\tilde{S}(\mathbf{V})$, has three important properties:

1. It is convex with respect to $\mathbf{V}$. Furthermore, for the discrete choice models discussed in this text (where the choice probability is positive for all finite values of $\mathbf{V}$), the satisfaction function is also strictly convex.
2. The partial derivative of the satisfaction function with respect to the systematic utility of an alternative, equals the choice probability of that alternative, that is,

$$\frac{\partial \tilde{S}(\mathbf{V})}{\partial V_k} = P_k(\mathbf{V}) \qquad \forall\, k \in \mathscr{K} \qquad [10.16a]$$

3. The satisfaction is monotonic with respect to the size of the choice set, that is,

$$\tilde{S}(V_1, \ldots, V_K, V_{K+1}) \geq S(V_1, \ldots, V_K) \qquad [10.16b]$$

†This, of course, holds also for the choice probability $P_k(\mathbf{V}) = P_k[\mathbf{V}(\mathbf{a})]$.

The first property of the satisfaction function stems directly from the convexity of the maximum operator (see Problem 10.4). The second property holds only under certain regularity conditions,† which are satisfied by all the models discussed in this book. It can be proved directly from the definitions of the satisfaction and the choice function (see Problem 10.5). The third property holds only for noninteracting alternatives (i.e., if the introduction of the extra alternative does not decrease the measured utility of any other alternative). Under these conditions, this property can, again, be proved on the basis of the definition of the satisfaction function and the properties of the maximization operator (see Problem 10.6).

For logit models, the form of the satisfaction function can be explicitly derived from the second property above (Eq. [10.16a]). Given that

$$\frac{\partial \tilde{S}(\mathbf{V})}{\partial V_k} = \frac{e^{V_k}}{\sum_l e^{V_l}} \qquad [10.17a]$$

The satisfaction function is

$$\tilde{S}(\mathbf{V}) = \ln \sum_l e^{V_l} \qquad [10.17b]$$

The integration constant in this case is zero, since a proper boundary condition for Eq. [10.17a] is that $\tilde{S}(\mathbf{V}) = V_k$ in case $V_l \rightarrow -\infty$, $\forall\, l \neq k$. In this case the kth alternative is the only reasonable one and the satisfaction equals the expected utility of the kth alternative. Equation [10.17b] can also be derived directly from the definition of the satisfaction (Eq. [10.15b]).

For probit models, the satisfaction cannot be expressed analytically. Using Clark's approximation, however, $\tilde{S}(\mathbf{V})$ can be calculated by simply carrying the successive approximations process one more step, to include the last utility function (U_k—see the discussion preceding Eq. [10.13], and the Appendix.) This results in an auxiliary variable which is the maximum utility of the entire set of alternatives. The mean of this variable is the satisfaction.

The satisfaction can also be calculated by using Monte Carlo simulation. To do this, the utility of the chosen alternative, U^n, has to be recorded at every iteration ($U^n = \max_{\forall l} \{U_l^n\}$). The satisfaction is then given by

$$\tilde{S} \simeq \frac{1}{N} \sum_{n=1}^{N} U^n \qquad [10.18]$$

where N is the number of iterations (drawings).

Aggregate Predictions

The choice function relates the choice probability to a set of variables, **a**. Some of the variables in the vector **a** represent characteristics (such as income) which vary from one individual to another. Thus **a** is distributed across a

†These regularity conditions require that the shape of the utilities density function not depend on the measured utilities. Note that probit models in which $\Sigma \neq \Sigma(\mathbf{a})$ do meet these regularity conditions.

given population of decision makers according to some density function, $f(\mathbf{a})$. Given the choice function of alternative k, $P_k(\mathbf{a})$, the share of the population under study who choose alternative k is given by

$$\bar{P}_k = \int_{\mathbf{a}} P_k(\mathbf{a}) f(\mathbf{a}) \, d\mathbf{a} \qquad [10.19]$$

This is a multiple integral that may be difficult to evaluate. It is the expectation of $P_k(\mathbf{a})$ with respect to the (multivariate) distribution of $\mathbf{a}$ (i.e., $E_{\mathbf{a}}[P_k(\mathbf{a})]$).

This integral can be approximated by dividing the population into groups of similar values of $\mathbf{a}$, computing $P_k(\mathbf{a})$ for each group, and averaging the results. Alternatively, $P_k(\mathbf{a})$ can be computed by Monte Carlo simulation. Such an approach includes drawing random values of $\mathbf{a}$ from $f(\mathbf{a})$, computing $P_k(\mathbf{a})$ for each one, and averaging the results. The issue of aggregate predictions is of secondary importance in network analysis. In this type of analysis the choice models do not typically include attributes that vary across individuals, and thus the aggregation problem does not arise. This is the reason that the modal split and trip distribution models presented in previous chapters can be considered as mode choice and destination choice models, respectively. Had these models included variables that vary across individuals, the analysis given in those chapters would have required that these models be appropriately aggregated. Such an aggregation would have added a significant computational burden to the equilibrium algorithms.

10.2 STOCHASTIC NETWORK LOADING

The UE model discussed in Parts II and III assumes that each individual follows the path with the shortest travel-time from origin to destination. Thus the network loading mechanism underlying the UE assignment can be modeled by an all-or-nothing assignment process, in which all the flow between each origin and destination is assigned to the shortest-travel-time path connecting this pair. The UE model results from the coupling of this assumption with the dependence between travel time and flow.

This section uses the notions of choice models reviewed in Section 10.1 to formulate a network loading mechanism that is more general than the all-or-nothing assignment. This model views the route choice in the general framework of random utility theory and discrete choice models. The focus of this section is on the network loading mechanism, which, in this case, is referred to as *stochastic* network loading. The dependence of link travel time on link flows (which underlies the equilibrium issue) is not treated here in order to concentrate on network loading when the path times are random. Equilibrium effects in conjunction with stochastic network loading are examined in Chapter 12.

This section includes four topics, each discussed separately. The first part of this section derives the stochastic network loading model from the process of route choice. The second part introduces a stochastic network paradox, which parallels, to some extent, Braess's paradox discussed in Chapter 5. The nature of the stochastic assignment paradox is explained in the third part, through the notion of the expected minimum travel time. The last part of this section examines the nature of the network flow in more detail under the assumptions of a stochastic network loading model.

Route Choice

Consider a population of drivers who are about to take a trip between a given origin and a given destination. The O–D pair under consideration is connected by many alternative paths, each associated with some travel time. Due to variations in perception and exogenous factors (such as weather, lighting, etc.) the path times are perceived differently by each driver. It is natural, then, to model the *perceived* travel time of each path as a random variable distributed across the population of drivers. Given his or her perception of travel time, each driver is assumed to choose the shortest travel time path from origin to destination. Each driver, however, will perceive path times differently, and therefore each driver may choose a different path. Since the perceived travel time of each path is a random variable, it is associated with some probability density function. This function gives the probability that a driver randomly drawn from the population will associate a given travel time with that path. The stochastic network loading problem is to determine, based on this distribution, how many travelers will use each path.† The link flows can be obtained from the paths' flow by the incidence relationships (Eq. [3.1b]).

To treat this problem analytically, let C_k^{rs} represent the perceived travel time on route k between origin r and destination s, where $k \in \mathscr{K}_{rs}$. C_k^{rs} is a random variable. Also, let c_k^{rs} be the measured, or actual travel time on route k between r and s. Assume now that

$$C_k^{rs} = c_k^{rs} + \xi_k^{rs} \qquad \forall\ k, r, s \tag{10.20}$$

where ξ_k^{rs} is a random error term associated with the route under consideration. Furthermore, assume that $E[\xi_k^{rs}] = 0$, or $E[C_k^{rs}] = c_k^{rs}$. In other words, the average perceived travel time is the actual travel time. (The average travel time can be interpreted as the travel time used in the deterministic analysis of Parts II and III.) If the population of drivers between r and s is large, the share of drivers choosing the kth route, P_k^{rs}, is given by

$$P_k^{rs} = \Pr\left(C_k^{rs} \le C_l^{rs}, \forall\ l \in \mathscr{K}_{rs}\right) \quad \forall\ k, r, s \tag{10.21}$$

In other words, the probability that a given route is chosen is the probability that its travel time is perceived to be the lowest of all the alternative routes.

†A stochastic network loading model can also be derived from the assumption that link travel times are inherently random. Each driver is then assumed to select paths on the basis of experience that may vary from driver to driver due, for example, to small differences in trip timing.

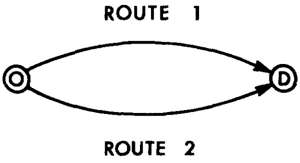


Figure 10.2 Network example with one O–D pair and two routes.

Equation [10.21] specifies a route-choice probability that can be interpreted in the framework of discrete choice models. The population of interest equals the trip rate from origin r to destination s. Each path, k, is an alternative associated with some utility function U_k^{rs}. The utility of each alternative (path) equals the negative path cost (i.e., $U_k^{rs} = -C_k^{rs}$) since it is assumed here that the travel time is the only determinant of route choice.† The utility maximization principle then implies Eq. [10.21]. The alternative with the highest (perceived) utility is the alternative with the lowest perceived travel time.

The various models for stochastic network loading differ from each other in the assumed distribution of the perceived travel times. Once this distribution is specified, the probability of selecting each alternative route can be calculated and the flow assigned accordingly. The path flow assignment then will be

$$f_k^{rs} = q_{rs} P_k^{rs} \qquad \forall\ k, r, s \tag{10.22a}$$

where P_k^{rs} is the route-choice probability given by Eq. [10.21]. The link flows can then be calculated as

$$x_a = \sum_{rs} \sum_k f_k^{rs} \delta_{a,k}^{rs} \qquad \forall\ a \tag{10.22b}$$

One of the practical reasons for using stochastic network loading models is the sensitivity of the flows in deterministic models to small changes in the network. Consider, for example, the small network in Figure 10.2. This network includes two paths, the travel times on which are t_1 and t_2, connecting a single O–D pair. According to a deterministic network loading model, all the flow from the origin to the destination (q) should be assigned to the lowest-travel-time path. If, for example, $c_1 = c_2 + \Delta t$, then $f_1 = 0$ and $f_2 = q$ even if $\Delta t \rightarrow 0$ (i.e., if the travel times are very close to each other). Each path in this network, however, may include various transportation facilities (such as bridges, intersections, etc.). A small error in the data measurement for any one of these facilities may cause a situation in which $c_1 < c_2$ by a small amount. Such a change, however, would mean that the calculated flow pattern over the network will be completely different with $f_1 = q$ and $f_2 = 0$.

Stochastic network loading models usually avoid such sensitivities. The

†This framework can be generalized, as in the deterministic case, by replacing the travel time with a measure of general travel impedance or disutility. Note also that a unit scaling factor between travel time units and utility "units" can be introduced without affecting the problem. In other words, U_k^{rs} can be defined to equal $-\theta C_k^{rs}$ where θ is a (positive) unit scaling parameter.

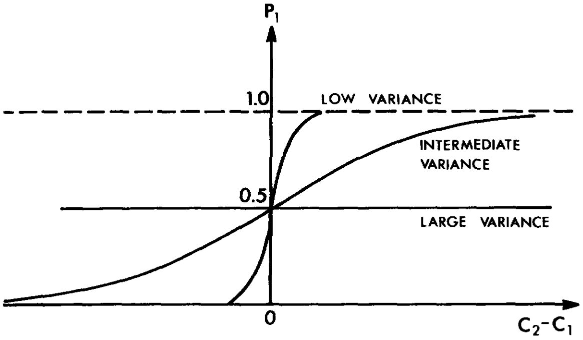


Figure 10.3 Effect of the variance of the paths' perceived travel-time distribution on the flow pattern for the network example depicted in Figure 10.2.

probability that route 1 is chosen will be approximately 0.50, if $c_1 \simeq c_2$. A small change in the network representation may mean that the share of a given route may change from just under 50% to just over 50% of the total O–D flow. Thus a small change in the travel times results in a small change in the flow pattern. The exact magnitude of the change in flow will depend on the variance of the distribution of the path travel times. Figure 10.3 demonstrates the effect of the travel time distribution variance on the flow pattern for the network in Figure 10.2. It describes the choice probability of route 1, P_1, as a function of the difference among the routes' measured travel times, $c_2 - c_1$, for various values of the travel-time variance.

All the curves shown in the figure depict an increase in P_1 as $(c_2 - c_1)$ increases (i.e., as c_1 becomes smaller, relative to c_2). For very low variance, the curve is very steep, meaning that travelers perceive the differences between c_1 and c_2 accurately and act accordingly. In fact, if the variance is zero, $P_1 = 0$ for $(c_2 - c_1) < 0$ and $P_1 = 1$ for $(c_2 - c_1) > 0$. This is the route-choice behavior associated with the deterministic model in which the drivers' perception of the travel times is assumed to be perfectly accurate. In a deterministic (all-or-nothing) model, all users will be assigned to the shortest (measured) travel-time path from their origin to their destination.

If the variance is larger, the change in the choice probability is more moderate since the perception of travel times is less accurate. At the limit, when the variance is extremely large, the actual travel times do not affect the perception, which is completely random. In this case $P_1 = 0.5$ for any value of $(c_2 - c_1)$, as shown in the figure. The actual magnitude of the perception variance should be determined empirically on a case-by-case basis. There is no reason, however, to think a priori that it will be zero, as implied by the deterministic approach.†

†As it turns out, this variance tends to be relatively small in practice. Furthermore, under some conditions, deterministic equilibrium models provide a good approximation for stochastic ones, as shown in Chapter 12.

Another Paradox of Traffic Flow

Section 5.5 showed that under certain conditions, network improvement (such as link additions) can result in an overall increase in transportation costs (or travel times). This phenomenon, known as Braess's paradox, can readily be explained on the basis of the difference between the UE flow pattern and the system-optimizing flow pattern. Thus, although counterintuitive, this phenomenon is not quite a paradox.

Braess's paradox occurs only in networks where travel times are flow dependent. If this is not the case, the UE flow pattern will be identical to the system-optimizing flow pattern and this (pseudo) paradox will not take place. This section discusses a similar counterintuitive phenomenon which arises in the context of stochastic network loading models. This phenomenon is not rooted in the dependence of the links' travel time on the flow but in the stochastic nature of the loading model, that is, in the randomness of the perceived path travel times. If the variance of the travel time is zero, the stochastic network loading is equivalent to a deterministic one, and the paradox discussed here will not arise.

The paradox under consideration can be demonstrated by using the simple two-path network shown in Figure 10.2. Again, assume that the measured travel times on route 1 and route 2 are c_1 and c_2, respectively, and similarly, the perceived travel times on these routes are C_1 and C_2. The O–D trip rate is q units of flow. If the mean path travel times, $\mathbf{c} = (c_1, c_2)$, are fixed, the flow on route 1, f_1, is given by

$$f_1 = qP_1(\mathbf{c}) \qquad [10.23a]$$

where $P_1(\mathbf{c}) = \Pr(C_1 \le C_2)$, is the probability that route 1 will be chosen by a motorist who is randomly drawn from the population of q drivers. The flow on route 2, f_2, is given by

$$f_2 = qP_2(\mathbf{c}) = q[1 - P_1(\mathbf{c})] \qquad [10.23b]$$

The total travel time in the system, c_T, is

$$\begin{aligned} c_T(\mathbf{c}) &= c_1 f_1 + c_2 f_2 \\ &= q[(c_1 P_1(\mathbf{c}) + c_2 P_2(\mathbf{c})] \qquad [10.24] \end{aligned}$$

To illustrate the paradox numerically, assume that $c_1 = 4.0$, $c_2 = 2.0$, and $q = 1.0$ in some appropriate units. Assume further that the choice model is given by the logit-like function

$$P_1 = \frac{1}{1 + e^{c_1 - c_2}}$$

The flows in this case are $f_1 = 0.119$ and $f_2 = 0.881$. The total travel time is

$$c_T = 0.119 \times 4.0 + 0.881 \times 2.0 = 2.238 \text{ (flow} \cdot \text{time) units}$$

Consider now an improvement of route 1 which reduces the travel time on this route from 4.0 time units to 3.0 time units. The flows in this case would be $f_1 = 0.269$ and $f_2 = 0.731$, with the total travel time being

$$c_T = 0.269 \times 3.0 + 0.731 \times 2.0 = 2.269 \text{ (flow} \cdot \text{time) units}$$

The total travel time, c_T, increased in this case even though one of the links was improved (!).

The numbers used in this example can be generalized for any route-choice model (see Eqs. [10.23]) and any values of c_1, c_2, and q as follows. Given Eq. [10.24], consider the effect of a small improvement in the actual travel time of route 1, c_1, on the total travel time in the system, $c_T(\mathbf{c})$. This marginal effect can be expressed as the partial derivative of $c_T(\mathbf{c})$ with respect to c_1. The derivative is

$$\frac{\partial c_T(\mathbf{c})}{\partial c_1} = q\left[c_1 \frac{\partial P_1(\mathbf{c})}{\partial c_1} + P_1(\mathbf{c}) + c_2 \frac{\partial P_2(\mathbf{c})}{\partial c_1}\right] \qquad [10.25a]$$

This equation can be simplified since $P_2 = 1 - P_1$, meaning that $\partial P_2(\mathbf{c})/\partial c_1 = - \partial P_1(\mathbf{c})/\partial c_1$. Equation [10.25a] can therefore be written as

$$\frac{\partial c_T(\mathbf{c})}{\partial c_1} = q\left[(c_1 - c_2) \frac{\partial P_1(\mathbf{c})}{\partial c_1} + P_1(\mathbf{c})\right] \qquad [10.25b]$$

If network improvements (such as a *reduction* in c_1) result in a *decrease* in the total travel time, $\partial c_T(\mathbf{c})/\partial c_1$ should be positive. For this derivative to be positive, the following condition should hold:

$$\frac{\partial P_1(\mathbf{c})}{\partial c_1}(c_1 - c_2) > -P_1(\mathbf{c})$$

or

$$(c_1 - c_2) < -\frac{P_1(\mathbf{c})}{\partial P_1(\mathbf{c})/\partial c_1} \qquad [10.26a]$$

The inequality sign is reversed in the last expression since $\partial P_1(\mathbf{c})/\partial c_1$ is a negative quantity; the probability of choosing a particular alternative decreases as its travel time increases. Consequently, the right-hand side of Eq. [10.26a] is a *positive* constant for any value of $\mathbf{c}$. Letting $K(\mathbf{c})$ denote this constant, Eq. [10.26a] can be rewritten as

$$c_1 - c_2 < K(\mathbf{c}) \qquad [10.26b]$$

Only if this condition is met does the total travel time decrease as c_1 is slightly decreased. If $c_1 > c_2 + K(\mathbf{c})$, a small improvement in c_1 will result in an *increase* in the total travel time, as happened in the numerical example above. Again, a counterintuitive phenomenon.

This "paradox" occurs when an inferior travel alternative is marginally improved. In the example under consideration, the paradox will occur when c_1 is reduced only if it was considerably larger than c_2 to begin with (in which case Eq. [10.26b] would not hold). Note that this was the case with the numerical example above. Because of the nonlinearity of the route choice model, "too many" travelers shift to the improved facility, even though it is still an inferior alternative, and consequently the total travel time in the system increases. Obviously, this paradox will not arise when an alternative that is better to start with is improved, since this will only cause more travelers to use the faster route and the total travel time will decrease. This can also be seen from Eq. [10.25b], which implies that when $c_1 \leq c_2$, $\partial c_T(\mathbf{c})/\partial c_1$ is always nonnegative (since $\partial P_1(\mathbf{c})/\partial c_1 < 0$).

This example illustrates how the total travel time in the system may increase when a component of the network is improved. As in Braess's paradox, the total travel time may also increase when a new link is added to the network. This case is illustrated in Figure 10.4, which depicts a network of four links, each with actual (measured) travel time of 1 unit (see Fig 10.4a). These four links form two paths between the origin and the destination. The measured travel time on these paths is $c_1 = c_2 = 2$ time units. The addition of the center link (with measured travel time of 0.1 units) in Figure 10.4b provides one additional (longer) path from the origin to the destination thereby seemingly improving the network. (The range of choices available to travelers between O and D is larger than before.) Due to the stochastic nature of the problem, however, the model is bound to assign some motorists to the longer path and the total system travel time will therefore increase. For example, in Figure 10.4b, the new average travel time will be between 2.0 and 2.1 units of time per traveler, as compared with 2.0 units before the introduction of the new link. The total travel time will increase accordingly.

In both examples considered here (the one depicted in Figure 10.2 and the one depicted in Figure 10.4), the total travel time in the network increased when the network was seemingly improved. If the users of the system were interviewed, however, they would have all been reporting an *improvement* in the travel time (!). Thus the travel time as travelers perceive it did decrease even though the total measured time increased as a result of the improvement. The situation is reminiscent of Braess's paradox in which, even though the system-optimizing objective function increased when a link was added, the UE

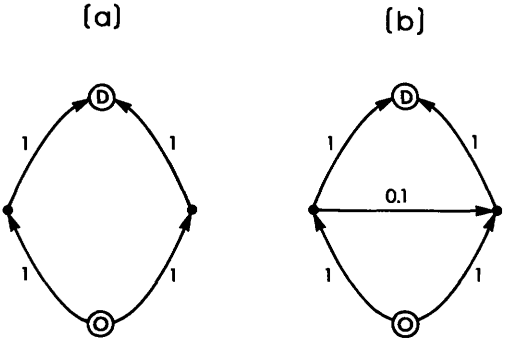


Figure 10.4 Stochastic network loading paradox: (a) a network with two equal travel-time paths connecting a single O–D pair; (b) a third path with higher travel time is added, causing the total network time to increase. (The numbers on the links indicate expected link travel times.)

objective function had decreased. In the problem discussed here there is also "something that decreases" even though the total measured travel time increases. This "something" is the expected total perceived travel time, which parallels the satisfaction concept mentioned in the preceding section. The following section presents this quantity, which plays a major role in stochastic assignment models.

The Expected Perceived Travel Time

Route choice models can be viewed as discrete choice models in which the utility of traveling over a given path, U_k^{rs}, is given by

$$U_k^{rs} = -\theta C_k^{rs} \qquad [10.27]$$

where θ is a (positive) unit scaling parameter. (See the discussion following Eq. [10.21] and the related footnote.) The satisfaction function for a (utility maximization) model of choice between the alternative paths connecting O–D pair r–s, $\tilde{S}_{rs}(\mathbf{c}^{rs})$, is given by (see Eq. [10.15])

$$\tilde{S}_{rs} = E\left[\max_{k \in \mathscr{K}_{rs}} \{U_k^{rs}\}\right] \qquad [10.28]$$

The emphasis in route-choice models, however, is on cost, impedance, or travel-time minimization rather than utility maximization. Let, then, $S_{rs}(\mathbf{c}^{rs})$ denote the expected perceived travel time between origin r and destination s. In other words,

$$S_{rs}(\mathbf{c}^{rs}) = E\left[\min_{k \in \mathscr{K}_{rs}} \{C_k^{rs}\}\right] \qquad [10.29]$$

The relationship between $S_{rs}(\mathbf{c}^{rs})$ and the satisfaction function, $\tilde{S}_{rs}(\cdot)$, can be derived by substituting $C_k^{rs} = (-1/\theta)U_k^{rs}$ (see Eq. [10.27]) in Eq. [10.29], that is,

$$\begin{aligned} S_{rs}(\mathbf{c}^{rs}) &= E\left[\min_k \left\{-\frac{1}{\theta}\, U_k^{rs}\right\}\right] \\ &= -\frac{1}{\theta}\, E\left[\max_k \{U_k^{rs}\}\right] = -\frac{1}{\theta}\, \tilde{S}_{rs}(\mathbf{c}^{rs}) \end{aligned} \qquad [10.30]$$

The expected perceived travel time function, then, is minus the satisfaction function, scaled in travel time units. It captures the expected disutility of travel between O–D pair r–s (measured in travel-time units) as perceived by a randomly selected motorist. (This motorist is randomly selected from the q_{rs} motorists going from r to s.) The total expected perceived travel time between O–D pair r–s is $q_{rs} S_{rs}(\mathbf{c}^{rs})$.

The properties of the expected perceived travel time are similar to those of the satisfaction except for its orientation toward travel-time minimization rather than utility maximization. Thus $S_{rs}(\mathbf{c}^{rs})$ is *concave* with respect to $\mathbf{c}^{rs}$. Furthermore, as with the satisfaction function, the marginal expected perceived travel time of O–D pair r–s, with respect to c_k^{rs}, is the probability of choosing that route, that is,

$$\frac{\partial S_{rs}(\mathbf{c}^{rs})}{\partial c_k^{rs}} = P_k^{rs}(\mathbf{c}^{rs}) \qquad [10.31a]$$

where $P_k^{rs}(\mathbf{c}^{rs})$ is the probability that the travel time on route k between r and s is *smaller* than the travel time on any other route between r and s. Finally, $S_{rs}(\mathbf{c}^{rs})$ is monotonic with respect to the number of available paths between r and s, that is,

$$S_{rs}(\mathbf{c}^{rs}, c_l^{rs}) \leq S_{rs}(\mathbf{c}^{rs}) \qquad [10.31b]$$

where the vector of paths times, $\mathbf{c}^{rs}$, does not include the measured travel time on path l, c_l^{rs}. Equations [10.31] follow directly from the second and third properties of the satisfaction function stated in Eqs. [10.16].

The paradoxes presented in the preceding section can now be readily explained by considering the expected perceived travel-time concept. The expected perceived travel time will always decrease (or at least will never increase) when the network is improved since motorists, in the stochastic loading model, are assumed to minimize their *perceived* travel time. To see this note that the derivative of $S_{rs}(\mathbf{c}^{rs})$ with respect to the measured travel time on any given path between r and s is always nonnegative (see Eq. [10.31a]). This implies that whenever the travel time on any path connecting O–D pair r–s decreases, the expected perceived travel time between r and s, $S_{rs}(\mathbf{c}^{rs})$, will decrease. The other property of the expected perceived travel time is that it is monotonically decreasing (or rather nonincreasing) with respect to the number of alternatives. Consequently, if a new path is added to the network, $S_{rs}(\mathbf{c}^{rs})$ will never increase. As this discussion demonstrates, the expected perceived travel time does not exhibit counterintuitive results when used to characterize flow patterns resulting from stochastic network loading.

Many parallels can be drawn between the stochastic assignment paradox discussed above and Braess's paradox discussed in the deterministic UE context. Braess's paradox results from the assumption that motorists choose their route so as to minimize their *own* travel time. Consequently, there is no reason to expect the *total* travel time in the system always to decrease when the network is improved. In the stochastic network loading model, travelers are assumed to choose a route that will minimize their *perceived* travel time. Consequently, there is no reason to expect the total *measured* travel time to decrease when the system is improved.

This concludes the presentation of the stochastic network loading paradox. The rest of this section discusses the expected perceived travel time in relation to other measures that can be used to characterize the flow pattern over a network. This exposition may help the reader develop some intuition regarding the nature of stochastic network loading.

If var $(C_k^{rs}) = 0$, $\forall\ k \in \mathscr{K}_{rs}$, the perceived travel time between O–D pair r–s equals the minimum (measured) travel time between r and s. In this case the entire O–D flow will be assigned to the shortest travel-time path, as in the deterministic network loading model (i.e., the all-or-nothing assignment). The travel time experienced by each traveler between r and s will be $\min_k \{c_k^{rs}\}$. This quantity will always be larger than (or equal to) the expected perceived travel time, that is,

$$E\left[\min_k \{C_k^{rs}\}\right] \leq \min_k \{c_k^{rs}\} \qquad [10.32]$$

Inequality [10.32] can be proved by applying iteratively property [10.31b], of monotonicity with respect to the size of the choice set (see Problem 10.8a).

Under stochastic network loading, the fraction of flow using the kth path between origin r and destination s is given by $P_k^{rs}(\mathbf{c}^{rs})$. The average travel time for all flow between this O–D pair is given by $\sum_k P_k^{rs}(\mathbf{c}^{rs})c_k^{rs}$. This average travel time is always larger than (or equal to) the travel time experienced by each motorist under deterministic network loading. In other words,

$$\sum_k P_k^{rs}(\mathbf{c}^{rs})c_k^{rs} \geq \min_k \{c_k^{rs}\} \qquad [10.33]$$

as can be shown by the reader (see Problem 10.8b).

Equations [10.32] and [10.33] imply that

$$q_{rs} E\left[\min_k \{c_k^{rs}\}\right] \leq q_{rs} \min_k \{c_k^{rs}\} \leq q_{rs} \sum_k c_k^{rs} P_k^{rs}(\mathbf{c}^{rs}) \qquad [10.34]$$

The right-hand side of this equation represents the total *measured* travel time between origin r and destination s under stochastic loading, while the left-hand side represents the total *perceived* travel time for this O–D pair. The middle expression is the total travel time in the case of a deterministic network loading. (Both inequalities hold as equalities if the variance of the perceived travel times is zero.) Consider now the case of a network where all the travel times are deterministically known. In this case all motorists use the minimum path and the total travel time is $q_{rs} \min_k \{c_k^{rs}\}$. When stochastic elements are introduced into the route-choice criterion, the total measured travel time over the network will increase (over the all-or-nothing travel time), but the motorists' *perceived* travel time will decrease.

The Nature of the Flow Variables†

The last subject discussed in this section is somewhat more mathematical. It deals with the nature of the flow assigned to the network by the model discussed in this chapter.

The flow assigned to any route, under the assumptions of stochastic network loading, is a random variable. The distribution of this random variable can be understood by examining a simple binary-choice situation, with two routes connecting an origin node to a destination node. The O–D flow is

†The reader may skip this section without loss of continuity.

q and the routes are numbered 1 and 2, respectively. The O–D flow can be distributed between the two routes in many ways. Each possible flow pattern (F_1, F_2) can occur with some probability. (Assume for the purposes of this discussion that q, F_1, and F_2 are all integers.)

The probability that the first route will be chosen, by a given motorist, is P_1 (this is the probability that the perceived travel time on route 1 is less than the perceived travel time on route 2). The probability that F_1 motorists will choose route 1 and F_2 motorists will choose route 2 (where $F_1 + F_2 = q$) is[†]

$$\Pr\,[F_1, F_2] = \binom{q}{F_1} P_1^{F_1}(1 - P_1)^{q-F_1} \tag{10.35}$$

This is a binomial distribution with parameters q and P_1. The mean of this binomial distribution is qP_1, which is identical to the flow assigned to route 1 according to the stochastic network loading—Eq. [10.22a]. Thus stochastic network loading models give the *mean* flow on each path and the *mean* flow on each link of the network. These means were denoted in this chapter by f_k^{rs} for the flow on path k between r and s, and x_a for the flow on link a.

The distribution of the path flow variables can also be derived for a general network. The probability of obtaining a given flow pattern, $\mathbf{F}^{rs} = (\ldots, F_k^{rs}, \ldots)$, between origin r and destination s, is given by the multinomial density function with parameters q_{rs} and $\mathbf{P}^{rs} = (\ldots, P_k^{rs}, \ldots)$. In other words,

$$\Pr\,(\mathbf{F}^{rs}) = q_{rs}! \prod_{k \in \mathscr{K}_{rs}} \frac{(P_k^{rs})^{F_k^{rs}}}{F_k^{rs}!} \qquad \forall\, r, s \tag{10.36}$$

where

$$\sum_k F_k^{rs} = q_{rs}$$

The moments of the distribution of F_k^{rs} for a given O–D pair, r–s, are

$$E[F_k^{rs}] = q_{rs} P_k^{rs} \qquad \forall\, k \tag{10.37a}$$

$$\operatorname{var}\,[F_k^{rs}] = q_{rs} P_k^{rs}(1 - P_k^{rs}) \qquad \forall\, k \tag{10.37b}$$

$$\operatorname{cov}\,[F_k^{rs}, F_l^{rs}] = -q_{rs} P_k^{rs} P_l^{rs} \qquad \forall\, k, l \tag{10.37c}$$

The covariance between path flows connecting different O–D pairs is zero.

In principle, the joint density function of the flow on all links can be approximated from Eqs. [10.37] by a multivariate normal distribution. To do this, the joint distribution of all path flows is first approximated by a single multivariate normal distribution function. In other words,

$$\mathbf{F} \mathrel{\dot\sim} \mathrm{MVN}\,(\mathbf{f}, \boldsymbol{\Sigma}) \tag{10.38}$$

where $\mathbf{F} = (\ldots, \mathbf{F}^{rs}, \ldots)$, $\mathbf{f} = (\ldots, \mathbf{f}^{rs}, \ldots)$, and $\boldsymbol{\Sigma}$ is a block diagonal covariance matrix whose elements are given by Eqs. [10.37b] and [10.37c]. Let $\boldsymbol{\Delta}$ be

[†] In this section F is used to denote both the random variable and the value it may attain in a particular experiment.

the link-path incidence matrix of the network under consideration, that is, $\mathbf{\Delta} = (\ldots, \mathbf{\Delta}^{rs}, \ldots)$, where $\mathbf{\Delta}^{rs}$ is the link–path incidence matrix associated with O–D pair r–s. The vector of link flows, $\mathbf{X} = (\ldots, X_a, \ldots)$, can be derived from the values of the path flows by using the incidence relationships (see Eqs. [3.2]):

$$\mathbf{X} = \mathbf{F} \cdot \mathbf{\Delta}^T \qquad [10.39]$$

Due to the properties of the normal distribution (see the Appendix), the fact that $\mathbf{F}$ is normally distributed means that $\mathbf{X}$ will also be normally distributed, that is,

$$\mathbf{X} \sim \text{MVN}\,(\mathbf{f} \cdot \mathbf{\Delta}^T, \mathbf{\Delta} \cdot \mathbf{\Sigma} \cdot \mathbf{\Delta}^T) \qquad [10.40]$$

Accordingly, the mean of the flow on link a, x_a, is given by

$$E[X_a] = x_a = \sum_{rs} \sum_{k} f_k^{rs} \delta_{a,k}^{rs}$$

where f_k^{rs} is the mean flow on path k between r and s. This is the link flow that stochastic network loading models solve for.

10.3 SUMMARY

This chapter generalizes the deterministic network loading mechanism used in parts II and III to cases in which travelers do not perceive the travel time accurately. The perceived travel time on each path is modeled as a random variable distributed across the population of motorists. Each motorist is assumed to choose the path that he or she perceives to be the shortest.

This framework is analogous to the one used in discrete choice modeling to look at the choice among various alternatives. These models assume that individuals' preferences can be captured by a utility measure associated with each alternative. The utility measures cannot be observed and are therefore modeled as random variables. Each individual is then assumed to choose the alternative with the maximum utility. In the case of choice between paths, the disutility of each alternative path is its perceived travel time, which motorists try to minimize.

The first section of this chapter explains the foundation of discrete choice models, presenting some fundamental concepts, such as the choice function and the satisfaction. It also presents two specific model forms, the logit and the probit. The concepts presented in that section are used in the second section to formulate the stochastic network loading model. In this model, the probability that a given path connecting a particular O–D pair is chosen is the probability that its perceived travel time is the smallest. The second section also introduces the expected perceived travel time, which parallels the notion of the satisfaction function of discrete choice models. The expected perceived travel time measures the travel time that individuals expect to spend when traveling from their origin to their destination. The differences between stochastic and deterministic loading models are clarified, in this section, at two levels. First, the stochastic paradox phenomenon emphasizes the fact that in stochastic network loading models each motorist is assumed to use the route with the lowest *perceived* travel time. Second, as the last section demonstrates, the flows themselves are random variables under stochastic network loading. The output of the model (the link-flow pattern) consists of the means of these random variables.

10.4 ANNOTATED REFERENCES

Econometric models with discrete dependent variables have been used for modeling binary outcomes of experiments by many researchers, primarily in the area of toxicology and biology. These models are described, for example, by Finney (1964). The modern approach to these models and their interpretation in the context of discrete choice started in the late 1960s and early 1970s with the work of McFadden (1974) and others on logit models, including the development of estimation methods. This work was applied to urban travel demand modeling, as described in the book by Domencich and McFadden (1975). The first logit models were applied to mode choice problems, and later to trip distribution and other transportation-related problems. Tye et al. (1982) survey many of these applications in a report for the National Cooperative Highway Research Program. Currently, the logit model is used in a variety of applications including geography, transportation, market research, health care, and many others.

The modern interest in probit models, in the context of travel demand analysis, started in the late 1970s. Daganzo et al. (1977) have adopted Clark's (1961) approximation formulas to the computation of the probit choice function. The Monte-Carlo simulation technique for estimating the probit choice probabilities was developed at the same time by Lerman and Manski (1978). Both research teams (Daganzo et al., and Lerman and Manski) found that Clark's approximation dominated the simulation method in the range of problems considered. Other approximate analytical methods developed in the same time include the numerical integration method of Hausman and Wise (1978). Many of these methods are summarized in a survey paper by Sheffi et al. (1982). The state of the art (as of 1979) with regard to discrete choice models in general, and the multinomial probit model in particular, is summarized in Daganzo's (1979) book on the subject (this reference also includes many original contributions).

The concept of satisfaction in the context of discrete choice models was suggested by Daganzo (1979). For logit models, however, it was developed earlier, as a system evaluation criterion, by Williams (1977), as an accessibility measure by Ben-Akiva and Lerman (1978), and in the context of networks by

Sheffi and Daganzo (1978). The aggregation concept is discussed in many of the above-mentioned references, as well as by McFadden and Reid (1975), Koppelman (1976), and Bouthelier and Daganzo (1979).

Daganzo and Sheffi (1977) developed the view of stochastic network loading models in the context of discrete choice analysis. The stochastic paradox discussed in Section 10.2 was pointed out by Sheffi and Daganzo (1978a) in the context of logit route-choice models. The distribution of the flows under stochastic network loading models was formulated by Daganzo (1977a), who also showed how this distribution can be used to develop statistical tests on the possible values of the flow pattern.

10.5 PROBLEMS

10.1. A discrete choice model of travel mode to work includes two alternatives: driving alone (alternative 1) and taking transit (alternative 2). The model is based on the following utility functions:

$$U_1 = \phi_1 - t + \xi_1$$

$$U_2 = \phi_2 - \hat{t} + \xi_2$$

where ϕ_1 and ϕ_2 are model parameters. Both error terms follow a uniform distribution between b and $-b$.

(a) Discuss the relative magnitudes of ϕ_1 and ϕ_2 that you would expect.

(b) Develop an analytical expression for P_1 as a function of ϕ_1, ϕ_2, t, $\hat{t}$, and b.

(c) Plot P_1 as a function of $\hat{t} - t$, for several representative values of ϕ_1 and ϕ_2. Assume that $b = 1$. Discuss the effect of $(\phi_1 - \phi_2)$ on the choice probability.

(d) Assume that $\phi_1 - \phi_2 = 1$ and plot P_1 versus $\hat{t} - t$ for several values of b. Discuss the effect of b on the choice probability and draw general conclusions.

***10.2.** Based on the assumption that the utilities are identically and independently distributed (i.i.d.) Gumbel variates (see Eq. [10.6a] and the associated footnote), prove that the logit choice probability formula holds.

10.3. Consider the following discrete choice model:

$$U_1 = 1 - t + \xi_1$$

$$U_2 = -\hat{t} + \xi_2$$

Plot P_1 versus $(\hat{t} - t)$ if the error terms are i.i.d. Gumbel variates. Assume now that $\xi_1 \sim N(0, \pi^2/6)$, $\xi_2 \sim N(0, \pi^2/6)$ and cov $(\xi_1, \xi_2) = 0$. Plot P_1 versus $(\hat{t} - t)$ under these assumptions and compare with the logit curves. What can you conclude from these curves?

10.4. Show that the function $f(\mathbf{x}) = \max_i (x_i)$ is convex in x_i. Given that, prove the convexity property of the satisfaction function.

***10.5.** Prove Eq. [10.16a]. [*Hint:* Express $P_k(\mathbf{V})$ in integral form and think about the region of integration.]

***10.6.** Prove Eq. [10.16b]. (Note that the underlying distribution changes as well.)

10.7. Consider a logit automobile/transit mode split model with the following utility functions:

$$\text{Auto:} \quad U_1 = 5 + 2 \cdot \text{INC} - t + \xi$$

$$\text{Transit:} \quad U_2 = 2 - \hat{t} + \xi$$

where INC is an income variable.

(a) What is the choice probability of alternative 1 for a group of decision makers with income INC = 2.0 if $\hat{t} - t = 2$?

(b) What is the mode split if the population includes two groups: $\frac{2}{3}$ of the population have INC = 1.0 and $\frac{1}{3}$ have INC = 2.0?

(c) What is the mode split if the income is uniformly distributed between 0.5 and 2.5?

(d) Compute the satisfaction if INC = 2.0, $\hat{t} = 4$, and $t = 2$.

(e) Compute the satisfaction if INC = 1.0 for $\frac{2}{3}$ of the population and INC = 2.0 for the rest.

(f) Compute the satisfaction if INC $\sim U(0.5, 2.5)$. (Still assume that $\hat{t} = 4$, $t = 2$.)

10.8. **(a)** Prove that $E[\min_k \{C_k^{rs}\}] \leq \min_k \{c_k^{rs}\}$.

(b) Prove that $\sum_k P_k^{rs}(\mathbf{c}^{rs})c_k^{rs} \geq \min_k \{c_k^{rs}\}$.

10.9. Assume that the following logit model is used for assigning traffic between the origin and the destination of the network shown in Figure 10.2:

$$P_1 = \frac{e^{-c_1}}{e^{-c_1} + e^{-c_2}}$$

(a) Assume further that the total O–D flow is 1 unit. Plot the total travel time in the system, as a function of c_1, for $c_2 = 0.3$ time unit. Show the region where the stochastic paradox may happen.

(b) Plot the expected perceived travel time in this network as a function of c_1 for $c_2 = 0.3$. Comment on the shape of this curve.

(c) Assume now that the motorists know the travel time exactly. Plot the total travel time in the system as a function of c_1, for $c_2 = 0.3$ under that assumption. Compare this curve to the ones obtained in (a) and (b).

10.10. The "paradox" discussed in connection with Figure 10.4 may not occur for all stochastic network loading models. Under what condition would it not occur? In other words, for what type of stochastic loading models would the addition of the center link not increase the total travel time?

10.11. Consider, again, the network discussed in Problem 10.9. Assume that $c_1 = 2$ and $c_2 = 1$. Assume further that the O–D flow is 10 units. What is the expected flow on route 1? What is the probability that the flow on route 1 will be 4 units of flow? What is the probability that it will be 4 units of flow or less? Assume now that $\mathbf{q} = 10{,}000$ flow units. What is the expected flow on route 1? What is the probability that the flow on route 1 will be less than 5000 but more than 3000 flow units?

11

Stochastic Network Loading Models

Section 10.2 described a (stochastic) network loading approach in which motorists' perception of travel time is assumed to be random. This model generalizes the all-or-nothing network loading mechanism used in the first parts of this text. Stochastic network loading models are a special case of discrete choice models. To apply these models, the probability distribution function of the (perceived) travel time on each path has to be known so that the path choice probability can be calculated.

This chapter considers two specific route-choice models. The first is based on the multinomial logit formulation and the second is based on the multinomial probit formulation. The logit and probit route-choice models are each presented separately and then compared to each other in terms of several characteristics, including realism of representation and computational efficiency.

Note that the number of alternative paths connecting a typical O–D pair in a network of realistic size is extremely large. Consequently, it is not possible to enumerate all these paths, calculate the probability that each will be chosen, and compute the flow assigned to each. Both stochastic network loading algorithms presented in this chapter are, therefore, link-based procedures which avoid path enumeration and perform the assignment using only link and node variables.

11.1 LOGIT-BASED LOADING MODELS

The logit formula was presented in Section 10.1 as a random utility model. It is based on the assumption that the utilities of all the alternatives in the choice set are identically and independently distributed (i.i.d.) Gumbel variates. A logit route choice model can be derived from the assumption that the utility of using the kth path between origin r and destination s, U_k^{rs}, is given by

$$U_k^{rs} = -\theta c_k^{rs} + \varepsilon_k^{rs} \qquad \forall\ k, r, s \qquad [11.1]$$

where c_k^{rs} is the measured travel time, θ is a positive parameter, and ε_k^{rs} is a random term, the distribution of which is given by the Gumbel density function. The random variable ε_k^{rs} in Eq. [11.1] is related to the random term in the percevied route travel time mentioned in Section 10.2 (the term ξ_k^{rs} in Eq. [10.20]) by a simple transformation: $\varepsilon_k^{rs} = -\theta\xi_k^{rs}$, $\forall\ r, s$. Since the error terms associated with all paths are assumed to be identically distributed, ε_k^{rs} can be replaced by ε^{rs} in Eq. [11.1]. Following Eq. [10.6a], this specification of the utility function implies that

$$P_k^{rs} = \frac{e^{-\theta c_k^{rs}}}{\sum_l e^{-\theta c_l^{rs}}} \qquad \forall\ k, r, s \qquad [11.2]$$

Instead of using the term "utility" in discussing stochastic network loading problems, Section 10.2 used the notion of perceived travel time. The perceived travel time on the kth path between origin r and destination s is C_k^{rs}, where (see Eq. [10.20])

$$C_k^{rs} = c_k^{rs} + \xi_k^{rs} \qquad \forall\ k, r, s$$

Using the transformation of ξ_k^{rs} mentioned above, the perceived path travel time can be expressed as

$$C_k^{rs} = c_k^{rs} - \frac{1}{\theta}\varepsilon^{rs} \qquad \forall\ k, r, s \qquad [11.3]$$

Again, the random components, ε, are assumed to be i.i.d. Gumbel variates.

The parameter θ is a constant that scales the perceived travel time. As mentioned in Section 10.1, the logit choice probabilities depend on utility differences; without proper scaling the choice probabilities would have depended on the units of measurement of travel time. For the logit model it can be shown that this parameter is inversely proportional to the standard error of the distribution of the perceived path travel times.† To understand the effect of this parameter consider a given O–D pair connected by many paths. Assume further that the route shares are given by a logit model such as formula [11.2]. If θ is very large, the perception error is small and users will tend to select the minimum measured travel-time path. A small value of θ indicates a large perception variance, with travelers using many routes, including some that may be significantly longer (in terms of measured travel time) than the true shortest path. In the limit, where $\theta \to 0$, the share of flow on all paths will be equal, regardless of path travel times.

This section describes an algorithmic approach for logit-based network

†If ε^{rs} in Eq. [11.3] is a random Gumbel variate, var $(C_k^{rs}) = \pi^2/6\theta^2$.

loading. This model is then critically evaluated in the context of what is required by a stochastic assignment model.

STOCH Algorithm

The algorithm given here is known in the literature as the STOCH method or *Dial's algorithm*. The procedure effectively implements a logit route-choice model at the network level. It does not, however, assign choice probabilities (and flows) to *all* paths connecting each O–D pair. Instead, it assumes that many of these paths constitute unreasonable travel choices that would not be considered in practice. Consequently, the STOCH procedure includes a preliminary phase which identifies the set of "reasonable" paths connecting each O–D pair. The O–D flow is then only assigned to these paths, using the logit formula with parameter θ.

Figure 11.1a depicts a small contrived network which is used throughout this section to illustrate the various steps of the algorithm. The network includes 9 nodes and 12 links, the travel time on which is shown in the figure. Assume now that the problem is to assign a flow of 1000 trips per time unit, between node 1 and node 9, which are connected by six alternative paths. The links of this network are denoted by their end nodes; namely, a link leading from (upstream) node i to (downstream) node j is denoted by $i \rightarrow j$.

The steps of the algorithm are described below. The algorithm defines a path between some O–D pair as "reasonable" if it includes only links that take the traveler further away from the origin and closer to the destination. Such links can be identified by associating two labels with each node i: $r(i)$ and $s(i)$. The first label, $r(i)$, denotes the travel time from origin node r to node i along the minimum travel time path, while $s(i)$ denotes the travel time from node i to destination node s along the minimum path. Each reasonable path includes only links $i \rightarrow j$ such that $r(i) < r(j)$ and $s(i) > s(j)$.

The steps of this algorithm for one O–D pair (r–s) are outlined below. These steps should be repeated for each origin–destination pair in the network.

Step 0: *Preliminaries:*

(a) Compute the minimum travel time from node r to all other nodes. Determine $r(i)$ for each node i.

(b) Compute the minimum travel time from each node i to node s. Determine $s(i)$ for each node i.

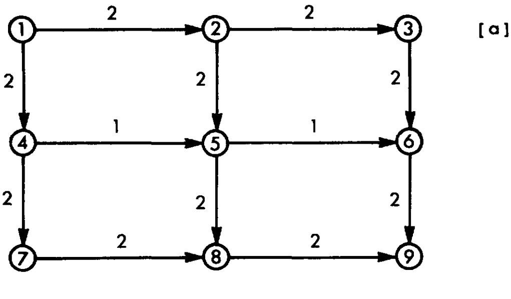


Figure 11.1 Network example demonstrating the application of STOCH: (a) the network, including link travel times.

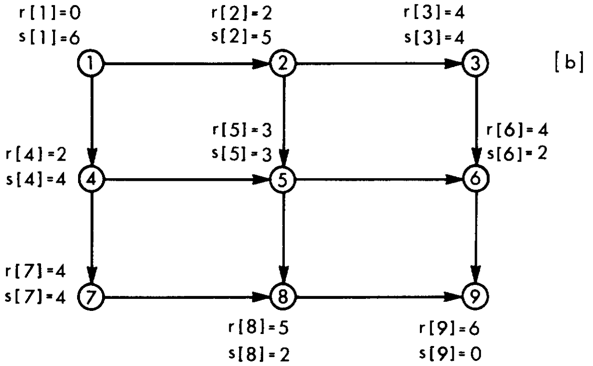


Figure 11.1(b) The forward and backward node potentials $r(i)$ and $s(i)$ (node 1 is the origin, r, and node 9 is the destination, s).

(c) Define $\mathcal{O}_i$ as the set of downstream nodes of all links leaving node i.

(d) Define $\mathcal{F}_i$ as the set of upstream nodes of all links arriving at node i.

(e) For each link $i \to j$ compute the "link likelihood", $L(i \to j)$, where

$$L(i \to j) = \begin{cases} e^{\theta[r(j) - r(i) - t(i \to j)]} & \text{if } r(i) < r(j) \text{ and } s(i) > s(j) \\ 0 & \text{otherwise} \end{cases} \qquad [11.4]$$

In this expression $t(i \to j)$ is the (measured) travel time on link $i \to j$.

Figure 11.1b shows the values of $r(\cdot)$ and $s(\cdot)$ for all nodes, as calculated for the example network under consideration. Note that according to these labels not all of the six paths connecting node 1 to node 9 can be considered reasonable. For example, whereas path $1 \to 2 \to 5 \to 6 \to 9$ is reasonable, path $1 \to 4 \to 7 \to 8 \to 9$ is not, because $s(4) = s(7)$.

Figure 11.1c depicts the link likelihoods (with $\theta = 1$) for all the links in this example. Note that $L(i \to j) = 1$ for all the links on the minimum path between the origin and the destination, while $L(i \to j) = 0$ for all the "unreasonable" links.

Step 1: ***Forward pass.*** Consider nodes in ascending order of $r(i)$, starting with the origin, r. For each node, i, calculate the "link weight,"

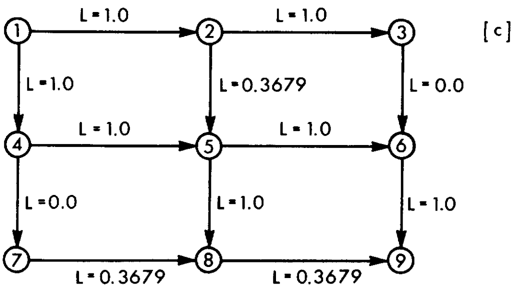


Figure 11.1(c) The link likelihoods (for $\theta = 1$).

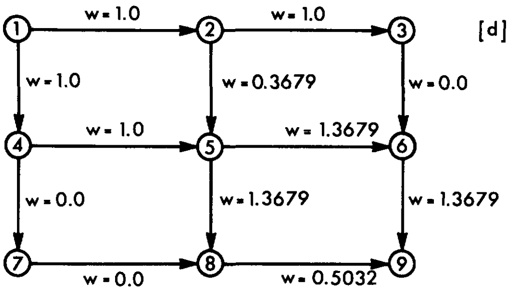


Figure 11.1(d) The link weights.

$w(i\to j)$, for each $j \in \mathcal{O}_i$ (i.e., for each link emanating from i), where

$$w(i\to j) = \begin{cases} L(i\to j) & \text{if } i = r \quad \text{(i.e., if node } i \text{ is the origin)} \\ L(i\to j) \sum\limits_{m\in\mathcal{F}_i} w(m\to i) & \text{otherwise} \end{cases} \qquad [11.5]$$

When the destination node, s, is reached, this step is completed. The link weights corresponding to the link likelihoods shown in Figure 11.1c are shown in Figure 11.1d. To see how these were calculated consider, for example, the weight calculation for link 5→ 8. Following Eq. [11.5], $w(5\to 8)$ is given by

$$w(5\to 8) = L(5\to 8) \sum_{m\in\mathcal{F}_5} w(m\to 5)$$

where $\mathcal{F}_5$ includes nodes 2 and 4 (since links 2→ 5 and 4→ 5 enter node 5). Thus

$$w(5\to 8) = 1.0000(0.3679 + 1.0000) = 1.3679$$

Step 2: ***Backward pass***. Consider nodes in ascending values of $s(j)$ starting with the destination, s. When each node, j, is considered, compute the link flow $x(i\to j)$ for each $i \in \mathcal{F}_j$ (i.e., for each link entering j), by following the assignment:

$$x(i\to j) = \begin{cases} q_{rs} \dfrac{w(i\to j)}{\sum\limits_{m\in\mathcal{F}_j} w(m\to j)} & \text{for } j = s \quad \begin{array}{l}\text{(i.e., if node } j \text{ is the} \\ \text{destination)}\end{array} \\ \left[\sum\limits_{m\in\mathcal{O}_j} x(j\to m)\right] \dfrac{w(i\to j)}{\sum\limits_{m\in\mathcal{F}_j} w(m\to j)} & \text{for all other links } i\to j \end{cases} \qquad [11.6]$$

This step is applied iteratively until the origin node is reached. Note that the sum in the denominator of the quotient includes all links arriving at the downstream node of the link under consideration. The sum of the flow variables is taken over all links emanating at the downstream node of the link under consideration.

The link flows for the example network under consideration (assuming

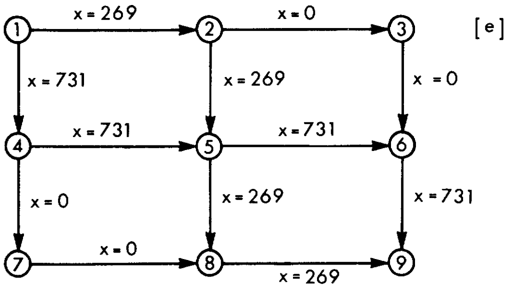


Figure 11.1(e) The link flows.

that the O–D flow equals 1000) are given in Figure 11.1e. The calculation of these flows can be illustrated by showing how $x(2 \to 5)$ has been determined. When the flow on link $2 \to 5$ is to be calculated, the flows on links $(8 \to 9)$, $(6 \to 9)$, $(5 \to 6)$, and $(5 \to 8)$ are already known. Thus

$$x(2 \to 5) = [x(5 \to 6) + x(5 \to 8)] \frac{w(2 \to 5)}{w(2 \to 5) + w(4 \to 5)}$$

$$= (731 + 269) \frac{0.3679}{0.3679 + 1.0} = 269$$

This concludes the description of the mechanics of STOCH.

The flow generated by this algorithm is equivalent to a logit-based path assignment between every origin and every destination, given that only reasonable paths are considered. To see this, note that each link likelihood, $L(i \to j)$, is proportional to the (logit model) probability that link $i \to j$ is used by a traveler chosen at random from among the population of trip-makers between r and s, given that the traveler is at node i. The probability that a given *path* will be used is proportional to the product of all the likelihoods of the links comprising this path. The probability of using path k between r and s, P_k^{rs}, is then (see Problem 11.3)

$$P_k^{rs} = K \prod_{ij} [L(i \to j)]^{\delta_{ij,k}^{rs}} \qquad [11.7]$$

where K is a proportionality constant, and the product is taken over all links in the network. The incidence variable, $\delta_{ij,k}^{rs}$ ensures that P_k^{rs} will include only those links in the kth path between r and s. Substituting the expression for the likelihood (Eq. [11.4]) in the last equation, the choice probability of choosing a particular (reasonable) path becomes

$$P_k^{rs} = K \prod_{ij} \exp \{\theta[r(j) - r(i) - t(i \to j)]\delta_{ij,k}^{rs}\}$$

$$= K \exp \left\{\theta \sum_{ij} [r(j) - r(i) - t(i \to j)]\delta_{ij,k}^{rs}\right\}$$

$$= K \exp \{\theta[u_{rs} - c_k^{rs}]\} \qquad [11.8]$$

The last equality results from the two following summations:

$$\sum_{ij} [r(j) - r(i)]\delta^{rs}_{ij,k} = r(s) - r(r) = u_{rs} \qquad [11.9a]$$

where u_{rs} is the travel time along the shortest path from r to s, and

$$\sum_{ij} t(i \rightarrow j)\delta^{rs}_{ij,k} = c^{rs}_k \qquad [11.9b]$$

In order for Eq. [11.8] to be a proper probability statement, the proportionality constant has to be set such that

$$\sum_k P^{rs}_k = 1.0$$

This means that

$$K = \frac{1}{\sum_l e^{\theta[u_{rs} - c^{rs}_l]}}$$

whence

$$P^{rs}_k = \frac{e^{\theta[u_{rs} - c^{rs}_k]}}{\sum_l e^{\theta[u_{rs} - c^{rs}_l]}} = \frac{e^{-\theta c^{rs}_k}}{\sum_l e^{-\theta c^{rs}_l}} \qquad [11.10]$$

Equation [11.10] depicts a logit model of route choice among the reasonable paths connecting O–D pair r–s.

Table 11.1 lists the six alternative paths along with their respective choice probabilities and flows for the example under consideration. When these path flows are assigned to the network (see Figure 11.2), the resulting link flow pattern is identical to that generated by the algorithm.

The STOCH algorithm does not require path enumeration. It does require, however, two minimum-path calculations for every O–D pair in the network, meaning that it is not practical for large networks. Furthermore, the algorithm does not take advantage of the nature of minimum-path procedures, which generate the shortest path from a single origin (the root node) to *all* network nodes.

A more efficient algorithm can be devised by redefining the criteria for

TABLE 11.1 Path Choice Probabilities and Path Flows

Path	Travel Time	P^{rs}_k	Path Flow
1 → 2 → 3 → 6 → 9	8	0	0
1 → 2 → 5 → 6 → 9	7	0.197	197
1 → 2 → 5 → 8 → 9	8	0.072	72
1 → 4 → 5 → 6 → 9	6	0.534	534
1 → 4 → 5 → 8 → 9	7	0.197	197
1 → 4 → 7 → 8 → 9	8	0	0

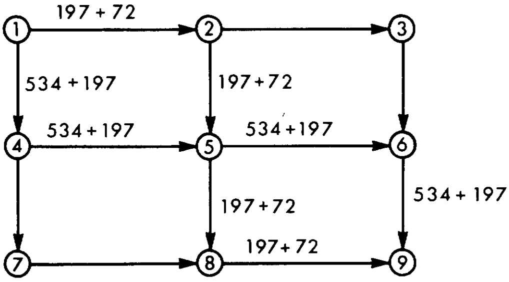


Figure 11.2 Assignment of logit-based flows to the network example of Figure 11.1.

reasonable paths. Under the new definition, a path would be considered reasonable if it includes only links that do not take the traveler back toward the origin. In other words, a link $(i \rightarrow j)$ is reasonable if and only if $r(i) \leq r(j)$. The second part of the original requirement, that these links bring the traveler closer to the destination, is dropped. In fact, $s(i)$ need not be computed at all.

The algorithm can now be executed for each origin, setting the link likelihood to zero whenever $r(i) > r(j)$. The link weights are computed exactly as before. The only change is in step 2, where the volume destined to a particular node should be added to the outbound flow (leaving that node) when the link is considered. This step starts with the most distant destination and proceeds in descending values of $r(j)$. For each link $i \rightarrow j$, the flow is determined by the following expression (replacing Eq. [11.6]):

$$x(i \rightarrow j) = \left[q_{rj} + \sum_{m \in \mathcal{O}_j} x(j \rightarrow m) \right] \frac{w(i \rightarrow j)}{\sum_{m \in \mathcal{F}_j} w(m \rightarrow j)} \qquad [11.11]$$

where q_{rj} is the trip rate from the origin node, r, to node j [and $x(j \rightarrow m)$ is defined as zero if $\mathcal{O}_j$ is empty (i.e., for destinations on the edge of the network)]. The step terminates when the origin node, r, is reached. At this point, the assignment of flow from the origin to *all* destinations is complete.

When applied to the example network used throughout this section, the modified algorithm results in the flows depicted in Figure 11.3. This flow pattern differs from the one generated by the original algorithm due to the different definition of reasonable path. According to the modified criterion, all six paths are reasonable.

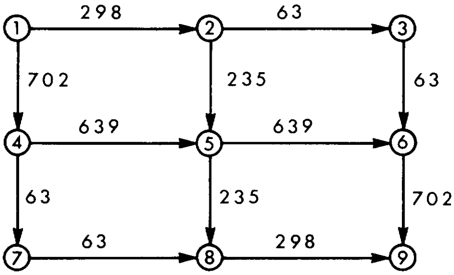


Figure 11.3 Link-flow pattern generated by the single-pass STOCH algorithm.

This modified procedure is known as the "single-pass" algorithm. Its computational requirements are only modestly higher than those of an all-or-nothing procedure. Consequently, it can be applied to a realistic-size network. Furthermore, there is no a priori behavioral reason or any empirical evidence that would favor the original (double-pass) algorithm over the single-pass version. Computational consideration should, therefore, prevail in logit-based traffic assignment.

Critique of Logit-Based Network Loading

The STOCH algorithm assigns flow to a transportation network according to a logit route-choice model. This model, however, may exhibit some unreasonable properties resulting from some of the assumptions underlying the logit choice probability function.

To illustrate the problem, consider the simple network shown in Figure 11.4a. This network includes three routes connecting an origin node to a destination node. The travel time on each of these routes is 1 unit. The bottom two routes overlap over a portion of their length which is designated, in the figure, by ρ. According to a logit-based assignment model, the flow on all these three routes would be equal. In other words, each route would carry one-third of the total O–D flow.

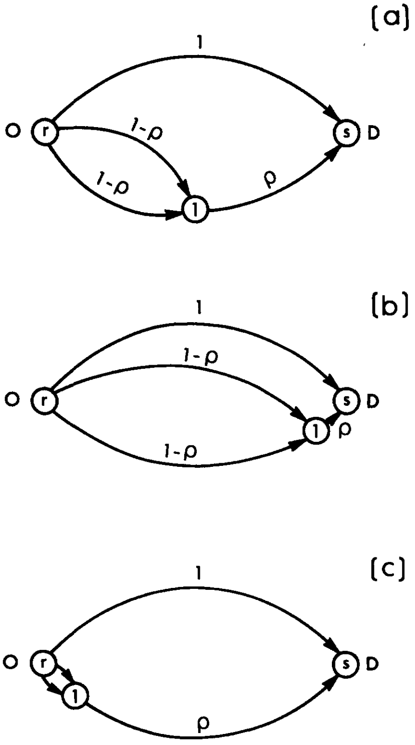


Figure 11.4 Example network used to illustrate the inability of logit-based assignment to account for network structure: (a) a network with three paths, two of which overlap by ρ time units (the numbers on the links represent travel times); (b) small overlap between the bottom paths; (c) large overlap between the bottom paths.

This assignment may seem reasonable, especially if the amount of overlap between the bottom routes, ρ, is relatively small. In this case, shown in Figure 11.4b, it may be reasonable to assume that most travelers perceive the three routes as three distinct alternatives with similar travel times. If, however, the amount of overlap is large, this may not be the case. For example, Figure 11.4c depicts a situation in which ρ is very large; in this case travelers may perceive the bottom two routes as a single alternative. Surely, at the limit (as $\rho \rightarrow 1$) this will happen and each remaining route (top and bottom) will carry one-half of the flow. When $0 \leq \rho \leq 1$, the top route should carry between one-half and one-third of the flow, and each of the bottom routes should carry between one-fourth and one-third of the flow. The logit model, which assigns one-third of the flow to each route regardless of the network topology, is thus overestimating the flow on overlapping routes.

The nature of this deficiency of logit-based assignment methods can be further understood by considering, again, Figure 11.4c. In the limit, when the overlap is large, the coding of the bottom corridor as a single route or as two separate routes may be only a matter of ad hoc procedures or coding convenience. Such considerations should not affect the resulting flow. If the logit network loading procedure is used, however, the flow patterns resulting from coding the bottom corridor as a single path or as two alternative paths will be significantly different from each other.

This deficiency of the logit model is not unique to route-choice problems. Consider, for instance, an application of the logit model discussed in Section 10.1 to the case of mode choice between automobile and rail transit. Now, it is possible to view the transit alternative as two distinct options (say, the odd-numbered rail cars and the even-numbered rail cars). This would mean that the choice is among three alternatives: automobile, odd-numbered cars, and even-numbered cars (the last two alternatives having identical utility functions). Even though such a representation should not change the actual transit mode share, the application of the logit model to the three alternatives' formulation would result in a larger transit mode share than when transit is modeled as a single alternative.

Like the route-choice example discussed above, this mode choice anomaly results from the logit's inability to account for correlation between the alternatives' utilities. This inability stems from the *independence* assumption with regard to the utilities' random components. In the mode choice example given above, the utilities of the two transit alternatives are completely correlated (since they represent the same physical mode), while in the aforementioned route-choice example, the perceived travel times on the two bottom routes are correlated by virtue of their overlapping segment. In the mode choice example, this correlation means that if the odd-numbered cars are perceived to be, say, faster than they actually are, the even-numbered cars will also be perceived this way by the same choice maker. In the network example, the correlation means that if a driver perceives one of the bottom routes to be faster than it really is, this driver is also likely to consider the other bottom

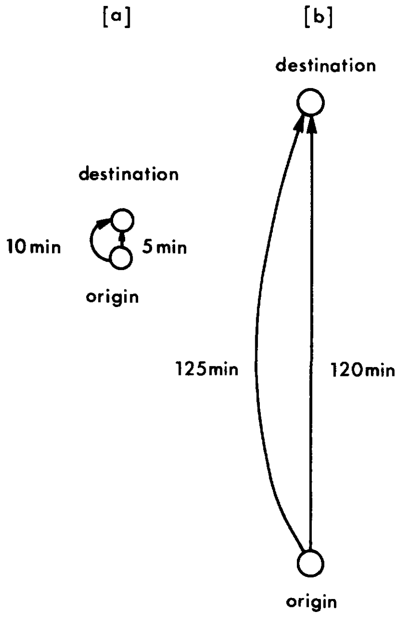


Figure 11.5 Two-route network example: (a) two short routes; (b) two long routes.

route as faster. The reason for this is that the error may be in the perception of travel time on the overlapping segment, in which case both bottom routes would be equally affected. Consequently, the extent of this correlation is determined by the relative length of the common segment. Thus, there is a need to account for the correlation between the perceived travel times of the various routes in order to properly reflect the topology of the network. This cannot be accomplished with the logit model.

Another possible anomaly that may arise in conjunction with a logit route-choice model is due to the fact that the path choice probabilities are determined solely on the basis of travel-time differences. To see the problem consider, for example, two binary-route-choice cases, such as the ones shown in Figure 11.5. In the first case (shown in Figure 11.5a) the two routes' travel times are 5 and 10 minutes, respectively, while in the second case (Figure 11.5b) the two travel times are 120 and 125 minutes, respectively. In the first case, it is likely that almost all motorists will use the shorter (5 minutes) path, while in the second case, it is likely that a significant number of travelers† would use the longer (125-minute) path rather than the 120-minute path. This expected route-choice behavior is due to the variance in perception, which is likely to be related to the length of the routes. In other words, the second scenario (125 minutes versus 120 minutes) offers more opportunity for "wrong" perception (in the sense of choosing the longer travel-time route) than the first case (10 minutes versus 5 minutes). In contrast with the route

†Less than half of the total flow, though.

choice behavior mentioned above, an application of the logit model to the two cases shown in Figure 11.5 would result in the same flow pattern. In other words, the flow share of the longer route in Figure 11.5a would be the same as the flow share of the longer route in Figure 11.5b.

Again, this deficiency of the logit is not unique to network loading models. It is rooted in the logit's underlying assumption that the utilities' error terms are *identically* distributed with the same, fixed variances. The logit model is not applicable in cases where the error terms' distribution depends on the alternatives' characteristics or on the systematic part of the utility functions. In the aforementioned example, the distribution of the perceived path travel times should be a function of the respective lengths of the alternative routes, whereby longer routes are associated with larger variance of perception. Such a characteristic, however, cannot be accommodated by the logit model.

The above-mentioned limitations of the logit model should not cloud the fact that the single pass STOCH algorithm is remarkably efficient, requiring an amount of computational effort that is only modestly higher than the simplistic all-or-nothing assignment models. There may be network problems to which logit is applicable, in which case the algorithm or the ideas behind it should be utilized. Furthermore, the logit model itself is widely used in travel demand analysis, as well as in other fields. The properties mentioned above do not generally bias the results of the model in cases in which the perceived utilities of the alternatives considered are only slightly correlated and when their probability distributions are similar.

The next section discusses a probit-based network loading procedure that alleviates many of the difficulties associated with logit-based network loading models. This is accomplished, however, at increased computational costs.

11.2 PROBIT-BASED LOADING MODEL

The probit model was presented in Section 10.1 as a discrete choice model based on random utility theory. This section applies the probit model to the choice of route between an origin and a destination, where the perceived path travel times are normally distributed. The first part of this section shows that the path travel-time distribution can be derived from the probability distribution of perceived link travel times. These relationships between the distribution of the perceived travel times over links and paths respectively are utilized in the derivation of a probit-based stochastic network loading algorithm. The second part of the section describes the algorithm, which is based on a Monte Carlo simulation of the perceived link travel times. Finally, the probit model is compared to the logit model in terms of the applicability of both approaches to route-choice situations, as well as in terms of computational efficiency.

Derivation of Perceived Path Travel Times

As mentioned above, the probit route-choice model assumes that the perceived path travel time along any given path is normally distributed with a mean equal to the actual (measured) travel time. This distribution can be derived from the distribution of the link travel times as follows.

Let T_a denote the travel time on link a, as perceived by a motorist randomly chosen from the population of drivers. T_a is a random variable that is assumed to be normally distributed with mean equal to the measured link travel time and with variance that is proportional to the measured link travel time. In other words,

$$T_a \sim N(t_a, \beta t_a) \qquad [11.12]$$

where β is a proportionality constant. In fact, β may be interpreted as the variance of the perceived travel time over a road segment of unit travel time. Assuming that nonoverlapping road segments are perceived independently, the variance of a segment with travel time t_a would be βt_a, as indicated by expression [11.12].

Note that the uncertainty in the perception of travel time can be assumed to be related to the link length rather than its travel time. Consequently, the variance in Eq. [11.12] can be defined to be, for example, βl_a, where l_a is the length of link a (and β is the variance of the perceived travel time over a road segment of unit length). Alternatively, the variance can be defined as βt_a^0 where t_a^0 is the free flow travel time on link a [i.e., $t_a^0 = t_a^0(0)$ and β is interpreted as the variance of the perceived travel time over a road segment of unit travel time at free flow]. In this chapter it is assumed that var $(T_a) = \beta t_a$, for simplicity.

The normality assumption may not be entirely appropriate for modeling the distribution of the perceived link travel times, since these times cannot be negative. Consequently, a nonnegative distribution with long tail to the right (positive skewness), such as the gamma distribution, could have been more appropriate. It is, however, reasonable to believe that the relative magnitude of the perception error is small with respect to t_a, and thus, even if the gamma distribtuion were used, the calculations are in the region where gamma distributions are well approximated by normal distributions. The assumption that the perceived link travel times are normally distributed is thus not unreasonable.† Note that since the perceived travel times of nonoverlapping road segments are assumed to be independent, the random variables $\{T_a\}$ are mutually independent.

The distribution of the perceived *path* travel times can now be obtained from the incidence relationships, that is,

$$C_k^{rs} = \sum_a T_a \delta_{a,k}^{rs} \qquad \forall\ k, r, s \qquad [11.13]$$

†It should be noted that the algorithm described below is applicable to any link time distribution, including gamma.

Equation [11.13] states that the perceived travel time on route k between r and s is the sum of the perceived travel times on all the links comprising this path. Due to the properties of the normal distribution, the perceived path travel times will also be normally distributed with mean, c_k^{rs}, given by

$$c_k^{rs} = \sum_a t_a \delta_{a,k}^{rs} \qquad [11.14a]$$

and a variance given by

$$\text{var}\,(C_k^{rs}) = \sum_a \beta t_a \delta_{a,k}^{rs} = \beta c_k^{rs} \qquad [11.14b]$$

This follows directly from Eqs. [11.12] and [11.13]. The path travel times, however, are not independent: If two paths, k and l, have some links in common, their respective perceived travel times should be correlated. The covariance between the perceived travel times of two paths is thus related to their common portion, that is,

$$\text{cov}\,(C_k^{rs}, C_l^{rs}) = \sum_a \beta t_a \delta_{a,k}^{rs} \delta_{a,l}^{rs} = \beta c_{k,l}^{rs} \qquad [11.14c]$$

where $c_{k,l}^{rs}$ is the measured travel time on the common segments of paths k and l between origin r and destination s.

The distribution of the perceived path travel times can also be derived from the properties of the multivariate normal distribution by using matrix notation.† Let $\mathbf{T} = (\ldots, T_a, \ldots)$, denote the perceived link travel-time vector and assume that $\mathbf{T}$ is normally distributed, that is,

$$\mathbf{T} \sim \text{MVN}\,(\mathbf{t}, \mathbf{\Sigma}') \qquad [11.15]$$

where $\mathbf{t} = (\ldots, t_a, \ldots)$, and $\mathbf{\Sigma}' = [\beta \cdot \mathbf{t} \cdot \mathbf{I}]$, where $\mathbf{I}$ is the identity matrix. The link covariance matrix in Eq. [11.15] is diagonal, with the nonzero entries being βt_a. (In other words, this covariance matrix includes only the variance terms as its nonzero entries. All the covariance terms are zero.) Let $\mathbf{\Delta}^{rs}$ be the link-path incidence matrix for O–D pair r–s, as defined in Eq. [3.2] (i.e., the elements of this matrix are $\delta_{a,k}^{rs}$). The vector of perceived travel times for the paths joining origin r and destination s, $\mathbf{C}^{rs} = (\ldots, C_k^{rs}, \ldots)$, is given by $\mathbf{C}^{rs} = \mathbf{T} \cdot \mathbf{\Delta}^{rs}$. The joint density function of this vector is given by

$$\mathbf{C}^{rs} \sim \text{MVN}\,(\mathbf{c}^{rs}, \mathbf{\Sigma}^{rs}) \qquad \forall\, r, s \qquad [11.16a]$$

where

$$\mathbf{c}^{rs} = \mathbf{t} \cdot \mathbf{\Delta}^{rs} \qquad \forall\, r, s \qquad [11.16b]$$

$$\mathbf{\Sigma}^{rs} = \mathbf{\Delta}^{rs^T} \cdot [\beta \mathbf{t} \cdot \mathbf{I}] \cdot \mathbf{\Delta}^{rs} \qquad \forall\, r, s \qquad [11.16c]$$

The diagonal terms of $\mathbf{\Sigma}^{rs}$ are the variances described in Eq. [11.14b], while the off-diagonal terms are the covariances given in Eq. [11.14c].

†Readers who do not find matrix algebra useful can skip this paragraph or consult the Appendix.

Once the density function of the perceived path travel times is known, it can be used to determine the share of O–D flow that should be loaded onto each path. The path-choice probability cannot, however, be calculated by using any of the analytical approximation methods discussed in Section 10.1, including Clark's approach. The reason is that these methods require path enumeration; a computationally prohibitive task for networks of the size typically used in urban transportation planning. Furthermore, even in small networks in which paths could be enumerated, Clark's approximation cannot be applied since it is computationally expensive to use for more than 10 or 20 alternative choices (it is also inaccurate in these cases). Consequently, the only feasible method in this context is an adaptation of the Monte Carlo simulation procedure described in Section 10.1.

Algorithm

The algorithm for probit-based stochastic network loading is relatively simple. At each iteration the density function of the perceived travel time of each link is sampled once. This results in a set of realizations (drawings) of perceived link travel times which is used in an all-or-nothing assignment. The flow between each origin and each destination is assigned to the shortest travel-time path between the origin and the destination based on the simulated perceived link travel times. This process of sampling and assignment is then repeated several times. When the process terminates, the results of the individual iterations are averaged for each link to yield the final flow pattern.

The advantage of the simulation method is that it does not require the sampling of the perceived *path* travel times; only perceived link travel times are sampled at every iteration thus avoiding path enumeration. The perceived path times are built "automatically" as the minimum paths are generated. The perceived travel time used in a particular iteration for a given path is the sum of the perceived link times drawn, in the simulation process, from the density functions associated with the links along this path.

The Monte Carlo simulation process of drawing perceived travel times may, in theory, produce negative perceived link travel times. Such a travel time may "throw off" the shortest-path routine used in the all-or-nothing assignment. In practice, however, the probability of a negative link travel time is negligible and consequently, the normal density function can be truncated at zero with no significant loss of accuracy.

The number of sampling/assignment iterations to be performed can be decided on the basis of the variance of the average link flows at each iteration. To do this, let $X_a^{(l)}$ denote the flow on link a, resulting from the lth application of the sampling–assignment sequence.† The link flow estimate at this point is

†$X_a^{(l)}$ can be viewed as a realization drawn from the link flow probability density function discussed at the end of Section 10.2.

$x_a^{(l)}$, where

$$x_a^{(l)} = \frac{1}{l} \sum_{m=1}^{l} X_a^{(m)} \qquad [11.17a]$$

The (sample) standard deviation for every link a in the sample is given by

$$\sqrt{\frac{1}{l-1} \sum_{m=1}^{l} (X_a^{(m)} - x_a^{(l)})^2}$$

Let $\sigma_a^{(l)}$ denote the standard deviation of $x_a^{(l)}$. In other words, $\sigma_a^{(l)}$ is given by

$$\sigma_a^{(l)} = \sqrt{\frac{1}{l(l-1)} \sum_{m=1}^{l} (X_a^{(m)} - x_a^{(l)})^2} \qquad \forall\ a \qquad [11.17b]$$

This standard deviation can be used to develop a stopping criterion for the algorithm. For example, the algorithm could be terminated if

$$\max_a \{\sigma_a^{(l)}\} \le \kappa \qquad [11.18a]$$

where κ is a prespecified constant indicating the required accuracy. Alternatively, the stopping criterion can be based, for example, on the magnitude of the variance relative to the magnitude of the mean flow, that is

$$\frac{\sum_a \sigma_a^{(l)}}{\sum_a x_a^{(l)}} \le \kappa' \qquad \text{or} \qquad \max_a \left\{\frac{\sigma_a^{(l)}}{x_a^{(l)}}\right\} \le \kappa'' \qquad [11.18b]$$

where κ' and κ'' are some other (nondimensional) constants. The algorithm can now be summarized as follows:

Step 0: *Initialization.* Set $l := 1$.

Step 1: *Sampling.* Sample $T_a^{(l)}$ from $T_a \sim N(t_a, \beta t_a)$ for each link a.†

Step 2: *All-or-nothing assignment.* Based on $\{T_a^{(l)}\}$, assign $\{q_{rs}\}$ to the shortest path connecting each O–D pair r–s. This step yields the set of link flows $\{X_a^{(l)}\}$.

Step 3: *Flow averaging.* Let $x_a^{(l)} = [(l-1)x_a^{(l-1)} + X_a^{(l)}]/l$, $\forall\ a$.

Step 4: *Stopping test*

(a) Let

$$\sigma_a^{(l)} = \sqrt{\frac{1}{l(l-1)} \sum_{m=1}^{l} [X_a^{(m)} - x_a^{(l)}]^2} \qquad \forall\ a$$

(b) If $\max_a \{\sigma_a^{(l)}/x_a^{(l)}\} \le \kappa$, stop. The solution is $\{x_a^{(l)}\}$. Otherwise, set $l := l + 1$ and go to step 1.

†Recall that this distribution may be truncated at zero.

Note that the flow has to be averaged at every iteration so that the standard error can be compu'ed in order to test for convergence. The sequence of flow averages does not have to be stored for each link, though. Step 3 demonstrates how $x_a^{(l)}$ is computed from $x_a^{(l-1)}$ and $X_a^{(l)}$ rather than directly from the sequence $X_a^{(1)}, X_a^{(2)}, \ldots, X_a^{(l)}$. The standard error of the link-flow estimate can also be calculated without storing all the link flows in the computer's memory, by using $\sigma_a^{(l-1)}$, $x_a^{(l)}$, and $X_a^{(l)}$ to compute $\sigma_a^{(l)}$ (see Problem 11.5).

The number of iterations required for this procedure to converge is not too high for large networks. Usually, an acceptable level of accuracy can be achieved after four to six iterations. This happens because in large networks each path includes many links, the perceived travel time of which is sampled independently at each iteration. These sampled times are added to each other when the perceived path times are calculated, generating, in effect, an averaging process that tends to lead to a relatively rapid convergence of the simulation procedure.

Comparison of Probit and Logit Loading Models

The last part of Section 11.1 included a critical evaluation of logit-based route-choice models. Such models were criticized on two grounds. The first is the lack of sensitivity to network topology, a problem that results in the assignment of too much flow to overlapping routes. The second is the dependence of the route-choice probabilities on travel-time differences alone. This section shows how both concerns can be alleviated by the use of the probit model.

Figure 11.6a displays, again, the single O–D network example presented

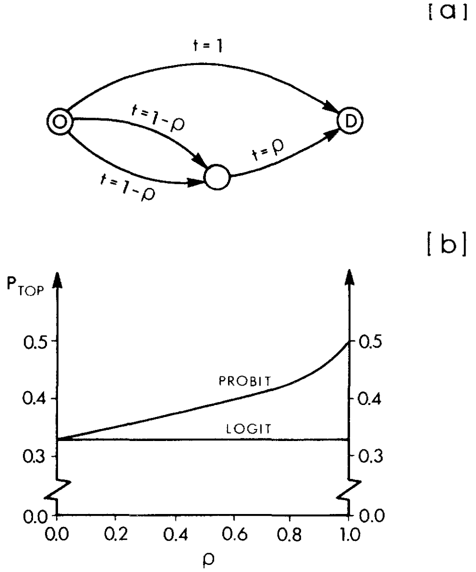


Figure 11.6 Comparison between logit and probit network loading models: (a) an example network, including an overlapping segment between the bottom routes; (b) the share of flow on the top route as a function of the amount of overlap.

in Section 11.1 (see Figure 11.4 and the related discussion). This network includes three paths, with the bottom two overlapping. The measured travel time on each of these paths is 1 time unit and the travel time along the bottom overlapping segment is ρ time units. Figure 11.6b displays the choice probability for the top path in Figure 11.6a, P_{TOP}, as a function of ρ.

According to the logit model, $P_{\text{TOP}} = \frac{1}{3}$ regardless of the value of ρ, as illustrated by the horizontal line shown in Figure 11.6b. The other curve in the figure depicts the probit choice probability of the top path, for various values of ρ, assuming that $\beta = 1.0$. This curve exhibits the expected shape, with $P_{\text{TOP}} = \frac{1}{3}$ for $\rho = 0$ and P_{TOP} increasing as ρ increases until, at $\rho = 1.0$, $P_{\text{TOP}} = \frac{1}{2}$. As explained in Section 11.1, this assignment is reasonable since, when $\rho \to 0$, the network includes three paths of the same length, whereas $\rho \to 1$ means that the network includes only two such paths.

The probit curve shown in Figure 11.6b could be obtained by using Monte Carlo simulation. This example, however, is small enough so that the analytical approximation method can be used. The reader can follow the formulas in the Appendix to verify that the choice probability of the top path can be expressed as

$$P_{\text{TOP}} = 1 - \Phi\left(\sqrt{\frac{1-\rho}{2\pi + \rho - 1}}\right) \qquad [11.19]$$

where $\Phi(\cdot)$ is the standard normal cumulative function. This is the probit curve plotted in Figure 11.6b.

The second difficulty associated with logit is that it cannot account for the dependence between the perception variance and the measured path travel times. As noted in Section 11.1, this means that a choice between a 5-minute-long path and a 10-minute-long path would result in the same flow distribution as a choice between 120-minute and 125-minute long paths. Figure 11.5 shows the two route-choice situations under consideration. This difficulty would not occur with a probit route-choice model, as shown in the following numerical example.

Consider, first, the choice between the shorter routes (i.e., the 5-minute route and the 10-minute one). According to the underlying probit assumption, the perceived travel time on the first route, C_1, is normally distributed with mean 5 and variance 5β, while the perceived travel time on the second route, $C_2 \sim N(10, 10\beta)$. Since these paths do not overlap, the choice probability of the first route, P_1,is given by

$$P_1 = \Pr(C_1 < C_2) = \Pr(C_1 - C_2 < 0) \qquad [11.20a]$$

The random variable $(C_1 - C_2)$ is normally distributed with mean -5 and variance $5\beta + 10\beta = 15\beta$. The probability that this random variable assumes a value less than zero can be calculated by evaluating the cumulative standard normal function at zero, that is,

$$P_1 = \Phi\left(\frac{0-(-5)}{\sqrt{15\beta}}\right) \qquad [11.20b]$$

Assuming that $\beta = 1$, this expression means that $P_1 = 0.90$ and $P_2 = 0.10$. In other words, 90% of the motorists would prefer the 5-minute route to the 10-minute route.

Consider now the choice between the 120-minute route and the 125-minute route. The respective perceived travel times on these routes are normally distributed [i.e., $C_1 \sim N(120, 120\beta)$ and $C_2 \sim N(125, 125\beta)$]. The probability that route 1 will be chosen is given by Eq. [11.20a]. The perceived travel-time difference, in this case, is also normally distributed with mean -5, but the variance is $120\beta + 125\beta = 245\beta$. Assuming that $\beta = 1$, the probability of choosing route 1 is then

$$P_1 = \Phi\left(\frac{5}{\sqrt{245\beta}}\right) = 0.63 \qquad [11.21]$$

meaning that $P_2 = 0.37$. As expected, more motorists choose the route with the higher mean travel time in this case. Thus the probit model results in the expected route flows.

Note that if, in the last example, the two routes overlapped for most of their length, the flow allocation should be similar to the flow allocation between the two shorter routes. To see this, assume that the two long routes (i.e., 120 minutes and 125 minutes) overlap and the travel time on the overlapping segment is 115 minutes. Such a situation is illustrated in Figure 11.7. The covariance between the perceived travel time of the two routes is proportional to their common portion, as indicated by Eq. [11.14c]. Thus

$$\text{cov}\,(C_1, C_2) = 115\beta \qquad [11.22a]$$

The variance of the difference between the perceived travel times of the two

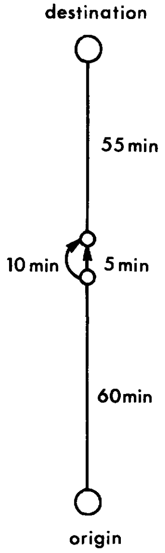


Figure 11.7 Example of two-route network in which the two routes overlap along most of their length.

routes equals the sum of the travel time variances minus twice the travel time covariance between the two routes, that is,

$$\text{var}\,(C_1 - C_2) = 120\beta + 125\beta - 2 \cdot 115\beta = 15\beta \qquad [11.22b]$$

Assuming that $\beta = 1$, the choice probability of route 1 is given in this case by

$$P_1 = \Phi\left(\frac{5}{\sqrt{15\beta}}\right) = 0.90 \qquad [11.22c]$$

This probability equals the one obtained in the case of the two short routes. This means that in the case of thc overlapping routes, the choice is made on the basis of the segments that are different between the two alternatives. The common segment does not affect the choice, as might be expected.

In conclusion, it seems that the flow assignment generated by the probit model is more reasonable than the one generated by a logit model. This realism is achieved, however, at higher computational costs. The probit assignment algorithm described in this section may require several (four to six) iterations for a large network. Each such iteration involves a full all-or-nothing assignment. The computational requirements of the STOCH algorithm, in comparison, are less than those required for two all-or-nothing assignments.

11.3 SUMMARY

This chapter outlined two stochastic network loading models which are both based on discrete choice models: the first on the multinomial logit and the second on the multinomial probit. Each network loading model is explained in detail, including an assignment algorithm corresponding to that model.

The first section of this chapter presented the logit model for stochastic network loading and the STOCH algorithm. This algorithm assigns the flow, using the logit model, to a subset of the paths connecting each O–D pair. Two definitions of this subset are presented, corresponding to two versions of the algorithm. The more efficient of these approaches is the single-pass version, in which each path is included in the subset of used paths only if none of the links along this path takes the motorist back toward the origin. This algorithm requires a computational effort that is only moderately higher than that required for an all-or-nothing assignment.

Notwithstanding the computational efficiency of the STOCH algorithm, the logit approach for stochastic network loading is limited in some respects. As shown in Section 11.1, the logit model assigns too much flow to overlapping paths. Also, it cannot account for a perception variance that depends on the length of the paths to be traveled.

Section 11.2 presents an application of the probit model to stochastic network loading. This model can be developed from the assumption that the perceived travel time on each link is normally distributed. This assumption leads to a multivariate normal distribution of the perceived travel times on all paths connecting each O–D pair.

To load the network according to probit choice probabilities, a Monte Carlo simulation can be used. This simulation consists of repeated iterations of random drawings of perceived link travel times and corresponding all-or-nothing assignments. The flow generated by this algorithm will stabilize for large transportation networks, after several iterations.

Probit network loading requires a larger computational effort than logit loading. As shown in Section 11.2, however, it alleviates some of the problems associated with logit-based loading models.

11.4 ANNOTATED REFERENCES

The STOCH algorithm for applying the logit model to transportation networks was suggested by Dial (1971). Subsequently, the algorithm was incorporated in the UMTA (1976) Transportation Planning System, which is a battery of computer programs supported by the Urban Mass Transportation Administration of the U.S. Department of Transportation.

The logit approach to stochastic network loading has since been criticized by many authors based on the arguments given in Section 11.2. These authors include Mayberry (1970), Schneider (1973), Burrell (1976), Florian and Fox (1976), and Daganzo and Sheffi (1977).

The application of the probit model to stochastic network loading problems was suggested by Daganzo and Sheffi (1977), who showed that this model does not suffer from logit's drawbacks. Sheffi and Powell (1981) operationalized and tested the Monte Carlo simulation algorithm for probit network loading.

Several other stochastic loading models have been suggested in the literature. Von Falkenhausen (1966) suggested an assignment based on log-normal distribution of link times, while Burrell (1968) suggested one based on a discrete-uniform distribution. Both of these methods use Monte Carlo simulation. Daganzo and Sheffi (1977) evaluate these approaches and compare them to probit-based assignment. Other authors, such as Gunnarson (1972) and Tobin (1977), developed variants of the STOCH algorithm which were intended to alleviate some of the drawbacks of the logit model mentioned in Section 11.1. The resulting models, which are not strictly logit-based, are reviewed and evaluated by Sheffi (1979).

11.5 PROBLEMS

11.1. Using the algorithm description in Section 11.1, develop a detailed flowchart of the single-pass STOCH algorithm. Discuss the list-processing techniques that should be considered in coding this algorithm.

11.2. Write a computer program that accepts a network coded in forward star form and a set of O–D trip rates, and performs a (single-pass) STOCH assignment. The output of the program is the set of link flows.

11.3. Show that the STOCH algorithm generates a logit assignment by following explicitly the steps of the algorithm. (In other words, show that Eq. [11.7] holds.)

11.4. Apply the single-stage STOCH algorithm to the network in Figure 11.1a and find the resulting flow pattern. Assume that the link travel times are as shown in Figure 11.1a but the travel time on links 4→5 and 5→6 is 3 rather than 1. Assume further that the trip rate from node 1 to 9 is 1000 trips per hour and the trip rate from node 1 to 6 is 500 trips per hour.

11.5. Develop a formula for calculating $\sigma_a^{(l)}$ from $\sigma_a^{(l-1)}$, $x_a^{(l)}$, and $X_a^{(l)}$, for probit-based network loading. With this formula the variance of the estimated link flows can be calculated without storing the entire sequence of link flows (see the algorithm's description following Eq. [11.18b]).

11.6. Write a detailed flowchart of the simulation-based probit loading model described in Section 11.2. (You may want to solve Problem 11.5 first.) Following this flowchart, write a computer program to perform probit network loading.

11.7. Based on the description of Clark's approximation in the Appendix, develop an expression for the flow on link 2→3 for the network shown in Figure P11.1. Assume that the O–D flow is q between nodes 1 and 4 and all other O–D flows are zero. The link flow should be expressed in terms of the link travel times, the O–D flow, and the probit parameter, β, assuming that $T_a \sim N(t_a, \beta t_a)$.

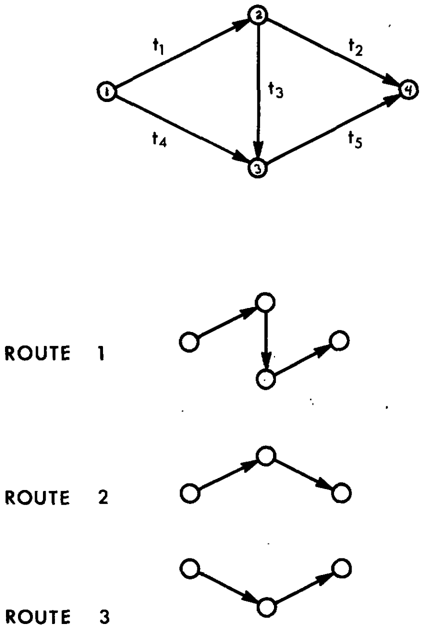


Figure P11.1

11.8. Consider a simple network including one O–D pair connected by two links. The perceived travel time on link 1 is uniformly distributed between 3 and 6. The perceived travel time on link 2 is uniformly distributed between 5 and 7 (and is independent of the perceived travel time on link 1). The O–D volume is 1 unit of flow.

(a) Calculate, analytically, the flow on each route.

(b) Perform a network loading using 10 iterations of Monte Carlo simulation. Compare your answer to the results of (a). Use the following sequence of random numbers: 0.923, 0.425, 0.568, 0.354, 0.320, 0.115, 0.863, 0.744, 0.147, 0.107, 0.587, 0.427, 0.837, 0.803, 0.967, 0.572, 0.225, 0.817, 0.034, 0.289. Use the first random number for link 1, the second for link 2, the third for link 1, the fourth for link 2, and so on.

11.9. How can variance-reduction techniques, which are used in many Monte Carlo simulations, be utilized to expedite the convergence of a probit-based network loading code? (*Note:* Variance reduction techniques are explained in almost every text dealing with Monte Carlo simulation.)

11.10. Suggest a Monte Carlo simulation-based model for conducting a logit assignment. What are the limitations of this approach? What type of distributions of perceived travel time are amenable to a simulation-based algorithm?

12 Stochastic User Equilibrium

The focus of the last two chapters was on the formulation and the solution of stochastic network loading models. These models assign a set of O–D trip rates to a transportation network in which the link travel times are fixed (i.e., they are not flow dependent). The assignment assumes that route choice is based on perceived rather than measured link travel times. The travel times perceived by motorists are assumed to be random variables, the distribution of which (across the population of motorists) is known. The various stochastic network loading models discussed in Chapter 11 can each be derived from a certain assumption regarding the distribution of these random variables (the perceived travel times).

This chapter introduces the equilibrium dimension into this problem. Here, the perceived travel times are modeled not only as random variables, but also as flow dependent. This dependence is accounted for by assuming that the mean travel time for each link is a function of the flow on that link. In other words, as in Parts II and III, $t_a = t_a(x_a)$, notwithstanding the fact that $t_a = E[T_a]$, where T_a is the perceived travel time on link a.

Given the O–D trip rates, $\{q_{rs}\}$, the stochastic equilibrium conditions can be characterized by the following equations:

$$f_k^{rs} = q_{rs} P_k^{rs} \qquad \forall\ k, r, s \qquad [12.1]$$

where P_k^{rs} is the probability that route k between r and s is chosen given a set of measured travel times, t. In other words, $P_k^{rs} = P_k^{rs}(\mathbf{t}) = \Pr\ (C_k^{rs} \leq C_l^{rs},\ \forall\ l \neq k \in \mathscr{K}_{rs} \mid \mathbf{t})$, where C_k^{rs} is the random variable representing the perceived travel time on route k between r and s, that is, $C_k^{rs} = \sum_a T_a \delta_{a,k}^{rs}$, $\forall\ k, r, s$. In addition, the regular network constraints have to hold, that is,

$$t_a = t_a(x_a) \qquad \forall\ a \tag{12.2a}$$

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\ r, s \tag{12.2b}$$

where $t_a = E[T_a]$ and $x_a = \sum_{rs} \sum_k f_k^{rs} \delta_{a,k}^{rs}$, $\forall\ a$. Note that the choice probability is conditional on the values of the mean link travel times at equilibrium. If these times are known (or fixed), the flow pattern that solves Eqs. [12.1] and [12.2b] can be determined by a simple stochastic network loading. In this chapter, however, these times are assumed to be flow dependent, as indicated by Eq. [12.2a]. Note also that Eq. [12.2b] is automatically satisfied if the path flows are given by Eq. [12.1] since $\sum_k f_k^{rs} = \sum_k q_{rs} P_k^{rs} = q_{rs} \sum_k P_k^{rs} = q_{rs}$, $\forall\ r, s$.

Equation [12.1] characterizes the *stochastic user-equilibrium* (SUE) condition defined in Section 1.3. At SUE, no motorist can improve his or her *perceived* travel time by unilaterally changing routes. This follows directly from the interpretation of the choice probability as the probability that the perceived travel time on the chosen route is the smallest of all the routes connecting the O–D under consideration. At equilibrium (SUE) the measured travel time on all used paths is *not* going to be equal. Instead, the travel time will be such that Eq. [12.1] is satisfied for the equilibrium path flows. These path flows, in turn, will be associated with link flows that satisfy Eqs. [12.2] for the equilibrium travel times.

As mentioned in Chapter 1, stochastic user equilibrium (SUE) is a more general statement of equilibrium than the UE conditions. In other words, the UE conditions are a particular case of SUE; when the variance of travel time perception is zero, the SUE conditions are identical to the UE conditions. This is shown below.

The first step is to write the route choice probability, P_k^{rs}, in a form that is applicable to both continuous and discrete perceived travel time distributions. The route choice probability is then defined as

$$\Pr\,[C_k^{rs} < C_l^{rs},\ \forall\ l \neq k \mid \mathbf{t}] \leq P_k^{rs} \leq \Pr\,[C_k^{rs} \leq C_l^{rs},\ \forall\ l \neq k \mid \mathbf{t}] \qquad \forall\ k, r, s \tag{12.3}$$

Using this definition, the SUE condition can be written as follows:

$$\Pr\,[C_k^{rs} < C_l^{rs},\ \forall\ l \neq k \mid \mathbf{t}] \leq \frac{f_k^{rs}}{q_{rs}} \leq \Pr\,[C_k^{rs} \leq C_l^{rs},\ \forall\ l \neq k \mid \mathbf{t}] \qquad \forall\ k, r, s \tag{12.4}$$

This definition of SUE is applicable both to cases in which the perceived route travel times, $\{C_k^{rs}\}$, are continuous variables and to cases in which these variables are discrete. (It is important to note that only in the discrete case is there a nonzero probability for two or more routes to be tied for the shortest.) Equation [12.4], then, is a generalization of Eq. [12.1]. In cases where the path travel times are continuous random variables, the probability that the perceived travel times on any two paths are equal is zero, and the expression on the left-hand side of Eq. [12.4] is equal to the expression on the right-hand side of this equation. In such cases, both sides of Eq. [12.4] hold as an equality and this equation is identical to Eq. [12.1].

To show that the UE conditions are a particular case of SUE, one has to examine the case of deterministic travel times (which is obtained when the perception variance is assumed to be zero). In this case, the perceived travel times equal the measured travel times ($\mathbf{C} = \mathbf{c}$). Also, since the focus of the discussion is on UE, the travel time should be modeled as a discrete variable (recall that the possibility of equal travel times is central to the UE definition). If the path travel times are discrete deterministic variables, two or more paths can be tied for the shortest, in which case neither side of Eq. [12.4] can hold as an equality and be used to characterize the equilibrium flow pattern. This statement is proven below by contradiction.

Consider a case in which two paths between origin r and destination s are tied for the shortest. In this case,

$$\Pr\,[c_k^{rs} < c_l^{rs},\ \forall\ l \neq k \,|\, \mathbf{t}] = 0 \qquad \text{for all paths } k \qquad [12.5a]$$

If the left-hand side of Eq. [12.4] held as an equality, Eq. [12.5a] would have implied that $f_k^{rs} = 0\ \forall\ k$ (that is, the flow on all paths is zero). This means that condition [12.2b] is violated. Also, under the same circumstances (i.e., when two paths are tied for the shortest),

$$\Pr\,[c_k^{rs} \leq c_l^{rs},\ \forall\ l \neq k \,|\, \mathbf{t}] = \begin{cases} 1 & \text{for the two shortest paths} \\ 0 & \text{for all other paths} \end{cases} \qquad [12.5b]$$

If the right-hand side of Eq. [12.4] held as an equality, the flow on each one of the shortest paths between r and s at equilibrium would be equal to q_{rs}. This means that too much flow was assigned to the network and again, condition [12.2b] would have been violated. Consequently if the perceived path travel times are discrete and deterministic variables, $f_k^{rs}/q_r \neq \Pr\,[C_k^{rs} < C_l^{rs},\ \forall\ l \neq k \,|\, \mathbf{t}]$ and $f_k^{rs}/q_{rs} \neq \Pr\,[C_k^{rs} \leq C_l^{rs},\ \forall\ l \neq k \,|\, \mathbf{t}]$.

To interpret equilibrium condition [12.4] in the deterministic (and discrete) case (i.e., when $\mathbf{C} = \mathbf{c}$), note first that the probability statements on both the right- and left-hand sides of this expression can be either 1 or zero. This is because the event to which they are assigned can be either true or false when all path times are deterministic. The flow on a particular path can therefore be either zero or positive. Two cases can then be distinguished:

1. If $f_k^{rs} > 0$, it must be that the probability on the right-hand side of Eq. [12.4] is equal to one. In other words, path k between r and s must be the shortest path (or one of a set of shortest paths).
2. If $f_k^{rs} = 0$, it must be that the probability on the left-hand side of Eq. [12.4] equals zero. In other words, path k cannot be the shortest path, and there is at least one path shorter than (or as short as) k.

These conditions can be summarized as follows for every path k, connecting O–D pair r and s:

$$f_k^{rs} > 0 \quad \text{implies} \quad c_k^{rs} \leq c_l^{rs} \qquad \forall\, l \neq k \tag{12.6a}$$

$$f_k^{rs} = 0 \quad \text{implies} \quad c_k^{rs} \geq c_l^{rs} \qquad \text{for at least one } l \neq k \tag{12.6b}$$

Equations [12.6] define a subset of paths with equal travel times, the flow on which is positive, and state that the flow is zero on all other paths. The travel time on all the paths carrying no flow is greater than or equal to the travel time on the paths that do carry flow. This is equivalent to the definition of the UE criterion. Consequently, it can be concluded that Eqs. [12.6] state the UE criterion, meaning that the SUE conditions [12.4] can be regarded as a generalization of the UE conditions. The latter is obtained from the SUE conditions if the variances of the perceived path travel times are assumed to be zero.

Two comments are in order here. First, note that since the formulation of the SUE conditions given in Eq. [12.1] holds for cases in which the perceived path travel times are assumed to be continuous variables, it holds for the logit and probit models discussed in Chapter 11. Second, note that link flows in the SUE model are random variables, as explained in Section 10.2 in conjunction with stochastic network loading models. The variables $\{x_a\}$ and $\{f_k^{rs}\}$ used in the definition of the SUE conditions are the means of these random variables.

Given the definition of stochastic user equilibrium, this chapter discusses three topics separately. First, Section 12.1 formulates and investigates a mathematical program, the solution of which is the SUE flow pattern. Section 12.2 then describes an algorithmic solution approach to this problem, and finally, Section 12.3 discusses the convergence of this algorithm and extends the SUE concept in several directions.

12.1 SUE EQUIVALENT MINIMIZATION

As was the case with all the equilibrium problems discussed in this text, a direction solution of the stochastic equilibrium equations is difficult. The approach used, instead, is to formulate a minimization program, the solution of which is the desired set of equilibrium flows. This section shows the equivalence of the solution of the minimization program and the SUE conditions by checking the first-order conditions of the program. Furthermore, the problem is shown to have a unique minimum (even though it is not convex).

Program Formulation

Consider the following minimization problem:

$$\min_{\mathbf{x}} z(\mathbf{x}) = -\sum_{rs} q_{rs} E\left[\min_{k \in \mathscr{K}_{rs}} \{C_k^{rs}\} \mid \mathbf{c}^{rs}(\mathbf{x})\right] + \sum_a x_a t_a(x_a) - \sum_a \int_0^{x_a} t_a(\omega)\, d\omega \tag{12.7}$$

The flow pattern that minimizes Eq. [12.7] also satisfies the SUE conditions. Furthermore, at the solution point, the network constraints are satisfied and consequently, Eq. [12.7] can be minimized as an *unconstrained* program.

Before these statements regarding Eq. [12.7] are proved, note that this objective function includes, in its first term, the expected perceived travel time function discussed in Section 10.2 (see Eq. [10.29] and the related discussion). In other words, Eq. [12.7] can be rewritten as

$$\min_{\mathbf{x}} z(\mathbf{x}) = -\sum_{rs} q_{rs} S_{rs}[\mathbf{c}^{rs}(\mathbf{x})] + \sum_a x_a t_a(x_a) - \sum_a \int_0^{x_a} t_a(\omega)\, d\omega \qquad [12.8]$$

where

$$S_{rs}[\mathbf{c}^{rs}(\mathbf{x})] = E\left[\min_{k \in \mathcal{K}_{rs}} \{C_k^{rs}\} \mid \mathbf{c}^{rs}(\mathbf{x}) \right] \qquad [12.9]$$

The conditioning of the random variable C_k^{rs} on $\mathbf{c}^{rs}(\mathbf{x})$ in Eq. [12.9], implies that the expectation is taken at a given flow level, $\mathbf{x}$. Two of the properties of the expected perceived travel time function are important for the analysis that follows. First, this function is *concave* with respect to $\mathbf{c}^{rs}$, and second (see Eq. [10.31a])†

$$\frac{\partial S_{rs}(\mathbf{c}^{rs})}{\partial c_k^{rs}} = P_k^{rs} \qquad [12.10]$$

where P_k^{rs} is the probability of choosing path k between r and s. These properties are used below to investigate the first-order conditions of program [12.7]. The validity of this formulation is first illustrated, however, for a small example.

Consider the simple network shown in Figure 12.1. This network includes one O–D pair connected by two routes, the measured travel time on which is given by

$$c_1 = 1 + 2f_1 \quad \text{time units} \qquad [12.11a]$$

$$c_2 = 2 + f_2 \quad \text{time units} \qquad [12.11b]$$

The total O–D flow is $q = 4$ (i.e., $f_1 + f_2 = 4$ flow units). The stochastic network loading is assumed to be a logit route-choice model with parameter $\theta = 1$. In other words, at equilibrium,

$$\frac{f_1}{4} = P_1 = \frac{e^{-c_1}}{e^{-c_1} + e^{-c_2}} = \frac{1}{1 + e^{c_1 - c_2}} \qquad [12.11c]$$

$$f_2 = 4 - f_1 \quad \text{flow units} \qquad [12.11d]$$

†For probit route-choice models this property holds only if the variance of the perceived path travel time is not a function of the mean path travel time. This implies that the probit route-choice model cannot be based on the assumption that $T_a \sim N(t_a, \beta t_a)$. Instead, the variance of the perceived link travel time can be defined, for example, as βt_a^0, where t_a^0 is the free flow travel time on link a, or as βl_a, where l_a is the link length. As argued in Chapter 11 (see Eq. [11.12] and the related discussion) such a definition may also constitute a more realistic model of travel time perception.

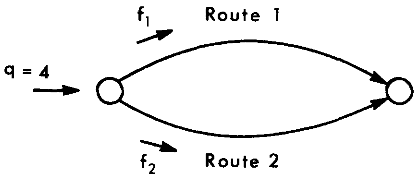


Figure 12.1 Network example with two paths connecting a single O–D pair.

where P_1 is the probability of choosing the first route. The equilibrium flows and travel times, in this example, can be found by solving Eqs. [12.11] simultaneously. Substituting Eqs. [12.11a] and [12.11b] into [12.11c], and using the total flow constraint [12.11d], Eq. [12.11c] can be expressed as

$$\frac{f_1}{4} = \frac{1}{1 + e^{(3f_1 - 5)}} \qquad [12.12]$$

The solution of this equation can be found, numerically, to be $f_1 = 1.75$† flow units, meaning that $f_2 = 2.25$ flow units, $c_1 = 4.50$ time units, and $c_2 = 4.25$ time units. Substituting these path times back into Eq. [12.11c], the result is $P_1 = 0.438$ and $f_1 = 1.75$ flow units, reaffirming that this is the equilibrium solution. Note, however, that $c_1 \neq c_2$ at equilibrium; the network is in SUE, not UE.

The same example can now be solved by using program [12.8]. The equivalent program for this example can be written as follows:

$$\min z(f_1, f_2) = -4S[c_1(f_1), c_2(f_2)] + f_1(1 + 2f_1) + f_2(2 + f_2)$$
$$- \int_0^{f_1} (1 + 2\omega)\, d\omega - \int_0^{f_2} (2 + \omega)\, d\omega \qquad [12.13a]$$

Carrying out the integration, this program becomes

$$\min z(f_1, f_2) = -4S[c_1(f_1), c_2(f_2)] + f_1^2 + \tfrac{1}{2}f_2^2 \qquad [12.13b]$$

Equation [12.13b] has to be minimized in terms of both flow variables, since the flow conservation constraint is not included in the problem (and thus it cannot be substituted in). Both derivatives of $z(f_1, f_2)$ have to vanish at the minimum. The derivative of $z(f_1, f_2)$ with respect to f_1 is

$$\frac{\partial z(f_1, f_2)}{\partial f_1} = -4\, \frac{\partial S[c_1(f_1), c_2(f_2)]}{\partial c_1} \frac{dc_1(f_1)}{df_1} + 2f_1$$

$S[c_1(f_1), c_2(f_2)]$ is the expected perceived travel time function, however, and thus $\partial S[\cdot\,, \cdot]/\partial c_1 = P_1$ (where P_1 is given by Eq. [12.11c]). Furthermore, from Eq. [12.11a], $dc_1/df_1 = 2$. The first-order condition with respect to f_1 is, then,

$$\frac{\partial z(\mathbf{f})}{\partial f_1} = -4P_1 \cdot 2 + 2f_1 = -8P_1 + 2f_1 = 0 \qquad [12.14a]$$

†More precisely, $f_1 = 1.750327$.

Similarly, the first-order condition with respect to f_2 can be written as

$$\frac{\partial z(\mathbf{f})}{\partial f_2} = -4P_2 \cdot 1 + f_2 = -4P_2 + f_2 = 0 \qquad [12.14b]$$

To solve Eqs. [12.14], note first that if Eq. [12.14b] is multiplied by 2 and these equations are added, the result is

$$-8P_1 + 2f_1 - 8P_2 + 2f_2 = 0$$

Since $P_1 + P_2 = 1$, this means that

$$f_1 + f_2 = 4 \qquad [12.15]$$

Equation [12.15] states that the flow conservation condition has to hold at the point that minimizes $z(f_1, f_2)$. This shows that, at least in the case of this example, program [12.13a] can be solved as an unconstrained minimization and still yield a solution that satisfies the flow conservation constraint.

Once the flow conservation constraint has been established, it can be substituted into Eq. [12.14a] (or [12.14b]) and the resulting equation can be solved for f_1. First, however, Eq. [12.14a] has to be expressed explicitly in terms of the logit path choice probability. This expression, after the above-mentioned substitution, becomes

$$(-8)\frac{1}{1 + e^{c_1(f_1) - c_2(4 - f_1)}} + 2f_1 = 0$$

or

$$\frac{4}{1 + e^{(3f_1 - 5)}} = f_1 \qquad [12.16]$$

This equation is identical to Eq. [12.12], meaning that its solution is $f_1 = 1.75$ time units and consequently $f_2 = 2.25$ time units. The solution obtained by minimizing the SUE objective function is thus identical to the solution obtained by solving the equilibrium conditions directly. Furthermore, the SUE objective function was minimized as an unconstrained program.

The SUE objective function for this example is illustrated in Figure 12.2. This figure depicts the contours of $z(f_1, f_2)$ in the two-dimensional space defined by f_1 and f_2. As can be seen from the figure, the function is "well behaved" with a unique minimum at $z(1.75, 2.25) = -9.10$. The diagonal line shown in the figure is the feasible region for the SUE equilibrium problem (not for the program, which is unconstrained). In other words, it is the line along which $f_1 + f_2 = 4$. As the figure shows, the minimum of the SUE objective function satisfies the flow conservation constraint, at least for this example. These characteristics of the minimum of the SUE equivalent minimization are proved below for general networks.

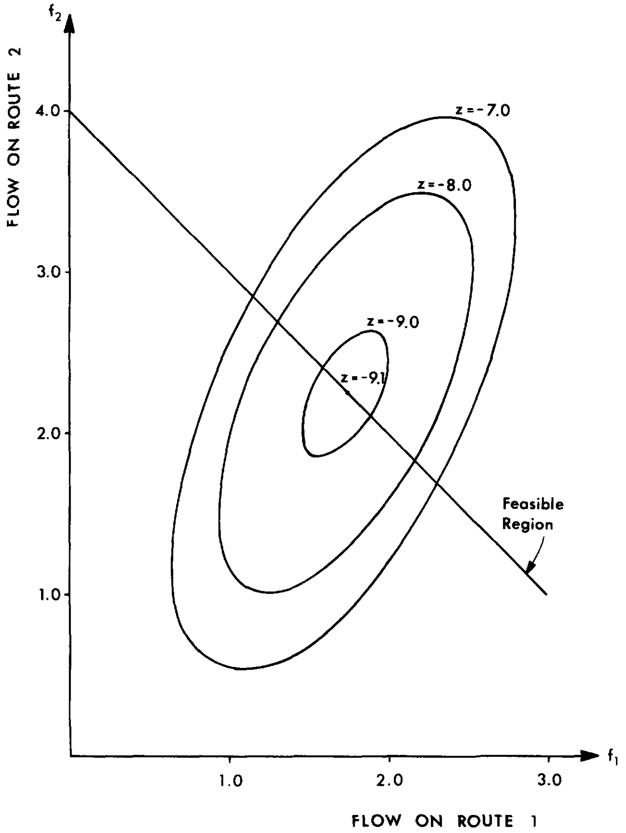


Figure 12.2 Contour lines of the SUE objective function for the network example in Figure 12.1 (Eq. [12.13] with $S[c(f)] = -\ln [e^{-c_1(x_1)} + e^{-c_2(x_2)}]$). The line shows the locus of points meeting the flow conservation condition. Note that the minimum of the objective function is on the line.

Equivalence Conditions

To prove the equivalence between the solution of the program given in Eq. [12.8] and the stochastic user equilibrium equations, the first-order conditions of this program have to coincide with the SUE conditions (Eqs. [12.1]). Furthermore, since the aforementioned program is unconstrained, it should be established that the flow conservation constraints (Eqs. [12.2b]) hold at the minimum of this program.

The first-order conditions for unconstrained minimizations require only that the gradient vanishes at the minimum point. In other words,

$$\nabla z(\mathbf{x}) = \mathbf{0} \qquad [12.17]$$

The gradient is taken with respect to **x**, the vector of link flows. This gradient is derived below by focusing on a typical term, $\partial z(\mathbf{x})/\partial x_b$, that is, the partial derivative of $z(\mathbf{x})$ with respect to the flow on link b.

Program [12.8] includes three separate summation terms. The derivative of the first term with respect to x_b can be calculated as follows:

$$\frac{\partial}{\partial x_b}\left\{-\sum_{rs} q_{rs} S_{rs}[\mathbf{c}^{rs}(\mathbf{x})]\right\} = -\sum_{rs} q_{rs} \sum_k \frac{\partial S_{rs}(\mathbf{c}^{rs})}{\partial c_k^{rs}} \frac{\partial c_k^{rs}(\mathbf{x})}{\partial x_b} \qquad [12.18]$$

In this expression,

$$\frac{\partial S_{rs}(\mathbf{c}^{rs})}{\partial c_k^{rs}} = P_k^{rs} \qquad [12.19a]$$

following Eq. [12.10]. Furthermore,

$$\frac{\partial c_k^{rs}(\mathbf{x})}{\partial x_b} = \frac{\partial}{\partial x_b}\left[\sum_a t_a(x_a)\delta_{a,k}^{rs}\right] = \frac{dt_b(x_b)}{dx_b}\,\delta_{b,k}^{rs} \qquad [12.19b]$$

Using Eqs. [12.19], Eq. [12.18] can be rewritten as

$$\frac{\partial}{\partial x_b}\left[-\sum_{rs} q_{rs} S_{rs}(\,\cdot\,)\right] = -\sum_{rs} q_{rs} \sum_k P_k^{rs} \frac{dt_b}{dx_b}\,\delta_{b,k}^{rs} \qquad [12.20]$$

where dt_b/dx_b is used to abbreviate $dt_b(x_b)/dx_b$.

The second term in the objective function is the total measured travel time. The derivative of this term with respect to x_b is given by

$$\frac{\partial}{\partial x_b}\left[\sum_a x_a t_a(x_a)\right] = t_b + x_b \frac{dt_b}{dx_b} \qquad [12.21]$$

The derivative of the last term in the objective function is

$$\frac{\partial}{\partial x_b}\left[-\sum_a \int_0^{x_a} t_a(\omega)\, d\omega\right] = -t_b \qquad [12.22]$$

The results of Eqs. [12.20] through [12.22] can be combined to express the derivative of the SUE objective function with respect to a link-flow variable. This derivative is

$$\frac{\partial z(\mathbf{x})}{\partial x_b} = \left[-\sum_{rs}\sum_k q_{rs} P_k^{rs}\delta_{b,k}^{rs} + x_b\right]\frac{dt_b}{dx_b} \qquad \forall\, b \qquad [12.23]$$

(Note that the term "t_b" is canceled between Eqs. [12.21] and [12.22].) Assuming that the link performance functions are strictly increasing (i.e., $dt_b/dx_b > 0$), this gradient can vanish only if

$$x_b = \sum_{rs}\sum_k q_{rs} P_k^{rs}\delta_{b,k}^{rs} \qquad \forall\, b \qquad [12.24]$$

Equation [12.24] expresses the equilibrium link flows corresponding to the equilibrium path flows given in Eq. [12.1]. This can be verified by multiplying both sides of Eq. [12.1] by $\delta_{b,k}^{rs}$ and summing over all paths k connecting all O–D pairs r–s.

The gradient vector of $z(\mathbf{x})$ can be written explicitly, based on Eq. [12.23], as

$$\nabla z(\mathbf{x}) = \left[-\sum_{rs} q_{rs} \mathbf{P}^{rs} \mathbf{\Delta}^{rs^T} + \mathbf{x} \right] \cdot \nabla_{\mathbf{x}} \mathbf{t} \qquad [12.25]$$

where $\mathbf{P}^{rs} = (\ldots, P_k^{rs}, \ldots)$ is the vector of path choice probabilities for O–D pair r–s, $\mathbf{\Delta}^{rs}$ is the link–path incidence matrix for this O–D pair, and $\nabla_{\mathbf{x}} \mathbf{t}$ is the Jacobian of link travel times. The Jacobian matrix includes the derivative of each link performance function with respect to each link flow (see Eq. [8.25]). At this point it is assumed that there are no link interactions, and thus $\nabla_{\mathbf{x}} \mathbf{t}$ is a diagonal matrix with the terms $dt_a(x_a)/dx_a$ along its diagonal. Equation [12.23] gives the term in the bth position of the gradient vector.

The gradient of the SUE program can also be taken directly with respect to the path flows (see Problem 12.2). In this case a typical term would be

$$\frac{\partial z[\mathbf{x}(\mathbf{f})]}{\partial f_l^{mn}} = \sum_a (-q_{mn} P_l^{mn} + f_l^{mn}) \frac{dt_a}{dx_a} \delta_{a,l}^{mn} \qquad \forall\ l, m, n \qquad [12.26]$$

At the point that minimizes the objective function, each of these derivatives has to equal zero, meaning that

$$f_l^{mn} = q_{mn} P_l^{mn} \qquad \forall\ l, m, n \qquad [12.27]$$

Both sides of Eq. [12.27] can be summed over all paths connecting O–D pair m–n, resulting in the condition

$$\sum_l f_l^{mn} = q_{mn} \qquad [12.28]$$

since $\sum_l P_l^{mn} = 1$. Thus the flow pattern that solves the SUE program also satisfies the SUE conditions and the flow conservation constraints. Consequently, the SUE flow pattern can be obtained by minimizing the SUE program.

Uniqueness Conditions†

The usefulness of the SUE equivalent minimization program would be limited if it did not have a unique minimum. To demonstrate uniqueness, it is sufficient to show that the Hessian matrix of the SUE objective function is positive definite, implying that the program is strictly convex. This matrix is calculated below by concentrating on a typical term, $\partial^2 z(\mathbf{x})/\partial x_b\, \partial x_a$.

†This section can be skipped without loss of continuity.

Using Eq. [12.23] as a starting point, the second derivative of $z(\mathbf{x})$ can be calculated as follows:

$$\frac{\partial z^2(\mathbf{x})}{\partial x_b \, \partial x_a} = \left[-\sum_{rs} \sum_k q_{rs} \sum_l \frac{\partial P_k^{rs}}{\partial c_l^{rs}} \frac{\partial c_l^{rs}}{\partial x_a} \delta_{b,k}^{rs} + \frac{\partial x_b}{\partial x_a} \right] \frac{dt_b}{dx_b} + \left[-\sum_{rs} \sum_k q_{rs} P_k^{rs} \delta_{b,k}^{rs} + x_b \right] \frac{d^2 t_b}{dx_b \, dx_a}$$

From the link-path incidence relationships note that

$$\frac{\partial c_l^{rs}}{\partial x_a} = \frac{dt_a}{dx_a} \delta_{a,l}^{rs} \qquad [12.29a]$$

$$\frac{\partial x_b}{\partial x_a} = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases} \qquad [12.29b]$$

Substituting the relationships in Eqs. [12.29] into the second derivative above, this expression can be written as

$$\frac{\partial z^2(\mathbf{x})}{\partial x_a \, \partial x_b} = \begin{cases} -\sum_{rs} q_{rs} \sum_k \sum_l \frac{\partial P_k^{rs}}{\partial c_l^{rs}} \left(\frac{dt_b}{dx_b} \delta_{b,l}^{rs} \right) \left(\frac{dt_b}{dx_b} \delta_{b,k}^{rs} \right) + \frac{dt_b}{dx_b} & \\ \quad + \left[-\sum_{rs} \sum_k q_{rs} P_k^{rs} \delta_{a,k}^{rs} + x_b \right] \frac{d^2 t_b}{dx_b^2} & \text{for } a = b \\ -\sum_{rs} q_{rs} \sum_k \sum_l \frac{\partial P_k^{rs}}{\partial c_l^{rs}} \left(\frac{dt_a}{dx_a} \delta_{a,l}^{rs} \right) \left(\frac{dt_b}{dx_b} \delta_{b,k}^{rs} \right) & \text{for } a \neq b \end{cases} \qquad [12.30]$$

Equation [12.30] gives the term in the bth row and the ath column of the Hessian matrix of $z(\mathbf{x})$. The Hessian, according to Eq. [12.30], can be expressed as the sum of three separate matrices (see Problem 12.3), as follows:

$$\nabla^2 z(\mathbf{x}) = \sum_{rs} q_{rs} [(\nabla_{\mathbf{x}} \mathbf{t} \cdot \boldsymbol{\Delta}^{rs}) \cdot (-\nabla_{\mathbf{c}} \mathbf{P}^{rs}) \cdot (\nabla_{\mathbf{x}} \mathbf{t} \cdot \boldsymbol{\Delta}^{rs})^T] + \nabla_{\mathbf{x}} \mathbf{t} + \nabla_{\mathbf{x}}^2 \mathbf{t} \cdot \mathbf{R} \qquad [12.31]$$

The Hessian is an $A \times A$ matrix, where A is the number of links in the network. The following matrix notations are used in Eq. [12.31]:

1. $\nabla_{\mathbf{x}} \mathbf{t}$ is the $A \times A$ Jacobian of the link-travel-time vector introduced in Chapter 8. In the case examined here, where no link interactions exist, it is a diagonal matrix, the ath element of which is dt_a/dx_a.
2. $\boldsymbol{\Delta}^{rs}$ is the link-path incidence matrix for O–D pair r–s introduced in the beginning of Chapter 3. It is an $A \times K_{rs}$ matrix (where K_{rs} is the number of paths between r and s), including $\delta_{a,k}^{rs}$ as a typical term.
3. $\nabla_{\mathbf{c}} \mathbf{P}^{rs}$ is the Jacobian of the choice probability vector $(\ldots, P_k^{rs}, \ldots)$ for

O–D pair r–s with respect to the path travel times. It is a $K_{rs} \times K_{rs}$ matrix with terms of the form $\partial P_k^{rs}/\partial c_l^{rs}$.

4. $\nabla_{\mathbf{x}}^2 \mathbf{t}$ is a diagonal $A \times A$ matrix which includes terms of the form $d^2 t_a / dx_a^2$ along its diagonal.
5. $\mathbf{R}$ is a diagonal $A \times A$ matrix, the ath element of which is

$$\left(-\sum_{rs} \sum_{k} q_{rs} P_k^{rs} \delta_{a,k}^{rs} + x_a\right)$$

The Hessian in Eq. [12.31] can now be analyzed component by component.

The first matrix on the right-hand side of Eq. [12.31] is the following:

$$\sum_{rs} q_{rs}[(\nabla_{\mathbf{x}} \mathbf{t} \cdot \mathbf{\Delta}^{rs}) \cdot (-\nabla_{\mathbf{c}} \mathbf{P}^{rs}) \cdot (\nabla_{\mathbf{x}} \mathbf{t} \cdot \mathbf{\Delta}^{rs})^T] \qquad [12.32]$$

To see the nature of this matrix, consider first the choice probability Jacobian, $\nabla_{\mathbf{c}} \mathbf{P}^{rs}$. This matrix is negative semidefinite since it is the Hessian of the expected perceived travel time, $S_{rs}(\mathbf{c}^{rs})$, which is concave (see the discussion following Eq. [10.30]) in $\mathbf{c}^{rs}$. Consequently, $(-\nabla_{\mathbf{c}} \mathbf{P}^{rs})$ is a positive-semidefinite matrix. Equation [12.32] includes then a quadratic form (of nonzero terms) applied to a positive-semidefinite matrix; it is therefore positive semidefinite (see Problem 12.3b).

The second matrix in the sum that comprises the Hessian (in Eq. [12.31]) is $\nabla_{\mathbf{x}} \mathbf{t}$. This is a diagonal matrix with positive entries (assuming that all link performance functions are strictly increasing). It is therefore positive definite. The sum of this matrix and the above-mentioned one is a positive-definite matrix (since it is the sum of a positive-semidefinite and a positive-definite matrix).

The third matrix in the sum on the right-hand side of Eq. [12.31] is the product of two separate matrices. The first one, $\nabla_{\mathbf{x}}^2 \mathbf{t}$, is a diagonal matrix with positive terms (assuming that the link performance functions are convex). The second matrix in the product, $\mathbf{R}$, includes terms such as $[-\sum_{rs} \sum_k q_{rs} P_k^{rs} \delta_{a,k}^{rs} + x_a]$ along its diagonal. These terms can be either positive or negative. Thus the last matrix is not, in general, positive definite. Note, however, that as the equilibrium point is approached $(x_a - \sum_{rs} \sum_k q_{rs} P_k^{rs} \delta_{a,k}^{rs}) \rightarrow 0$ and the last matrix vanishes. This means that at the point that satisfies the SUE equations, the SUE equivalent program is strictly convex. (The Hessian at this point is positive definite.) At all other points, however, the Hessian of $z(\mathbf{x})$ is the sum of a positive-definite matrix and an indefinite matrix (i.e., a matrix that can be either positive or negative definite) and is therefore indefinite. The only conclusion from this evidence is that the equilibrium point is, in fact, a local minimum. It is not possible to conclude that other local minima do not exist.

To prove that the SUE point is the only minimum of the SUE program, a simple transformation of the performance functions can be used. Let $x_a(t_a)$ denote the inverse of $t_a(x_a)$; this inverse exists since $t_a(x_a)$ is monotone. The inverse function, $x_a(t_a)$, is increasing for all positive t_a and positive for all

$t_a > t_a^{\min}$, where $t_a^{\min} = t_a(0)$. A link performance function and its inverse are shown in Figure 12.3a and b, respectively.

The SUE program can be posed as a function of $\mathbf{t}$ rather than of $\mathbf{x}$, by introducing the change of variables $x_a = x_a(t_a)$ into Eq. [12.7]. The objective function then becomes

$$z(\mathbf{t}) = -\sum_{rs} q_{rs} E\left[\min_k \{C_k^{rs} \mid \mathbf{c}^{rs}\}\right] + \sum_a x_a(t_a)t_a - \sum_a \int_{t_a^{\min}}^{t_a} v \frac{dx_a(v)}{dv}\, dv \tag*{[12.33]}$$

This expression, after integration by parts, reduces to

$$z(\mathbf{t}) = -\sum_{rs} q_{rs} E\left[\min_k \{C_k^{rs} \mid \mathbf{c}^{rs}\}\right] + \sum_a \int_{t_a^{\min}}^{t_a} x_a(v)\, dv \tag*{[12.34]}$$

The reader is asked (in Problem 12.4) to work the details of the change of variable leading to Eq. [12.33].

The gradient of $z(\mathbf{t})$ is given by

$$\nabla z(\mathbf{t}) = -\sum_{rs} q_{rs} \mathbf{P}^{rs} \boldsymbol{\Delta}^{rsT} + \mathbf{x} \tag*{[12.35a]}$$

A typical term of this gradient is

$$\frac{\partial z(\mathbf{t})}{\partial t_a} = -\sum_{rs} q_{rs} \sum_k P_k^{rs} \delta_{a,k}^{rs} + x_a \tag*{[12.35b]}$$

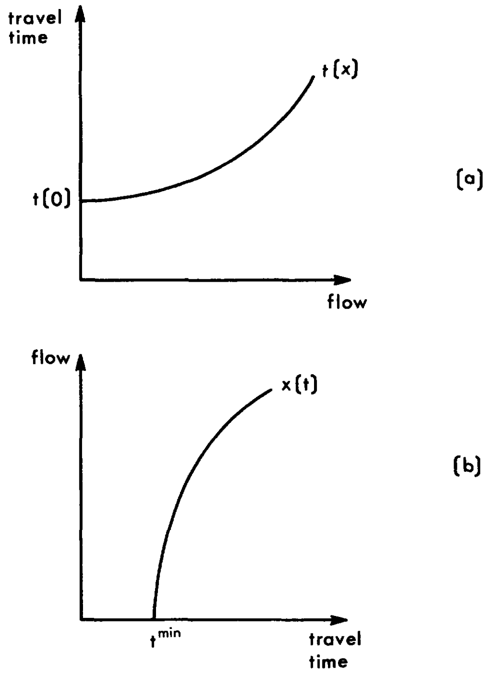


Figure 12.3 Inversion of a link performance function: (a) a link performance function; (b) the inverse of the performance function in (a).

Comparing Eq. [12.35b] with [12.23] (or Eq. [12.35a] with [12.25]), note that the gradients of $z(\mathbf{x})$ and $z(\mathbf{t})$ always have the same signs and vanish at the same points. Furthermore, the Hessian of $z(\mathbf{t})$ is given by

$$\nabla^2 z(\mathbf{t}) = \sum_{rs} q_{rs}(\Delta^{rs})(-\nabla_{\mathbf{c}} \mathbf{P}^{rs})(\Delta^{rs})^T + \nabla_{\mathbf{t}} \mathbf{x} \qquad [12.36]$$

This expression is the sum of two matrices. The first one is a positive-semidefinite matrix, as can be shown by the reader following the aforementioned arguments regarding matrix [12.32]. The second matrix in the sum, $\nabla_{\mathbf{t}} \mathbf{x}$, is diagonal with the nonzero terms being $dx_a(t_a)/dt_a$, $\forall$ a. Since $x_a(t_a)$ is increasing, these terms are all positive and the matrix $\nabla_{\mathbf{t}}(\mathbf{x})$ is positive definite. The Hessian of $z(\mathbf{t})$, $\nabla^2 z(\mathbf{t})$, is therefore positive definite since it is the sum of a positive-definite and a positive-semidefinite matrix. This means that $z(\mathbf{t})$ is strictly convex, having a single stationary point which is its minimum.

The functions $z(\mathbf{x})$ and $z(\mathbf{t})$ are related by a monotonic transformation. In other words, each point of $z(\mathbf{x})$ corresponds to one and only one point of $z(\mathbf{t})$. Furthermore, the gradients of both functions always have the same sign and vanish at the same points. The gradient of $z(\mathbf{t})$, however, vanishes only once, at the minimum of $z(\mathbf{t})$. It must be, thei., that $z(\mathbf{x})$ also has a unique minimum at this point.

This establishes the uniqueness of the minimum of $z(x)$. This function is strictly convex at the vicinity of the minimum but not necessarily convex elsewhere. Nevertheless, its local minimum is also a global one.

12.2 SOLUTION ALGORITHM

Section 12.1 described the SUE equivalent minimization program. This program is given by (see Eq. [12.7])

$$\min z(\mathbf{x}) = -\sum_{rs} q_{rs} E\left[\min_k \{C_k^{rs}\} \mid \mathbf{c}^{rs}(\mathbf{x})\right] + \sum_a x_a t_a(x_a) - \sum_a \int_0^{x_a} t_a(\omega)\, d\omega$$

[12.37]

As shown in that section, the flow pattern that solves this program also satisfies the SUE conditions. Furthermore, the program has only a single minimum. This section looks at algorithmic solution approaches to the minimization of program [12.37].

As mentioned in Chapter 4, the basic algorithmic step of most minimization procedures can be written as

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \alpha_n \cdot \mathbf{d}^n \qquad [12.38]$$

where $\mathbf{x}^n$ is the value of the decision variable vector (the link flows in this case) at the nth iteration, α_n is a scalar representing the move size, and $\mathbf{d}^n$ is a descent direction vector computed at $\mathbf{x}^n$.

The gradient vector of the SUE objective function is given by Eq. [12.25]. A natural descent direction can be the one opposite to the direction of the gradient, that is,

$$\mathbf{d}^n = \left[\sum_{rs} q_{rs} \cdot \mathbf{P}^{rsn} \cdot \mathbf{\Delta}^{rsT} - \mathbf{x}^n \right] \cdot \nabla_{\mathbf{x}} \mathbf{t}$$

A simpler descent direction can be defined by dropping the Jacobian, to wit,†

$$\mathbf{d}^n = \sum_{rs} q_{rs} \cdot \mathbf{P}^{rsn} \cdot \mathbf{\Delta}^{rsT} - \mathbf{x}^n \tag{12.39a}$$

This vector is obviously a descent direction as can be shown by taking the dot product of $\mathbf{d}^n$ and the gradient, and demonstrating that $\nabla z(\mathbf{x}^n) \cdot \mathbf{d}^{nT} < 0$ (see Problem 12.6). Each component in this descent vector is given by

$$d_a^n = \sum_{rs} q_{rs} \sum_k P_k^{rsn} \delta_{a,k}^{rs} - x_a^n \tag{12.39b}$$

In the expressions above, $\mathbf{P}^{rsn} = (\ldots, P_k^{rsn}, \ldots)$ is the vector of route choice probabilities at the nth iteration. Each component of this vector is given by

$$P_k^{rsn} = P_k^{rs}(\mathbf{c}^{rsn}) = \Pr\left(C_k^{rs} \leq C_l^{rs}, \forall\, l \mid \mathbf{c}^{rsn}\right) \qquad \forall\, k, r, s \tag{12.40}$$

The descent direction components in Eq. [12.39b] are defined as the difference between two quantities. It can therefore be written in a form that parallels the one used to describe the descent direction of the convex combinations method (see the discussion following Eq. [5.6]). Let $\{y_a^n\}$ be a set of auxiliary variables, where

$$y_a^n = \sum_{rs} \sum_k q_{rs} P_k^{rsn} \delta_{a,k}^{rs} \qquad \forall\, a \tag{12.41}$$

In other words, $\{y_a^n\}$ is the set of link flows obtained by a stochastic network loading which is based on the set of path travel times $\{\mathbf{c}^{rsn}\}$ (see Eq. [12.40]). Using this variable, the components of the descent direction can be written as

$$d_a^n = y_a^n - x_a^n \qquad \forall\, a \tag{12.42a}$$

or, in vector form, as

$$\mathbf{d}^n = \mathbf{y}^n - \mathbf{x}^n \tag{12.42b}$$

Typically, most algorithms would search for the minimum of the objective function along the descent direction. This minimum would then be defined as the next (trial) solution. The distance (along the descent direction) from the current solution point ($\mathbf{x}^n$) and that minimum, is the move size, α_n.

This iterative process of finding the descent direction and minimizing the objective function along this direction cannot be easily carried out for the SUE program in Eq. [12.37]. The reason is twofold. First, the determination of the descent direction requires, at every iteration, a stochastic network loading (see Eq. [12.41]). In some cases, unfortunately, the link flows can only be estimated, rather than computed accurately. For example, a probit network loading can be carried out only by using Monte Carlo simulation. Regardless of

†This direction is the negative gradient of objective function $z(\mathbf{t})$ in Eq. [12.34] (see Eqs. [12.35]). Consequently, it is a descent direction for that problem as well. Thus, the algorithm described below can be regarded as a form of a steepest descent method for program [12.34].

how many times the simulation procedure is repeated (and the flow averaged), the resulting link flows are still random variables which can only be regarded as estimates of the actual link flows. In such cases, the vector of auxiliary link flows, $\mathbf{y}^n$, is a random variable, meaning that the direction vector is random. Consequently, the direction vector computed at a particular iteration may not point at a descent direction (even though on the average it is a descent vector). Such a search direction, which is not always a descent direction, may cause difficulty in applying any descent method.

The second difficulty with the application of a standard descent algorithm to the minimization of the SUE program is that the move size cannot be optimized since the objective function itself is difficult to calculate. The SUE objective function [12.37] includes the terms $S_{rs}[\mathbf{c}^{rs}(\mathbf{x})]$, the expected perceived travel time between each O–D pair, where (see Eq. [12.9])

$$S_{rs}[\mathbf{c}^{rs}(\mathbf{x})] = E\left[\min_k \{C_k^{rs}\} \mid \mathbf{c}^{rs}(\mathbf{x})\right] \qquad [12.43]$$

The expected perceived O–D travel time can only be formulated in terms of paths and its calculation requires, therefore, path enumeration. Alternatively, it can be estimated by a simulation process similar to the one used to compute the choice probabilities (in the case of probit models). Such calculations, however, are computationally very time consuming. Furthermore, it would mean that the value of the objective function will be a random variable, making the line search very difficult and in many cases infeasible. (Why?)

The next section describes an algorithm that can be used to minimize the SUE program despite all these difficulties.

Method of Successive Averages

The algorithm described in this section is known as the *method of successive averages* (MSA). The method is based on a predetermined move size along the descent direction. In other words, α_n in Eq. [12.38] is not determined on the basis of some characteristics of the current solution. Instead, the sequence of move sizes $\alpha_1, \alpha_2, \ldots$ is determined a priori.

For this method to converge, the function to be minimized has to satisfy several conditions. These conditions require that the objective function be twice continuously differentiable and that its gradient vanishes only once in the feasible region (which, in the case analyzed here, is unconstrained). In addition, the algorithm has to satisfy the following requirements: The search direction has to be a descent vector and the sequence of move sizes has to satisfy the following two conditions:

$$\sum_{n=1}^{\infty} \alpha_n = \infty \qquad [12.44a]$$

$$\sum_{n=1}^{\infty} \alpha_n^2 < \infty \qquad [12.44b]$$

If both the objective function and the algorithm satisfy those requirements, the algorithm will converge† to the minimum of that objective function. Furthermore, the sequence of solutions will converge to the minimum even if a random direction vector, $\mathbf{D}^n$, is used at every iteration, provided that $E[\mathbf{D}^n]$ is strictly a descent direction. In other words, the algorithm will converge if the search direction is a descent vector only on the average. Consequently, in cases in which the descent direction is not known with certainty, an unbiased estimate of this direction can be used at every iteration.

One of the simplest move-size sequences that satisfies both conditions [12.44] is the sequence used by the MSA, which is

$$\alpha_n = \frac{1}{n} \qquad [12.45]$$

With this sequence, the nth iteration of the algorithm would be

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \frac{1}{n}\,\mathbf{d}^n \qquad [12.46a]$$

Using the search direction definition in Eq. [12.42b], this algorithmic step can be written as

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \frac{1}{n}\,(\mathbf{y}^n - \mathbf{x}^n) \qquad [12.46b]$$

where $\mathbf{y}^n = \sum_{rs} q_{rs} \cdot \mathbf{P}^{rs^n} \cdot \mathbf{\Delta}^{rs^T}$ (see Eq. [12.41]). Equation [12.46b] means that the solution at each iteration is the average of the variables $\mathbf{y}$ in all the preceding iterations. In other words,

$$\begin{aligned}\mathbf{x}^{n+1} &= \frac{n-1}{n}\,\mathbf{x}^n + \frac{1}{n}\,\mathbf{y}^n \\ &= \frac{n-1}{n}\,\frac{n-2}{n-1}\,\mathbf{x}^{n-1} + \frac{n-1}{n}\,\frac{1}{n-1}\,\mathbf{y}^{n-1} + \frac{1}{n}\,\mathbf{y}^n \\ &= \frac{n-2}{n}\,\mathbf{x}^{n-1} + \frac{1}{n}\,(\mathbf{y}^{n-1} + \mathbf{y}^n) \\ &= \frac{1}{n}\sum_{l=1}^{n} \mathbf{y}^l \qquad [12.47]\end{aligned}$$

This is the source of the algorithm's name, the method of successive averages.

The random search direction, $\mathbf{D}^n$, used with this algorithm places an additional responsibility on the MSA procedure. On the one hand, the algorithm has to converge to the minimum (as in any other minimization), and on the other hand, it has to dissipate the errors introduced by the use of a random search direction. The conditions set forth in Eqs. [12.44] can be interpreted in light of these two requirements.

†More precisely, the sequence of solutions will converge *almost surely* to the minimum.

Equation [12.44a] guarantees that the algorithm will not stop prematurely (i.e., before the minimum has been reached). Although α_n is monotonically decreasing in n, the algorithm has the potential, at every iteration m, to move from the current point, $\mathbf{x}^m$, to any other feasible point, and in particular to the minimum, $\mathbf{x}^*$. This happens since condition [12.44a] also means that

$$\sum_{n=m}^{\infty} \alpha_n = \infty \qquad \text{for any positive integer } m \qquad [12.48]$$

This would not be true if the sequence used were, for example, $\alpha_n = 1/n^2$, in which case the algorithm might terminate before the minimum is reached. Condition [12.44a] is met, though, for $\alpha_n = 1/n$ and the algorithm would therefore converge when this sequence is used to determine the move size.

The flow pattern at the nth iteration is a random variable if the search direction is random. This random variable is denoted here by $\mathbf{X}^n = (\ldots, X_a^n, \ldots)$.† Equation [12.44b] guarantees that the variance of this random variable will diminish as the iterations progress. To demonstrate how this reduction works for the series $\alpha_n = 1/n$, consider the general iteration shown in Eq. [12.46b], which leads to the average in Eq. [12.47]. The random component is introduced into $\mathbf{x}^n$ by the Monte Carlo simulation process used to compute the choice probability in Eq. [12.41]. The auxiliary link flow at the nth iteration is a random variable denoted Y_a^n, $\forall\ a$. Equation [12.47] means that the variance of each component of $\mathbf{X}^n$, var (X_a^n), is given by

$$\text{var}\,(X_a^n) = \frac{1}{(n-1)^2} \sum_{l=1}^{n-1} \text{var}\,(Y_a^l) \qquad [12.49]$$

where var (Y_a^l) is the variance of the ath component of the auxiliary solution at the lth iteration. This variance of Y_a^n is clearly bounded even if the variance of the distribution of the link travel times is very large. The reason is that the Monte Carlo simulation used to calculate $\{Y_a^n\}$ draws perceived *travel times*, not flows. The auxiliary *flow* that a link can carry is bounded from below by zero, and from above by the sum of the O–D flows. The variance of Y_a^l is thus bounded by some value, $\sigma^2 < \infty$. This means that

$$\text{var}\,(X_a^n) < \frac{1}{(n-1)^2} \sum_{l=1}^{n-1} \sigma^2 \qquad [12.50]$$

Clearly, the right-hand side of Eq. [12.50] approaches zero as n grows, meaning that the variance approaches zero as the algorithm progresses.

As mentioned above, it is sufficient that the search direction be a descent

†This random variable should not be confused with the random variable discussed in the last part of Section 10.2 (see Eqs. [10.36] and [10.40]). There, the randomness was rooted in the perception of travel time. A realization of that variable was the travel time as perceived by an individual. Here, the emphasis is on the random search direction. $\mathbf{X}^n$ is random since the search direction from $\mathbf{x}^{n-1}$, $\mathbf{D}^{n-1}$, is a random variable. These two sources of randomness are, however, intimately related. The random variable discussed here can be looked upon as an estimate for the mean of the random variable discussed in Section 10.2, given the travel-time distribution.

direction only on the average for the algorithm to converge. When the choice probabilities are computed with Monte Carlo simulation, the stochastic network loading will generate an unbiased estimate of the search direction. This will be the case regardless of how accurate the simulation is (i.e., regardless of how many simulation iterations are performed).

The MSA algorithm can now be summarized as follows:

Step 0: *Initialization.* Perform a stochastic network loading based on a set of initial travel times $\{t_a^0\}$. This generates a set of link flows $\{x_a^1\}$. Set $n := 1$.

Step 1: *Update.* Set $t_a^n = t_a(x_a^n)$, $\forall\ a$.

Step 2: *Direction finding.* Perform a stochastic network loading procedure based on the current set of link travel times, $\{t_a^n\}$. This yields an auxiliary link flow pattern $\{y_a^n\}$.

Step 3: *Move.* Find the new flow pattern by setting

$$x_a^{n+1} = x_a^n + (1/n)(y_a^n - x_a^n).$$

Step 4: *Convergence criterion.* If convergence is attained, stop. If not, set $n := n + 1$ and go to step 1.

This algorithmic framework can be applied in conjunction with any stochastic network loading model. The auxiliary link flows are obtained in step 2, from the path choice probabilities, that is,

$$y_a^n = \sum_{rs} \sum_{k} q_{rs} P_k^{rs}(\mathbf{c}^{rsn}) \delta_{a,k}^{rs} \qquad [12.51]$$

If the perceived link travel times are assumed to be, for example, normally distributed, $\{P_k^{rs}(\mathbf{c}^{rs})\}$ in Eq. [12.51] follow a probit model. In other applications, $\{P_k^{rs}(\mathbf{c}^{rsn})\}$ may follow a logit model and be calculated by using the STOCH algorithm. Other stochastic network loading models can also be used. In fact, this algorithm will work even for the deterministic UE case, in which the flows, $\{y_a\}$, are obtained by an all-or-nothing assignment. Even though the move size is not optimized (as is the case with the convex combination algorithm) the MSA procedure will converge to the correct UE solution. This convergence, however, will be slower than the convergence of the convex combinations method.

The initialization step can be based on the free-flow travel times. In other words, $t_a^0 = t_a(0)$, $\forall\ a$, as was the case with many of the other algorithms described in this book.

The convergence criterion of the MSA poses some difficulties. Due to the nature of the sequence $\{\alpha_n\}$, it can be argued that the sequence $\{\mathbf{x}^n\}$ is "forced" to converge. Consequently, stopping criteria that are based on the relative change in link flow may be misleading when used to indicate convergence of the MSA. This, however, does not turn out to be a problem in applications, where the MSA can be set to terminate after a predetermined number of iterations. Alternatively, a convergence criterion can be based on the rate of the reduction in the objective function values between successive iterations.

A more serious problem in this context is that the convergence of the MSA is not monotonic. This results both from the random search direction (which may, sometimes, point in a direction in which the objective function increases) and the fixed move size (which may "overshoot" the reduction in the objective function even if the search direction is a descent vector). A convergence measure that is generally monotonically decreasing can, however, be obtained by basing the convergence measure on the flow in the last several iterations. In this case, let $\bar{x}_a^n$ denote the flow average over the last m iterations, that is,

$$\begin{aligned}\bar{x}_a^n &= \frac{1}{m}(x_a^n + x_a^{n-1} + \cdots + x_a^{n-m+1}) \\ &= \frac{1}{m}\sum_{l=0}^{m-1} x_a^{n-l} \qquad [12.52]\end{aligned}$$

where the "offset," m, is fixed a priori. (Note that this criterion can be used only for $n > m$.) In applications, a value of $m = 3$ may suffice. Thus convergence criteria that are based on flow similarity can be used in conjunction with $\bar{\mathbf{x}}^n$. For example, criteria such as

$$\frac{\sqrt{\sum_a (\bar{x}_a^{n+1} - \bar{x}_a^n)^2}}{\sum_a \bar{x}_a^n} \le \kappa \qquad [12.53]$$

perform reasonably well. Similarly, a convergence criterion that is based on the values of the objective function in the last few iterations can be used.

The last comment regarding the MSA algorithm has to do with the move-size sequence itself. The sequence $\alpha_n = 1/n$ is not the only one that satisfies Eqs. [12.44]. In general, any sequence such that

$$\alpha_n = \frac{k_1}{k_2 + n} \qquad [12.54]$$

where k_1 is a positive constant and k_2 is a nonnegative constant, can be used. In this case, k_1 determines the magnitude of the move, whereas k_2 acts as an offset to the starting step. The sequence $\alpha_n = 1/n$ is obtained, naturally, by setting $k_1 = 1$ and $k_2 = 0$.

Example

Consider the simple network shown in Figure 12.4. This network includes two links (routes) connecting one O–D pair. The link performance functions are given by

$$t_1 = 1.25\left[1 + \left(\frac{x_1}{800}\right)^4\right] \qquad [12.55a]$$

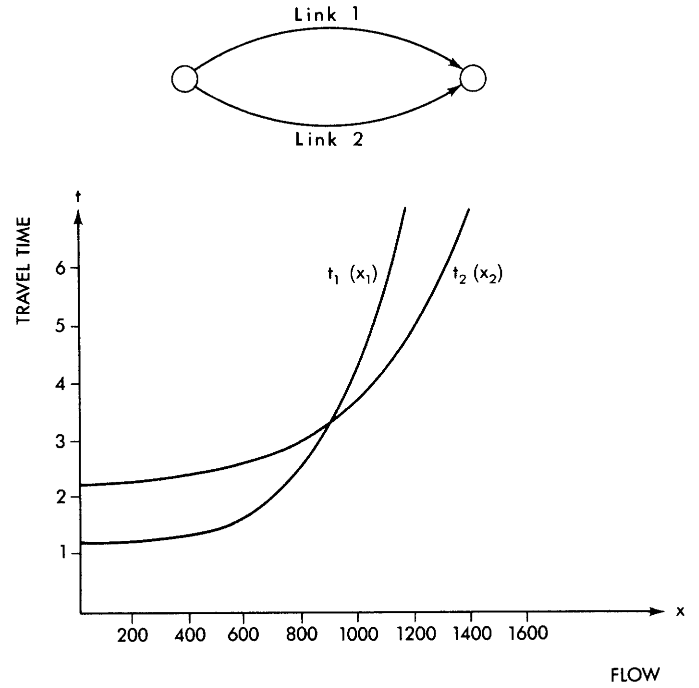


Figure 12.4 Network example with two paths connecting a single O–D pair.

$$t_2 = 2.50\left[1 + \left(\frac{x_2}{1200}\right)^4\right] \qquad [12.55b]$$

where x_a is measured in vehicles per hour and t_a in minutes. The O–D trip rate is q vehicles per hour (veh/hr). The particular SUE model used here is based on a logit route-choice model. The equilibrium condition is therefore

$$\frac{x_1}{q} = \frac{1}{1 + e^{\theta(t_1 - t_2)}} \qquad [12.56]$$

Assuming that $q = 4000$ veh/hr and $\theta = 1.0$ min^{-1}, Eqs. [12.55] and [12.56] can be solved simultaneously to give $x_1 = 1845$ veh/hr.

This problem can now be solved by using the method of successive averages. Note that the equivalent SUE minimization does not even have to be formulated, since it is never used explicitly in the solution procedure. The move-size sequence used in this example was $\alpha_n = 3/n$ (obtained by setting $k_1 = 3$ and $k_2 = 0$ in Eq. [12.54]).

Figure 12.5 displays the value of x_1^n at each of 30 iterations performed. The first observation from this figure is that the convergence pattern of the algorithm is relatively smooth and "well behaved." This result is promising because it can be expected that the characteristics of large networks will enhance the convergence rate. In large networks, each link is used by flow

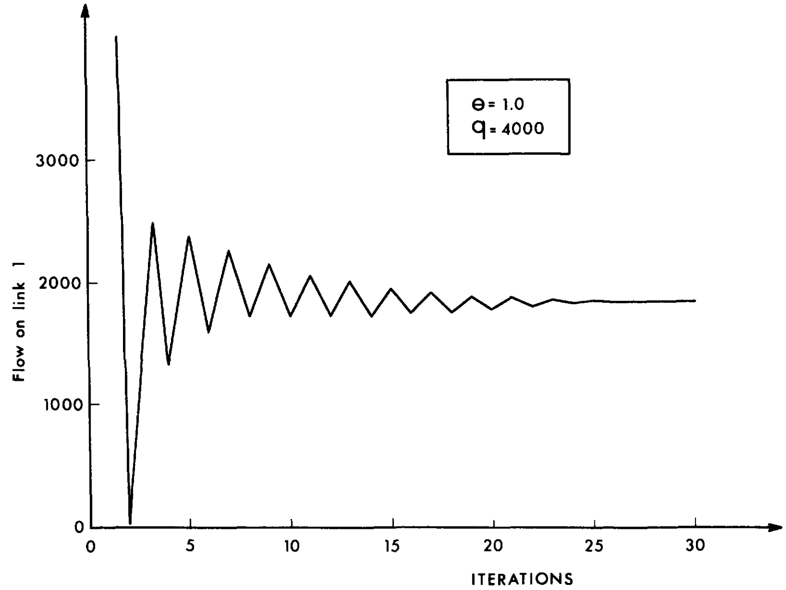


Figure 12.5 Convergence pattern of the MSA for the network in Figure 12.4; the case of relatively large perception variance ($\theta = 1.0$) and relatively high congestion level ($q = 4000$).

between many O–D pairs at each iteration and the errors tend to cancel each other.

A second observation that can be made from this figure is that at the first iteration, the flow pattern is infeasible in terms of the network constraints. At this point $x_1^2 = 4400$, a flow that is larger than the entire O–D flow.† This, however, does not impede the convergence of the algorithm, which returns immediately to the feasible region.

In general, the algorithm can be expected to converge faster with lower values of q and θ. Lower values of q indicate low congestion, implying small changes in the mean travel time between successive iterations. Low values of θ indicate large perception variance. Under such circumstances the mean travel time is less important and the perception variance plays a more important role in the flow determination. (At the limit, when $\theta \rightarrow 0$, the mean travel time is immaterial in determining the equilibrium flow pattern.) Figure 12.6 demonstrates this effect by depicting the convergence pattern at a lower O–D flow rate ($q = 2000$ veh/hr). The algorithm now converges at the fourth iteration (compared to the twentieth one before). If the variance would have been lower (i.e., higher value of θ), convergence would have been slower, as indicated by the convergence rate for $q = 2000$ veh/hr and $\theta = 2.0$ min^{-1} depicted in Figure 12.7. Convergence here is achieved only at the seventh iteration.

†The choice of the particular move-size sequence for this example was intended to create such an effect, so that the point discussed here can be demonstrated.

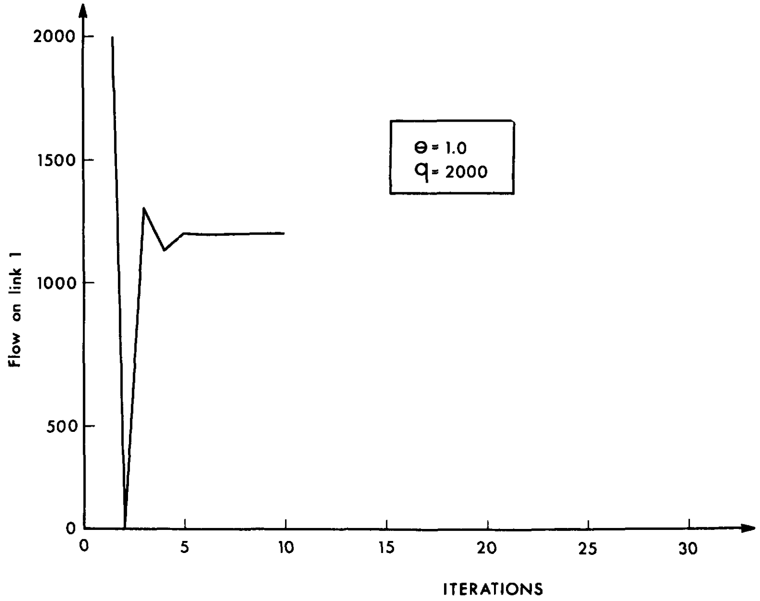


Figure 12.6 Convergence pattern of the MSA for the network in Figure 12.4; the case of relatively large perception variance ($\theta = 1.0$) and relatively low congestion level ($q = 2000$).

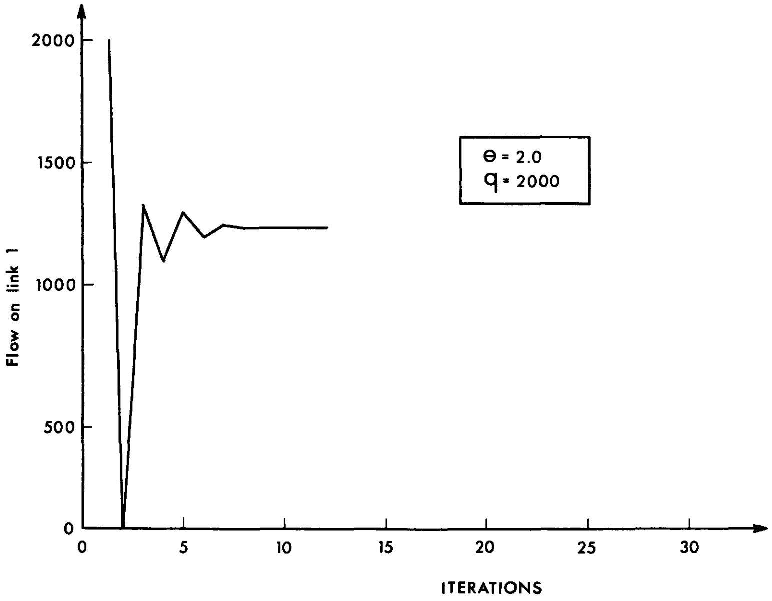


Figure 12.7 Convergence pattern of the MSA for the network in Figure 12.4; the case of relatively small perception variance ($\theta = 2.0$) and relatively low congestion level ($q = 2000$).

12.3 MORE ABOUT STOCHASTIC USER EQUILIBRIUM

This section extends the discussion of SUE and the MSA in several directions. First, the application of the MSA to probit-based SUE is explored. This application involves some computational trade-offs which are investigated in the first part of this section. The second part of the section extends the SUE framework to include link interactions. The last part looks at the possible differences between SUE and UE flow patterns, suggesting some guidelines for methodology selection. This discussion leads to a mention of a combined UE and SUE model.

Computational Trade-offs for Probit-Based SUE

When the MSA is applied to the solution of probit-based SUE problems, the algorithm involves two types of iterations. The first type are the sampling/assignment iterations associated with the stochastic network loading algorithm. These can be termed "inner iterations." The second type are the equilibrium iterations, which can be termed "outer iterations."

An interesting question here is what is the number of inner iterations that should be used (per outer iteration) so that the algorithm would be the most efficient. Clearly, if the number of inner iterations is large, the descent direction computed at every outer iteration will be more accurate and the algorithm will converge in a smaller number of outer iterations. The computational costs of traffic assignment over realistic-size networks, however, are not only a function of the number of outer iterations. Instead, these costs are a function of the number of all-or-nothing assignments executed. Consequently, the computational costs are proportional to the total number of (inner × outer) iterations.

The trade-off described here is very similar to the one discussed in Section 8.2 with regard to the diagonalization algorithm for solving UE problems with interdependent flows and travel times. In that case, the algorithm consisted of repeated solutions of standard UE problems (referred to as subproblems) each requiring an interative solution approach. The issue there was to determine the accuracy with which these subproblems need be solved, so that the overall algorithm would be the most efficient. As in the case discussed here, the computational costs were directly proportional to the total number of all-or-nothing assignments performed.

To get some intuition regarding the number of inner iterations that should be used in the case of probit SUE problems, the two-link network model shown in Figure 12.4 can be solved with varying the number of inner iterations per outer iteration. The probit assumptions made in this case were

$$T_1 \sim N(t_1, \beta t_1^0) \qquad [12.57a]$$

$$T_2 \sim N(t_2, \beta t_2^0) \qquad [12.57b]$$

$$\text{cov}\,(T_1, T_2) = 0 \qquad [12.57c]$$

where β is the variance of a road segment of unit travel time under free-flow conditions. The equilibrium conditions are

$$x_1 = q\ \Pr\,(T_1 \leq T_2) \qquad [12.58a]$$

$$x_2 = q - x_1 \qquad [12.58b]$$

as before.

Figures 12.8 through 12.10 show the convergence rate of the MSA by depicting x_1 versus the number of all-or-nothing assignments performed. In all cases the figures are drawn for q = 3000 veh/hr and β = 0.5. Figure 12.8 shows the convergence rate with 20 inner iterations per outer iteration. Convergence is achieved, in this case after, say, 80 to 100 all-or-nothing assignments. Figure 12.9 depicts the convergence pattern with five inner iterations per outer iteration. In this case, the same degree of convergence is achieved after 40 to 60 assignments. Figure 12.10 depicts the convergence pattern with only one inner iteration per outer iteration. This figure displays the best convergence pattern. Convergence is achieved after 20 to 40 all-or-nothing iterations.

It is interesting to note that the conclusion here is similar to the one reached with regard to the diagonalization algorithm. In both cases it is best to use the minimum amount of computational effort on the intermediate (or

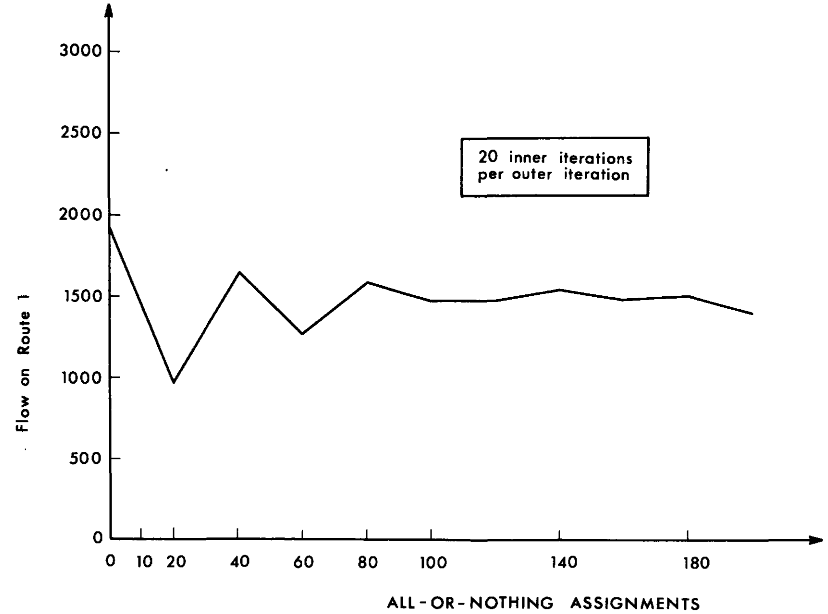


Figure 12.8 Convergence rate of the probit-based MSA for the network in Figure 12.4 (with q = 3000 and β = 0.5); the case of 20 simulation (inner) iterations per equilibrium (outer) interation.

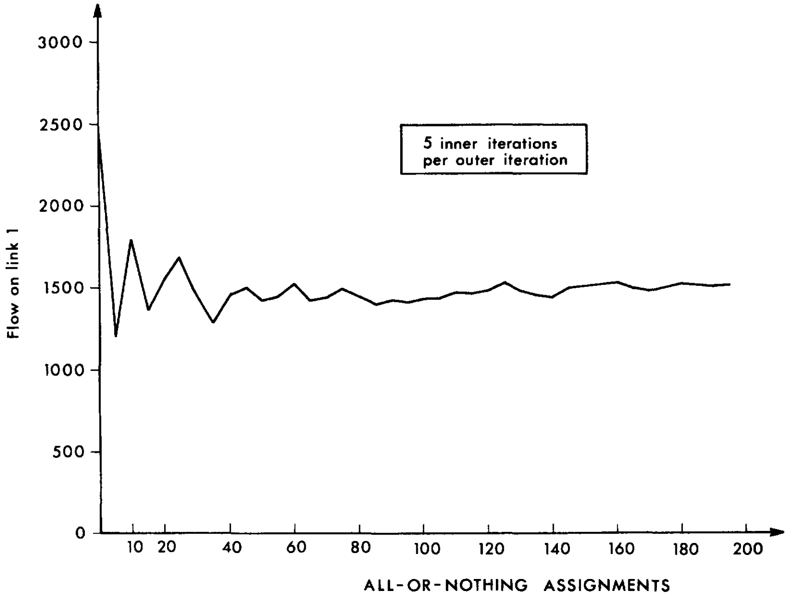


Figure 12.9 Convergence rate for probit-based MSA for the network in Figure 12.4 (with $q = 3000$ and $\beta = 0.5$); the case of five simulation iterations per equilibrium iteration.

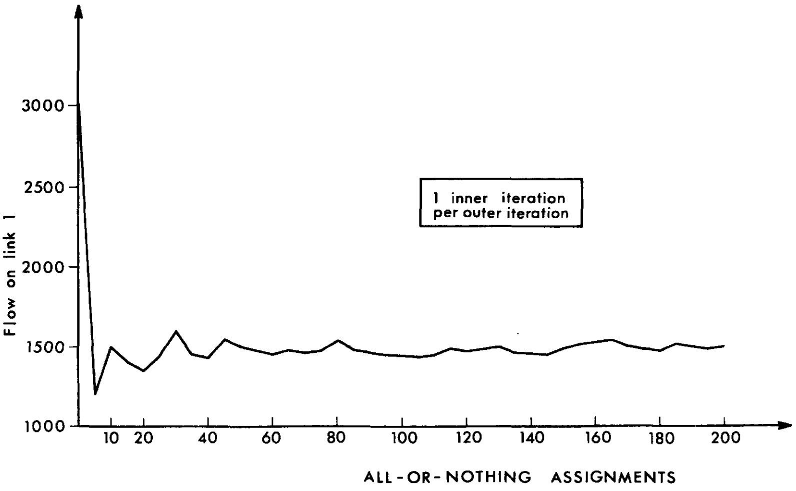


Figure 12.10 Convergence rate for probit-based MSA for the network in Figure 12.4 (with $q = 3000$ and $\beta = 0.5$); the case of a single simulation iteration per equilibrium iteration.

sub-) problems. In the diagonalization algorithm case this means that the streamlined algorithm discussed in Section 8.2 would be preferred to the original algorithm, as argued in that section. In the case of probit-based SUE problems and the MSA, it is best to use a single drawing of perceived travel time from each link, coupled with a single all-or-nothing assignment, at every outer iteration.

The convergence patterns demonstrated for the small example used in this case are indicative of the same trend in larger networks. The main difference between the network example used here and the larger networks is that convergence is faster for larger networks. The number of all-or-nothing assignments required for solving SUE problems over large networks may be about two or three times higher than the number of assignments required to solve standard UE problems over the same networks. In other words, it may require, say, 10 to 12 all-or-nothing assignments in cases where the UE problem would require 4 to 5 assignments.

SUE with Link Interactions

The formulation of the equivalent SUE program assumes that the travel time on each of the network links is a function of the flow on this link only. As mentioned in the introduction to Chapter 8, however, there are many cases in which this assumption does not hold.

As in the deterministic case, there is no equivalent minimization formulation that can be used to solve SUE problems with a general (asymmetric) link interaction pattern. If, however, the Jacobian of the link performance function is positive definite, there exists a unique flow pattern that satisfies the SUE conditions.

SUE problems with asymmetric link interactions can be solved by using the diagonalization algorithm introduced in Section 8.2. This algorithm proceeds by fixing all the link interaction terms at their current level at each iteration. This leads to a standard SUE subproblem which can be solved by the methods described in Chapter 12. Once the solution of the subproblem is obtained (in terms of a link flow pattern), all link travel times are updated and the iterations continue. The algorithm converges when the solution of two successive iterations is similar.†

This procedure includes two levels of subproblems. The first one consists of a solution of the SUE program at each iteration, using the MSA. This problem, in turn, may include a series of inner iterations within each outer iteration of the MSA, when the underlying network loading model is probit-based. The problem then is to determine the accuracy required in solving these subproblems.

As might be expected from the consideration of numerical issues in

†Stochastic variations can be smoothed out by using a moving average (see Eq. [12.52]) in the flow comparisons.

Section 8.2 and the first part of this section, it may be best not to solve the subproblems with great accuracy. In fact, one inner iteration per outer iteration, and one outer iteration per SUE subproblem, may be the best strategy to follow. The streamlined SUE diagonalization algorithm (for probit and other simulation-based network loading models) can thus be summarized as follows:

Step 0: *Initialization.* Draw a perceived link time, T_a^0, for each link, based on $t_a^0 = t_a(\mathbf{0})$, $\forall\ a$. Use all-or-nothing (based on $\{T_a^0\}$) to obtain a flow pattern $\mathbf{x}^1 = (\ldots, x_a^1, \ldots)$. Set $n := 1$.

Step 1: *Update.* Set $t_a^n = t_a(\mathbf{x}^n)$, $\forall\ a$. Draw T_a^n for each link, given t_a^n.

Step 2: *Direction finding.* Perform an all-or-nothing assignment based on $\{T_a^n\}$. This yields a set of auxiliary link flow $\{y_a^n\}$.

Step 3: *Move.* Set $x_a^{n+1} = x_a^n + (1/n)(y_a^n - x_a^n)$, $\forall\ a$.

Step 4: *Convergence test.* If $\mathbf{x}^{n+1} \simeq \mathbf{x}^n$, stop. Otherwise, set $n := n + 1$ and go to step 1.

This algorithm is similar to the MSA but for one difference; the flow update is based on all the link flows, due to the interaction effects. The move size in the foregoing description of the algorithm can be generalized to be $k_1/(k_2 + n)$, where k_1 and k_2 are constants (see Eq. [12.54]). The convergence test can be based on the absolute or relative magnitude of the change in the link flows between successive iterations (see Eq. [8.36]).

Note on Methodology Selection

As shown in Section 12.1, the SUE equilibrium conditions reduce to the UE conditions when the variance of the perceived travel times is zero. Thus the smaller this variance is, the better would a UE model approximate the results of an SUE model. The UE and the SUE models would be similar also in cases in which the network is highly congested. The reason for this lies in the shape of the link performance functions, as explained below.

Consider, for example, the simple two-link network in Figure 12.11. Assume that the volume-delay curves for this network are

$$t_1 = \frac{1}{1 - x_1} \qquad \text{for } x_1 < 1 \qquad [12.59a]$$

$$t_2 = \frac{1}{1 - x_2} + \delta \qquad \text{for } x_2 < 1 \qquad [12.59b]$$

as shown in the figure. The total O–D flow is $x_1 + x_2 = q$ (which must be less than 2 flow units for a solution to exist, due to the link capacities). Assume now that the O–D flow is very high (causing both links to operate close to

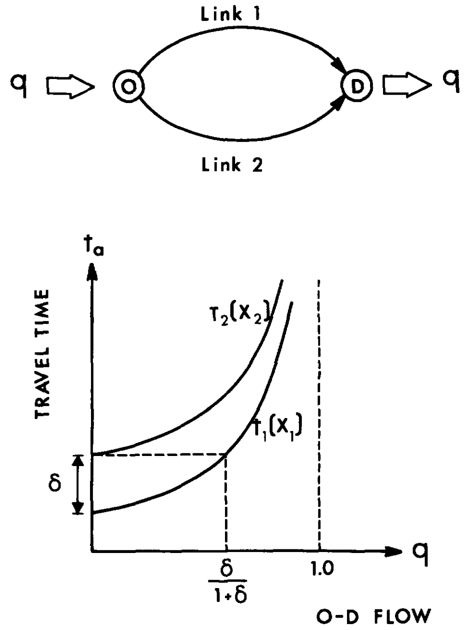


Figure 12.11 Simple network example with two links connecting one O–D pair. The bottom part shows the link performance functions.

their capacity) and that the network is in user equilibrium (i.e., $t_1 = t_2$). At this point, if one motorist (representing Δx units of flow) switches routes, say from link 1 to link 2, the travel time on link 1 would drop by Δt_1 and the travel time on link 2 would increase by Δt_2. If the flow is very high on both routes, the derivative of $t_a(x_a)$ at the UE solution is very large. Consequently, both Δt_1 and Δt_2 are very large and the resulting change in the routes' travel time would be noticeable by motorists even if the travel time is subject to random perception error. Therefore, no motorist has the incentive to make such a switch of routes, and the user equilibrium will also be the SUE solution for the system.

For a given (finite) perception variance and a given flow increment, Δx, there is always some level of flow at which the UE and the SUE solutions will practically coincide for the network example under consideration. Furthermore, as the O–D flow grows, the SUE solution will grow similar to the UE solution. The reason is that the derivative of the performance functions grows monotonically as the flow increases.

Figure 12.12 depicts the share of flow on link 1 (x_1/q) for the network under consideration, assuming a logit-based path choice model with parameter θ. In other words,

$$\frac{x_1}{q} = \frac{1}{1 + e^{\theta(t_1 - t_2)}} \qquad [12.60]$$

The figure shows this share as a function of the total O–D flow, q, for various

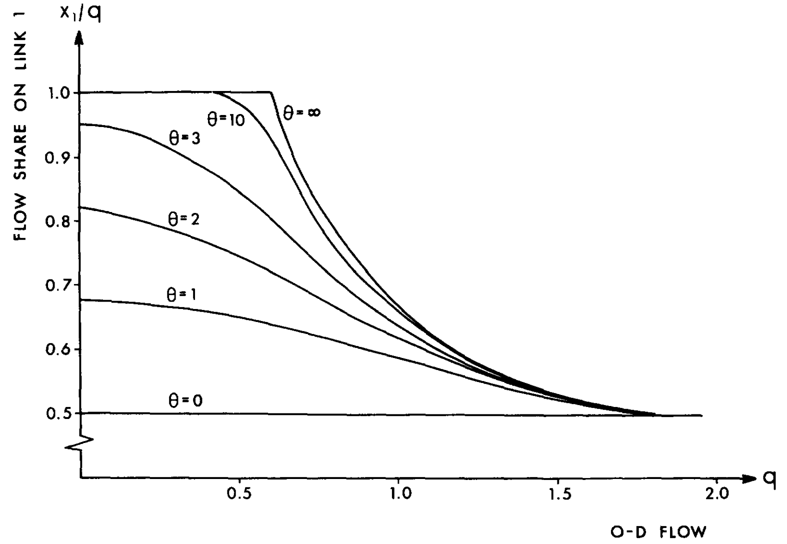


Figure 12.12 The equilibrium share of flow on link 1 versus the O–D flow, as a function of the logit assignment parameter, for the network in Figure 12.11.

values of θ. The topmost curve (labeled $\theta = \infty$) represents the UE solution of the system, as a function of q.

This figure clearly demonstrates the growing similarity between the SUE solution and the UE solution as θ grows larger. (Recall that large values of θ imply small variance of the perceived link times in logit models.) The figure shows also that the SUE and UE flows grow in similarity as q grows, regardless of the value of θ. In addition, note that the value of the SUE solution for a given θ does not change appreciably between $q = 0$ and $q > 0$, as long as q is small. Similar results hold true for larger networks and in cases in which congestion is not perfectly uniform.

The conclusion from these observations is that when the network is uniformly congested, the equilibrium effects are stronger than the effects of inaccurate travel-time perception. Consequently, the UE solution is a good approximation for the SUE solution for congested networks. At small values of q_{rs} the stochastic perception effects are stronger than the equilibrium effects. Consequently, stochastic network loading models (without any equilibrium effects) may be used to approximate SUE flow patterns in uncongested networks. The SUE approach should be used at intermediate congestion levels where neither approximation is accurate.

The method of successive averages, when applied to an uncongested network, will converge in a few (one or two) iterations and thus no special considerations should be given to this case. As shown earlier in this section, however, the MSA is slower to converge when it is applied to congested networks. If the network is very congested, it may be appropriate to use a UE approximation to the problem by ignoring the stochastic components. The problem can then be solved by applying the convex combinations method or any other appropriate algorithm.

Joint Travel Choices and Hypernetworks

Chapter 9 presented an extended network model that can account for several travel decisions, including the decision to take a trip, the choice of destination and the choice of mode of travel, in addition to the route choice. Both mode and destination choices were modeled by using a logit formula. Probit models, however, are conceptually easier to use since dummy links can be added to the network in a straightforward manner, making its equivalent network representation quite intuitive. The impedance on each dummy link (representing some travel choice such as a mode or a destination) can be taken directly from the appropriate discrete choice model to be the utility (or disutility) of the alternative under consideration. In other words, no function inversions are necessary as was the case with the logit model discussed in Chapters 7 and 9.

For example, consider a probit mode choice model, which includes transit and automobile. The utility functions estimated for the model are (see Section 10.1)

$$U_{\text{tr}} = \phi_1 - \phi_2 \hat{t} + \xi_{\text{tr}} \qquad [12.61\text{a}]$$

$$U_{\text{au}} = \phi_3 - t + \xi_{\text{au}} \qquad [12.61\text{b}]$$

where U_{tr} and U_{au} are the transit and automobile utility functions, respectively; ϕ_1, ϕ_2, and ϕ_3 are model parameters; $\hat{t}$ and t are the transit and automobile travel times, respectively; and ξ_{tr} and ξ_{au} are the transit and automobile utility error terms, respectively. For the probit model, it is assumed that

$$(\xi_{\text{tr}}, \xi_{\text{au}}) \sim \text{MVN}\,[(0, 0), \boldsymbol{\Sigma}] \qquad [12.62]$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of $\boldsymbol{\xi} = (\xi_{\text{tr}}, \xi_{\text{au}})$.

The joint (probit-based) mode split/traffic assignment program can be solved by considering the augmented network shown in Figure 12.13 (such a supernetwork has been termed *hypernetwork* in the literature). In this figure a

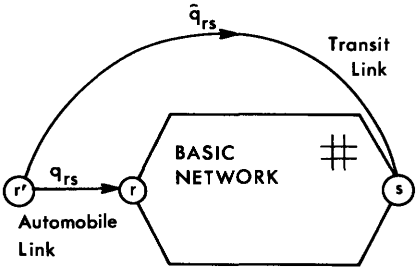


Figure 12.13 Hypernetwork representing joint (probit-based) modal split and (UE-based) traffic equilibrium model.

dummy node, r', is added to each origin, r. This node is then connected to both the destination, s, by a transit (dummy) link and the origin, r, by an automobile link. The equivalent travel time on the transit link, $r' \rightarrow s$, is the disutility of transit, $-U_{tr} = -\phi_1 + \phi_2 \hat{t} - \xi_{tr}$. The equivalent travel time on the automobile link, $r \rightarrow r'$, is the auto disutility, excluding the auto travel time that is incurred on the basic network. Thus, for this example, $t_{r'r} = -U_{au} = -\phi_3 - \xi_{au}$.

The equivalent travel time on each of the dummy links is a random variable. The travel time on the basic links is modeled as a deterministic entity. Thus the hypernetwork structure exhibits a separation between two types of links. The dummy links include a random equivalent travel time, but this travel time is not flow dependent. The basic network links are flow dependent but not random. This modeling approach thus includes only the most important effects for each type of link. It means that, for this example, the mode choice will be given by a simple probit model. The basic network, however, will exhibit a standard (deterministic) user equilibrium, given $\{q_{rs}\}$. The structure of the hypernetwork is very similar to the supernetworks discussed in Chapter 9. The latter concept is based primarily on logit models, while hypernetworks are based on probit models. In both cases, however, the basic network is augmented by dummy links and/or nodes.† Hypernetworks can be solved with a double-stage type of algorithm (see Chapters 7 and 9), in which the direction-finding step is executed as follows:

(a) Compute the minimum-path travel times between all O–D pairs over the basic network.
(b) Conduct a probit-based stochastic network loading over the hypernetwork, determining the O–D flows for the basic network.
(c) Assign the O–D flows to the paths found in (a) (over the basic network).

The intricate features of the hypernetwork approach and the details of the solution algorithm are beyond the scope of this chapter (see references in Section 12.5). It should be noted, however, that the probit loading in phase 2 can be accomplished with the analytic approximation mentioned in Section 10.1 (simulation is not necessary here) and that the algorithm does not utilize the fixed-move-size approach of the MSA, but an actual move-size optimization. This approach can also be used with heterogeneous populations of drivers (i.e., when the probit model includes variables, such as income, which vary across individuals).

†Alternatively, hypernetworks can be looked upon as a degenerate case of SUE in which the variance of the perceived travel time on some of the (augmented) network links (the basic links) is zero, and the mean travel time on other links (the dummy ones) is independent of the flow.

12.4 SUMMARY

This chapter discusses a traffic assignment problem in which the link travel times are assumed to be both random and flow dependent. The randomness of these travel times stems from the variability in their perception by motorists, whereas the flow dependency is rooted in congestion phenomena. It is still assumed that each motorist minimizes his or her (perceived) travel time and thus at equilibrium no driver can improve this perceived travel time by unilaterally changing routes. This is the definition of the stochastic user equilibrium (SUE), which is formalized in this chapter.

Section 12.1 formulates an unconstrained minimization program, the solution of which is the SUE flow pattern. The minimum of this program is unique (even though the program is not necessarily convex). Nevertheless, the solution of the equivalent SUE program is not straightforward since the objective function and its derivatives are formulated in terms of paths. Furthermore, the calculation of the objective functions and its derivatives sometimes involves Monte Carlo simulation, meaning that their values at any iteration may be subject to random variations.

The method of successive averages (MSA), presented in Section 12.2, is an algorithm that can be successfully applied in this environment. It converges even if the search direction is actually a descent direction only on the average, and it uses a predetermined sequence of step sizes so that the objective function need not be evaluated at any stage of the algorithm. The speed of convergence of the MSA is influenced by the problem parameters, including the level of network congestion and the variance of the travel-time perception. Convergence is faster at low congestion level and large perception variance.

The last section of this chapter presents convergence considerations which imply that for probit SUE problems (in which the network loading is accomplished in each step via simulation/assignment iterations), it is best to use only one such iteration per loading step.

The last section also deals with the issue of methodology selection. UE is a good approximation for SUE in highly congested networks and should therefore be used in such cases. Thus, in cases in which the SUE problem involves a considerable computational burden it can be approximated by a standard (deterministic) UE assignment.

12.5 ANNOTATED REFERENCES

Stochastic user equilibrium was first formulated by Daganzo and Sheffi (1977), who also show that it is a generalization of the UE criterion. The equivalent SUE program was developed by Sheffi and Powell (1982), following Daganzo (1979), who suggested a related formulation for economic equilibria with discrete choice models. Daganzo (1982) also applied his work to SUE problems, using the travel-time-based formulation (see Eq. [12.34]) rather than a flow-based formulation.

The MSA algorithm has been used for the network equilibrium problem (UE) as an incremental-assignment-type heuristic by many researchers, including Almond (1967) and Fisk (1980) who investigated its convergence. Powell and Sheffi (1982) provided a formal proof of convergence for the MSA in its applications to network equilibrium problems, including the UE formulation and Fisk's (1980) logit-based SUE formulation given in Problem 12.11. Their proof is based on the general convergence proof for these methods given by Blum (1954). A detailed discussion of the MSA is provided by Wilde (1964).

The convergence characteristics of the MSA for the SUE problem were investigated by Sheffi and Powell (1981) using a small test network. The advantages of using a single inner iteration at each outer iteration of the MSA were demonstrated by Mimis (1984) for larger networks. Some convergence results for logit equilibrium problems are reported by Sheffi (1981) and by Daganzo (1982). The latter reference includes a detailed development of the SUE approach, including its application to heterogeneous populations and networks with interacting link performance functions.

The guidelines for methodology selection given in the last section were suggested by Daganzo (1977b) and investigated by Sheffi and Powell (1981). The hypernetwork approach of combining UE and SUE models using probit-based network loading models was developed by Sheffi and Daganzo (1978a, 1980).

12.6 PROBLEMS

12.1. Show that Eqs. [12.6] hold if and only if the UE conditions hold.

12.2. Calculate the gradient of the SUE objective function with respect to path flows and verify the equivalency between this program and the SUE conditions.

12.3. **(a)** Consider the Hessian of the SUE objective function, given in Eq. [12.31]. Write out the term in the row corresponding to the *a*th link and the *b*th link in the network. Compare your answer to Eq. [12.30].

***(b)** Prove that if **H** is a positive definite matrix and *h* is a nonsingular square matrix, then $\mathbf{h} \cdot \mathbf{H} \cdot \mathbf{H}^T$ is a positive definite matrix.

12.4. Show that the change of variable $x_a = x_a(t_a)$ can be used to transform the SUE program, $z(\mathbf{x})$, into a function of **t**, $z(\mathbf{t})$ (see Eq. [12.33]). Given that, carry out the integration to obtain Eq. [12.34].

12.5. Consider a network including two links connecting a single O–D pair. The link performance functions are

$$t_1(x_1) = 2 + 0.01x_1^4$$

$$t_2(x_2) = 1 + 0.02x_2^4$$

The total O–D flow is $x_1 + x_2 = 6$ and the stochastic network loading model is based on the logit formula, that is,

$$\frac{x_1}{x_1 + x_2} = \frac{1}{1 + e^{\theta(t_1 - t_2)}}$$

Assume that $\theta = 0.5$.

(a) Formulate the SUE equivalent minimization.

(b) Show that the objective function has a unique minimum.

(c) Apply the MSA to this program and show the results of the intermediate iterations.

(d) Solve the problem analytically and check your answer to part (c).

12.6. **(a)** Show that **d** is a descent direction for a function, z if and only if $\nabla \mathbf{z} \cdot \mathbf{d}^T < 0$.

(b) Show that $\mathbf{d}^n$ in Eq. [12.39] is a descent direction for program [12.37].

12.7. **(a)** Devise an algorithm for computing (or estimating) the SUE objective function for probit-based stochastic assignment.

(b) Discuss the difficulties of conducting a line search over a convex function, the value of which can only be known with a certain error.

12.8. Describe in detail the modification that you will have to introduce into a standard UE convex combinations code in order for it to perform the MSA procedure coupled with (simulation-based) probit assignment.

12.9. Consider the network in Figure P12.1, where

$$t_1 = 3$$

$$t_2 = 5$$

$$t_3 = 2 + x_3$$

$$t_4 = 1 + 2x_4$$

$$q_{1,4} = 4$$

$$q_{2,4} = 2$$

Assume that route choice is governed by a logit model with parameter $\theta = 0.3$.

(a) Formulate the SUE objective function for this problem in terms of link flows only.

(b) Find the equilibrium flow pattern.

(c) How will the equilibrium flows change if θ is greater than 0.3?

12.10. Section 12.3 recommends one inner iteration per outer iteration for probit SUE problems. How should this recommendation change for **(a)** uncongested networks, **(b)** congested networks, **(c)** high variance of travel time perception, and **(d)** low variance of travel time perception?

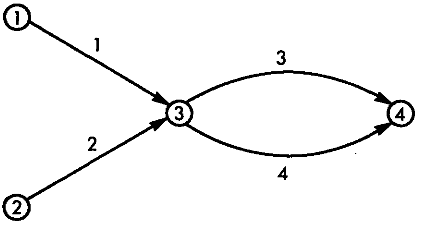


Figure P12.1

12.11. The program

$$\min z(\mathbf{f}) = \frac{1}{\theta} \sum_{rs} \sum_{k} f_k^{rs} \ln f_k^{rs} + \sum_{a} \int_{0}^{x_a} t_a(\omega)\, d\omega$$

subject to

$$\sum_{k} f_k^{rs} = q_{rs} \qquad \forall\ r, s$$

$$x_a = \sum_{rs} \sum_{k} f_k^{rs} \delta_{a,k}^{rs} \qquad \forall\ a$$

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s$$

has been suggested as an equivalent minimization program for logit-based SUE. Investigate its first- and second-order conditions. Compare this program to the formulation given in this chapter.

V

Input Data

13

Background for Developing the Analysis Inputs

An equilibrium analysis of a transportation network requires large amounts of input data. These include the physical layout of the network, the volume–delay curves, and the demand models used to represent the various travel choices included in the analysis. These models may include, for example, O–D demand functions for variable-demand equilibrium problems or modal split functions for joint mode/assignment equilibrium problems. In addition, most equilibrium assignment models require a specification of the O–D trip matrix. Such a matrix can be developed from a series of demand models (as done in the traditional four-step urban transportation planning process) or directly from link data.

This chapter describes some of the background material needed for developing a data set for equilibrium analysis. Section 13.1 discusses the principles behind link performance functions, Section 13.2 explains some fundamentals of econometric estimation theory used in calibrating demand models (such as the multinomial logit), and Section 13.3 outlines a technique for direct estimation of O–D matrices.

13.1 LINK PERFORMANCE FUNCTIONS

This section introduces some principles of traffic flow theory on which the derivation of link performance functions is based. It deals separately with traffic delays on road segments and delays at (both signalized and unsignalized) intersections.

The focus of the discussions in this section is on the connection between traffic models and link performance functions. These functions relate the travel time on a link to the traffic flow on that link.

Traffic Flow Models

To analyze the traffic flow along a road segment, consider the trajectory of vehicles' movements as shown in a *time–space diagram.* Such a diagram (an example of which is shown in Figure 13.1) is a plot of the movements of one or more vehicles in time, along a given road segment. The slope of each trajectory is the speed of the vehicle and the horizontal and vertical distances between the trajectories of two vehicles give the headway and spacing, respectively, between these vehicles, as shown in the figure.

The three fundamental traffic characteristics are the *flow*, the *density*, and the *speed.* All these quantities are defined as averages over a number of vehicles, as follows:

1. *Flow* (denoted q): average number of vehicles passing a fixed point in the road per unit of time
2. *Density* (denoted K): average number of vehicles present at some instant on a section of the road of unit length

The average speed can be measured either by a stationary observer (recording and averaging the speed of passing cars) or by averaging the flow of all cars on a given road segment at some instant. Accordingly, two speed averages can be defined:

1. *Space mean speed* (denoted $\bar{u}_s$): average speed of vehicles along a road segment at some instant

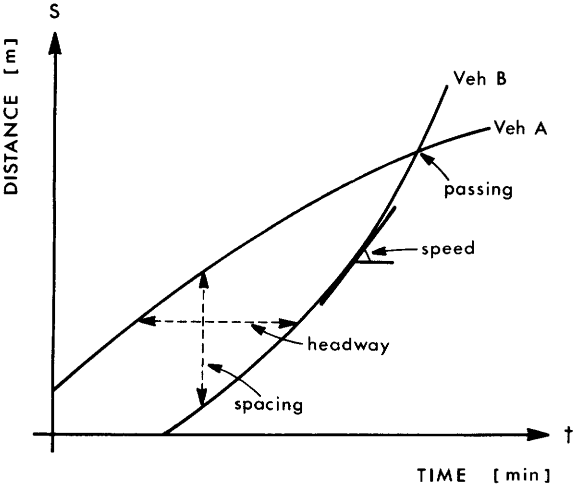


Figure 13.1 Time–space diagram of two vehicles' movements.

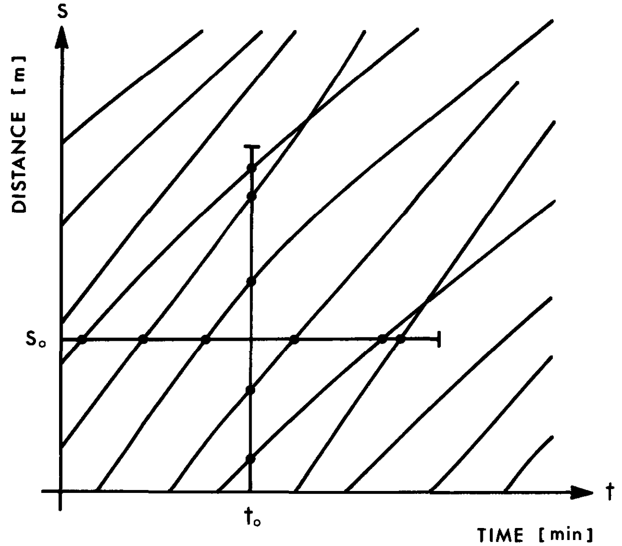


Figure 13.2 Measuring the fundamental traffic characteristics on a time–space diagram. The horizontal line segment intersects all car trajectories observed at s_0 for a length of time indicated by the length of this segment. The vertical segment intersects all car trajectories captured in an aerial photograph of the road segment at time t_0.

2. *Time mean speed* (denoted $\bar{u}_t$): average speed of vehicles passing through a given point

Figure 13.2 demonstrates the measurement of the fundamental traffic characteristics on a time–space diagram. A stationary observer can measure the flow and the time mean speed at a point, s_0, while the density and the space mean speed can be measured (e.g., by means of aerial photography) at an instant, t_0. Since traffic characteristics change in time and space, their measurement is not trivial. The measurement interval has to be not too small, so that the measured quantities will not be erratic, yet not too long, so that the interesting variations will not be averaged out.

The equation

$$q = \bar{u}_s K \qquad [13.1]$$

is known as the fundamental relationship of traffic flow. Usually, the flow is measured in veh/hr, the speed in miles (or km) per hour, and the density in veh/mile (or veh/km). Note that the fundamental relationships are expressed in terms of the space mean speed (which is always less than or equal to the time mean speed—see Problem 13.2). To exemplify these concepts, consider a 2-km-long race track with three cars, as shown in Figure 13.3. The speeds of the three cars are 100, 120, and 140 km/h, respectively. Given these data, all the

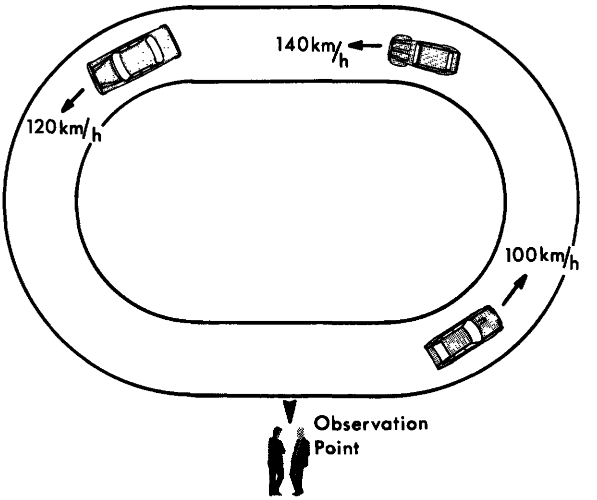


Figure 13.3 Racetrack example.

fundamental traffic characteristics can be calculated as follows:

1. *Density.* Three vehicles on a 2-km road mean that $K = 1.5$ veh/km.
2. *Flow.* A stationary observer will see, in 1 hour, 50 times the 100-km/h car, 60 times the 120-km/h car, and 70 times the 140-km/h car. Thus $q = 50 + 60 + 70 = 180$ veh/hr.
3. *Space mean speed.* Observed at an instant, $\bar{u}_s = \frac{1}{3}(100 + 120 + 140) = 120$ km/h.
4. *Time mean speed.* Computed by multiplying the number of times that each vehicle is observed (in 1 hour), by its speed and averaging, that is, $\bar{u}_t = (50 \cdot 100 + 60 \cdot 120 + 70 \cdot 140)/180 = 122$ km/h.

Note that these quantities satisfy the fundamental relationship, since $q = K\bar{u}_s$; also, $\bar{u}_s \leq \bar{u}_t$, as expected.

The basic premise of traffic flow models is that the speed is a decreasing function of the density. As the density increases, the spacing between vehicles decreases and drivers react by lowering their speed. One of the most widely used traffic models is the following line:†

$$u(K) = u_f - \frac{u_f}{K_j} K \tag{13.2}$$

where u_f is the free-flow speed (which is observed at zero density) and K_j is the jam density (at which the road exhibits a "parking lot syndrome").

Using this (or any other speed–density curve), the flow can be studied as

†The original model was estimated by Greenshields to be $u = 46 - 0.236K$, where the (space mean) speed is measured in mph and the density is veh/mile.

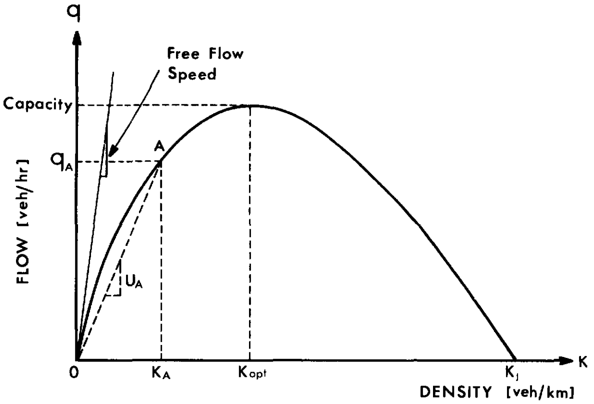


Figure 13.4 Fundamental diagram of traffic flow for a given road segment.

a function of the density since, in accordance with Eq. [13.1], $q = Ku(K)$. The plot of q versus K, which is known as the *fundamental diagram*, is shown in Figure 13.4. As the figure shows, the flow is zero if the density is zero ($K = 0$) or if the road is jammed (in which case $u = 0$). At some "optimal" density (K_{opt}) the throughput is maximized. The flow at this point is known as the road's *capacity*.

Any point on the fundamental diagram (such as A in Figure 13.4) represents a set of possible traffic conditions. The (space mean) speed at this point is the slope of the line connecting the point to the origin. Traffic situations in which the density is higher than K_{opt} represent unstable conditions with no passing opportunities (such as the conditions in a queue behind a slow-moving vehicle).

For the linear model given in Eq. [13.2], the fundamental diagram is a parabola, the equation of which is given by

$$q(K) = u_f K - \frac{u_f}{K_j} K^2 \qquad [13.3]$$

The capacity of the road according to this model is $\frac{1}{4}u_f K_j$. This capacity occurs at $K = K_j/2$, at a speed of $u_f/2$ (see Problem 13.4).

The speed–density model and the fundamental relationship define the relationships between all three traffic variables. The focus in this book is on the relationship betwen travel time on a given road segment and the flow on this segment. Such a function is shown in Figure 13.5, which depicts travel time versus flow for Greenshield's model given in Eq. [13.2] (see also the related footnote). The backward turn of this performance curve is not observed in urban streets, the performance function for which is given by a monotonically increasing curve (see, e.g., Figure 1.8). The reason for this is that most of the delay in urban networks is due to intersection queueing rather than interactions among vehicles in moving traffic. As shown in the next two sections, intersection delay is a monotonically increasing function of the flow.

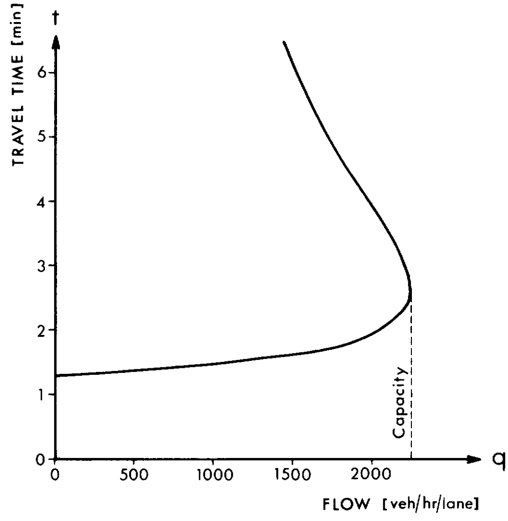


Figure 13.5 Travel time versus flow for Greenshield's speed–density relationships.

Furthermore, the magnitude of this delay is substantially larger than the delay in moving traffic along urban streets.

Delay at Signalized Intersections

To analyze the delay at a given approach to a signalized intersection, assume that vehicles arrive at the intersection at the rate of λ veh/hr.† Assume further that during the green period, vehicles can depart at a rate of μ veh/hr (which is the intersection's capacity).

The cycle time, c, as seen from the approach analyzed, can be divided into a green period, g, and a red period, r (which can be understood to include amber time and time lost due to startup delays). Obviously, $c = r + g$. Figure 13.6 depicts an ("input/output") schematic diagram of the process of automobile arrival to and departure from a signalized approach. It shows the cumulative arrivals to the intersection and the cumulative departures from it. During the red period the departure rate is zero and vehicles queue in front of the light. When the signal turns green, vehicles in the queue depart at a rate of μ veh/hr. After time g_0 from the start of the green period, the queue clears. At this point the arriving vehicles are departing at a rate of λ veh/hr, which equals the arrival rate, until the light turns red again.

The period required for the queue to dissipate, g_0, is given by (see Problem 13.6)

$$g_0 = r \frac{\lambda/\mu}{1 - \lambda/\mu} \qquad [13.4]$$

†In this section, λ is used instead of q to denote flow, due to the widespread use of this notation in the analysis of queueing systems.

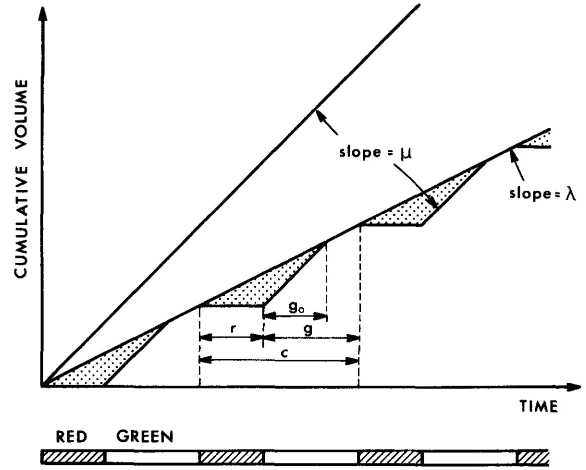


Figure 13.6 Input/output diagrams of the process of vehicle arrivals at and departures from a signalized approach. The figure shows cumulative volumes versus time, assuming constant arrival and departure rates.

The quantity λ/μ is known in queueing theory as the *utilization ratio* and is usually denoted by ρ. For the queue to dissipate in every cycle, it is required that $g_0 < g$, meaning that†

$$\frac{\lambda}{\mu} \leq \frac{g}{c} \qquad [13.5]$$

Given that condition [13.5] holds the total vehicle time of delay, d, in each cycle, equals the area of the triangle (the shaded area in the figure), that is,

$$\begin{aligned} d &= \frac{\lambda r}{2}(r + g_0) \\ &= \frac{\lambda r^2}{2(1-\rho)} \qquad [13.6] \end{aligned}$$

The total number of vehicles per cycle is λc, and thus the average delay per vehicle, $\bar{d}$, is given by

$$\bar{d} = \frac{r^2}{2c(1-\rho)} \qquad [13.7]$$

Formula [13.7] underestimates the average delay because it does not account for stochastic variations in the arrival rate. Even though $\rho \leq g/c$, momentary surges in traffic can cause some vehicles to wait through a full

†If this condition is not met the queue will grow indefinitely and so would the average delay.

cycle. Furthermore, even at the end of the green phase, after the dissipation of the queue, momentary variations in traffic may cause queueing and delay.

These considerations can be accounted for by assuming that in addition to the delay due to queueing during the red phase, the traffic is subject to random delays. Such random delays can be modeled by using queueing theory formulas. Assume that an imaginary queueing system is interposed between the arriving vehicle and the signal. The arrival rate to this queue follows a Poisson process with mean λ veh/hr. The departure rate from this queue into the signal is constant at a rate v veh/hr (i.e., on the average, it takes $1/v$ hours for a single vehicle to depart). Under these assumptions this queueing system can be modeled as an $M/D/1$ queue.† The average delay per customer in such a system is given by

$$\bar{d}' = \frac{R^2}{2\lambda(1 - R)} \qquad [13.8]$$

where R is the utilization ratio for this queue (i.e., $R = \lambda/v$). To express this result in terms of the departure rate from the traffic signal, note that the service rate during the green period is μ veh/hr. During the red period, the departure rate is zero. On the average, then, during the whole cycle the departure rate is $v = \mu(g/c)$ veh/hr. The effective utilization ratio is then

$$R = \frac{\lambda/\mu}{g/c} \qquad [13.9]$$

The two components of the average delay per vehicle ($\bar{d}$ Eqs. [13.7] and $\bar{d}'$ [13.8]) can be summed to give

$$\bar{w} = \frac{r^2}{2c(1 - \rho)} + \frac{R^2}{2\lambda(1 - R)} \qquad [13.10]$$

This formula tends to overestimate the delay by 5 to 15%. Consequently, the transformation $\hat{\bar{w}} = 0.9\bar{w}$ can be used to modify the delay formula in Eq. [13.10] so that $\hat{\bar{w}}$ will be within 5% of the actual delay. The delay formula used by traffic engineers was calibrated by Webster on the basis of Monte Carlo simulations. This formula is

$$\bar{w} = \frac{r^2}{2c(1 - \rho)} + \frac{R^2}{2\lambda(1 - R)} - 0.65\left(\frac{c}{\lambda^2}\right)^{1/3} R^{(2 + 5g/c)} \qquad [13.11]$$

where all times are measured in seconds. The last term in this sum can be looked upon as a correction factor to Eq. [13.10]. Webster's formula gives sufficiently accurate results even in cases in which the arrival rate does not exactly follow a Poisson process.

Figure 13.7 depicts the form of Webster's formula. It shows the delay, in seconds, as a function of the flow for a cycle of 90 seconds and green times of

†A $M/D/1$ queue is a queue with Markovian (Poisson) arrival process, deterministic service rate, and one server.

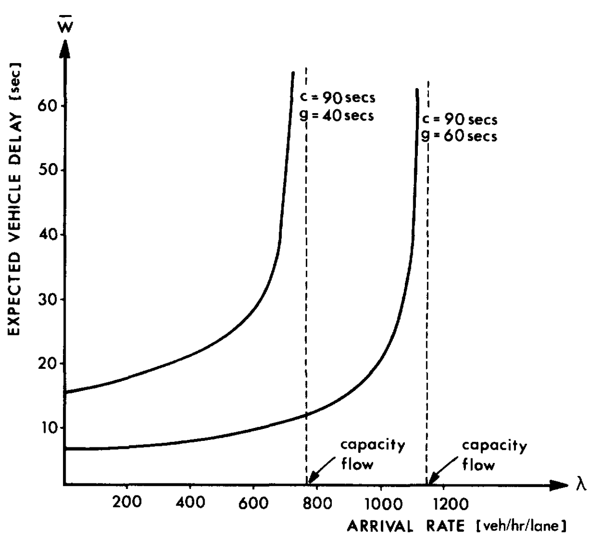


Figure 13.7 Expected vehicle delay at an approach to a signalized intersection as a function of the arrival rates. The curve depicts Webster's formula (Eq. [13.11]) for a cycle of 90 seconds, green times of 40 and 60 seconds, and a departure rate of 2.1 seconds per vehicle.

40 and 60 seconds. The departure time is assumed to be $1/\mu = 2.1$ seconds per vehicle (a value that is typically used in traffic engineering practice). The capacity of the approach [which is given by $\mu(g/c)$] is depicted by the dashed lines corresponding to $g = 40$ and $g = 60$ seconds. As the arriving flow increases and gets close to the approach capacity, the delays grow dramatically, as is the case with any queueing system. The magnitude of this delay dominates the delay through the approach when this curve is added to the on-street delay shown in Figure 13.5. Consequently, the link performance function used in equilibrium analysis is a monotonically increasing function of the flow, similar in shape to the curves shown in Figure 13.7.

Delay at Unsignalized Intersections

As an example of the derivation of a link performance function for approaches to unsignalized intersections, consider the delay on a minor approach of a "T" intersection. The situation is shown in Figure 13.8. The automobile arrivals on the major road (eastbound) are assumed to follow a Poisson process with mean λ veh/hr. The mean flow on the minor (northbound) approach is q veh/hr and it is also assumed to follow a Poisson process. This example assumes that the major road flow has priority over the minor road flow and thus a vehicle on the minor road will move into the intersection only if the gap in traffic (along the major road) is sufficiently large.

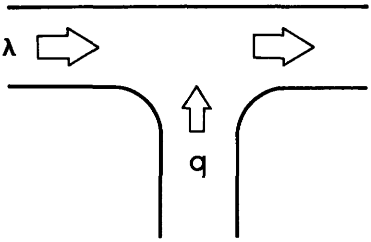


Figure 13.8 Example of a "T" intersection. The flow from the minor direction (q) has to yield to the flow in the major direction (λ).

The accepted gap should be larger than some *critical gap*, which is a fundamental concept in the analysis of unsignalized intersections.

The critical gap is the minimum length of time between two consecutive cars in the major street, which would allow a safe (merging) maneuver for an automobile in the minor street. Assuming that the critical gap, t_{cr}, is the same for all minor approach drivers, a gap will be accepted if its duration is larger than t_{cr}. Let $f_T(t)$ denote the probability density function of the major stream gaps [i.e., $f_T(t)dt = \Pr\,(t \leq T \leq t + dt)$, where T is the duration of a random gap]. The probability that a certain gap, of duration T, will be accepted, P(ACCEPT), is given by

$$P(\text{ACCEPT}) = \Pr\,(T \geq t_{cr}) = \int_{t_{cr}}^{\infty} f_T(t)\,dt \qquad [13.12]$$

Let p denote this probability.

The number of gaps that a car arriving at the minor street approach will have to wait until an acceptable gap will appear is a random variable. The probability that the motorist will have to wait N gaps before an acceptable gap appears is given by

$$P(N) = p(1 - p)^N \qquad [13.13]$$

where p is the acceptance probability given in Eq. [13.12]. Equation [13.13] describes a geometric probability mass function of N. It is the probability of rejecting N gaps and crossing on the $(N + 1)$st one. The expected number of rejected gaps is, then,

$$E[N] = \frac{1 - p}{p} \qquad [13.14]$$

which is the mean of the geometric probability mass function.†

The expected length of the wait is the product of the expected number of rejected gaps and the expected length of a rejected gap (which is not the same as the expected length of a random gap). The distribution of the rejected gaps equals the original distribution of the gaps, conditional on T being smaller

†The geometric probability mass function is usually defined as the probability of "success" in the Nth experiment. Note that Eq. [13.13] defines it here to be the success probability on the $(N + 1)$st experiment.

than t_{cr}. This distribution, $f_{T|T<t_{cr}}(t)$, is given by

$$f_{T|T<t_{cr}}(t) = \frac{f_T(t)}{\int_0^{t_{cr}} f_T(t)\,dt} \qquad [13.15]$$

The expected length of a rejected gap is the mean of this distribution, that is,

$$E[T \mid T < t_{cr}] = \frac{\int_0^{t_{cr}} t f_T(t)\,dt}{\int_0^{t_{cr}} f_T(t)\,dt} \qquad [13.16]$$

The expected waiting time $E[W]$, can now be calculated from Eqs. [13.14] and [13.16] to be†

$$E[W] = E[N]E[T \mid T < t_{cr}]$$

$$= \frac{\int_0^{t_{cr}} f_T(t)\,dt}{\int_{t_{cr}}^{\infty} f_T(t)\,dt} \, \frac{\int_0^{t_{cr}} t f_T(t)\,dt}{\int_0^{t_{cr}} f_T(t)\,dt} \qquad [13.17a]$$

Since the traffic on the major street was assumed to follow a Poisson process, the gap distribution, $f_T(t)$, follows a negative exponential distribution with the same parameter. Using this distribution Eq. [13.17a] can be easily evaluated; the mean wait time for an acceptable gap is given by

$$E[W] = \frac{1}{\lambda} e^{\lambda t_{cr}} - \frac{1}{\lambda} - t_{cr} \qquad [13.17b]$$

This is the expected delay for a car in the minor street, given that it is first in line. The expected delay for an automobile arriving randomly to the intersection can be calculated by using queueing theory concepts.

The queue in the minor street can be characterized, using queueing theory terminology, by a Poisson arrival process (with parameter q), a general distribution of "customer service time" (in this case it is the time that it takes to "process" a single car at the head of the queue, or the waiting time for an acceptable gap), and a single server. This type of queue is known as an $M/G/1$ queue. The mean waiting time for a "customer" in such a queue, $E[D]$, is given by a simple expression including the arrival and service rates. When applied to the case under consideration, the delay for traffic in the minor street will be given by

$$E[D] = \frac{qE[W^2]}{2(1 - qE[W])} + E[W] \qquad [13.18]$$

†Eq. [13.17a] assumes implicitly that the process of car arrivals in the major street is without memory. If this assumption is not made the expected duration of the first gap has to be modified.

In this expression, $E[W]$ is the expected waiting time for an acceptable gap (given in Eq. [13.17b]) and $E[W^2]$ is the second moment of this waiting time. The calculation of this second moment involves lengthy manipulations which are not repeated here. As it turns out,

$$E[W^2] = E[N^2]E[T \mid T < t_{cr}]^2 + E[N] \text{ var } [T \mid T < t_{cr}] \qquad [13.19]$$

The reader is asked (in Problem 13.9) to derive the second moment of W explicitly. Once derived, it can be substituted in Eq. [13.14] to compute the expected delay to the minor approach traffic.

Figure 13.9 depicts the shape of the performance function for an approach to an unsignalized intersection (i.e., $E[D]$ as a function of q) for various values of λ and t_{cr}. As expected, the capacity of the minor approach is smaller and congestion develops faster when λ is higher (i.e., when the flow in the major street is heavier and each (minor approach) driver has to wait longer for an acceptable gap). The same holds true when the value of t_{cr} is higher, in which case, again, each (minor approach) driver has to wait longer for an acceptable gap.

The value of the critical gap, t_{cr}, was assumed in this analysis to be the

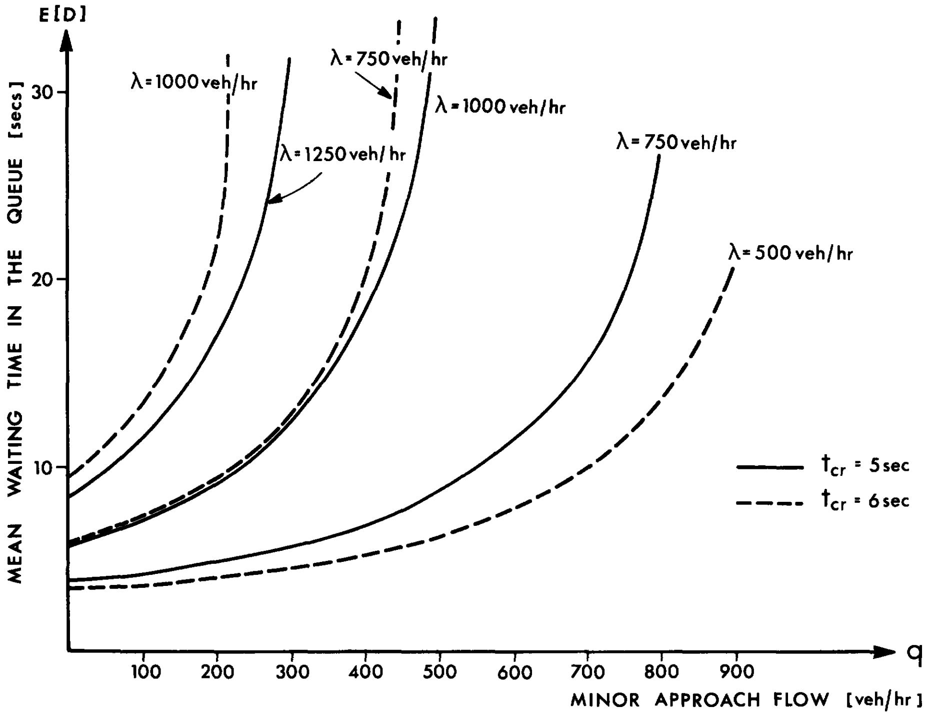


Figure 13.9 Expected delay at an unsignalized intersection as a function of the flow. The curve is drawn for various values of the flow in the major direction (λ) and the critical gap (t_{cr}).

same for all drivers. This can obviously be only an approximation. In order to measure the distribution of t_{cr} among drivers, the gap acceptance situation can be looked upon as a discrete choice problem in which the two choices are to accept a gap (i.e., advance into the intersection) or to wait for the next one. Observations on drivers' gap acceptance and rejection can be used to calibrate such a model, from which the distribution of values of t_{cr} across drivers can be derived. In fact, as shown in the references mentioned in Section 13.5, this approach can even be used to estimate the tendency of drivers to accept shorter gaps the longer they wait.

Common Link Performance Functions

Thus far, the material in this section described the analytical backgroup in traffic flow theory that underlies the development of link performance functions. The formulas used in analyzing urban networks, however, are significantly simpler than the intersection delay equations given above. For example, most studies ignore the interdependence between conflicting flow streams, such as the one developed above for unsignalized intersections. The reason for using simplified formulas is that during the equilibrium calculations the link performance functions have to be evaluated numerous times. A complicated functional form thus imposes a significant computational burden on the calculations.

A simplified function that is often used in practice is the equation developed by the U.S. Bureau of Public Roads (BPR). This equation is given by

$$t_a = t_a^0\left[1 + \alpha\left(\frac{x_a}{c_a'}\right)^{\beta}\right] \qquad [13.20]$$

In this formula, t_a and x_a are the travel time and flow, respectively, on link a, t_a^0 is the free-flow travel time, and c_a' is the "practical capacity" of link a. The quantities α and β are model parameters, for which the value $\alpha = 0.15$ min and $\beta = 4.0$ are typically used. Note that these values imply that the practical capacity of a link is the flow at which the travel time is 15% higher than the free-flow travel time. This does not equal the capacity of the road as defined before (see Figure [13.4] and the related discussion), that is, the maximum possible flow through a link.

In contrast with traffic flow theory the BPR curves are not asymptotic to any capacity value. A function that is asymptotic to a capacity flow was proposed by Davidson, based on queueing theory considerations. This function is

$$t_a = t_a^0\left[1 + J\,\frac{x_a}{c_a - x_a}\right] \qquad [13.21]$$

where c_a is the road's capacity and J is a parameter of the model. As with the BPR function, t_a^0 denotes the free-flow speed (i.e., the speed at zero flow).

Davidson's performance function can be easily estimated from field measurements. The parameter J controls the shape of the curve; its effect is shown in Figure 13.10, which depicts t_a/t_a^0 versus x_a/c_a. The latter quantity, x_a/c_a, is used frequently in traffic engineering and is known as the volume/capacity ratio.

In principle, the travel time through an urban street should reflect the sum of the average running time at a given flow level and the average time spent queueing at intersections, at that flow level. Congestion effects on the street itself can usually be assumed away, when compared to the magnitude of the intersection congestion. Consequently, some fixed travel time can be added to the intersection delay formula in order to use it as a link performance function. Note, for example, that the capacity of an approach to a signalized intersection can be only as high as $c_a(g/c)$, where c_a is the capacity of the road segment leading to the intersection, and (g/c) is the ratio of green time to cycle time for that approach. Consequently, the intersection congestion effects will dominate a link's performance long before the segment capacity will be reached.

As mentioned above, actual studies use simplified performance curves such as the BPR formula, Davidson's formula, or some other, similar expression. The BPR formula is used in many studies since it uses flows that are not restricted by link capacity. This means that the equilibrium algorithm does not have to be adjusted to maintain flow feasibility on all links, a process that can slow the algorithm considerably, with no appreciable change in accuracy.

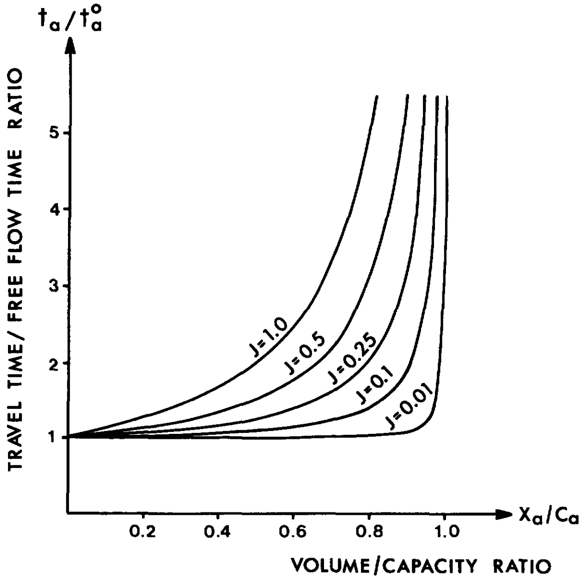


Figure 13.10 Davidson's link performance function; note the effect of the J parameter.

The curves themselves can be estimated directly from observations of flow and delay or from the more detailed functions. Usually, the road network is divided into several types of streets (classified by factors such as width, area type, and traffic regulations) and a performance function is estimated for each type. The methods that can be used to estimate performance function from the data are identical to the method described in the next section for estimating demand functions.

13.2 ESTIMATION OF DEMAND FUNCTION PARAMETERS

As mentioned in the introduction to this chapter, demand functions may be used within the equilibration process or they may be used exogenously to generate an O–D matrix for the traffic assignment model. In any event, such functions have to be estimated, or fitted for each urban area under study.

The demand functions that need to be estimated depend on the scope of the analysis. In some cases, a variable-demand function (such as the one used in Chapter 6) will suffice. In other cases modal split, trip distribution, and trip generation functions have to be estimated. This section describes the considerations underlying some econometric techniques that can be used to estimate these travel demand functions. The first part describes the well-known least-squares method, while the second part describes the principles of maximum likelihood estimation. This exposition leads to the description of the estimation of discrete choice models, in the third part of this section.

Curve Fitting and Least Squares

To demonstrate some of the basic notions of econometric estimation, consider a travel demand function such as $q_{rs} = D(u_{rs})$. This function gives the O–D trip rate, q_{rs}, from the O–D travel time, u_{rs}. In order to estimate this demand function its functional form has to be specified up to a set of parameters whose values are then estimated from data.

The data needed to estimate the model include observations of time (u_{rs}) and trip rate (q_{rs}) for a group of O–D pairs. A simple example of such a data set, including six observations, is shown in Figure 13.11a. These data can be used to estimate the model parameters. The functional form of this model, however, has to be specified first. The most widespread econometric model form is the linear one, in which these observations are used to estimate the parameters ϕ_1 and ϕ_2 of the equation

$$q_{rs} = \phi_1 + \phi_2 u_{rs} \qquad [13.22]$$

The most common criterion used to choose appropriate values for ϕ_1 and ϕ_2 is the minimum sum of squares (or least squares). In other words, the line given by Eq. [13.22] is chosen so that the sum of the squared vertical distances

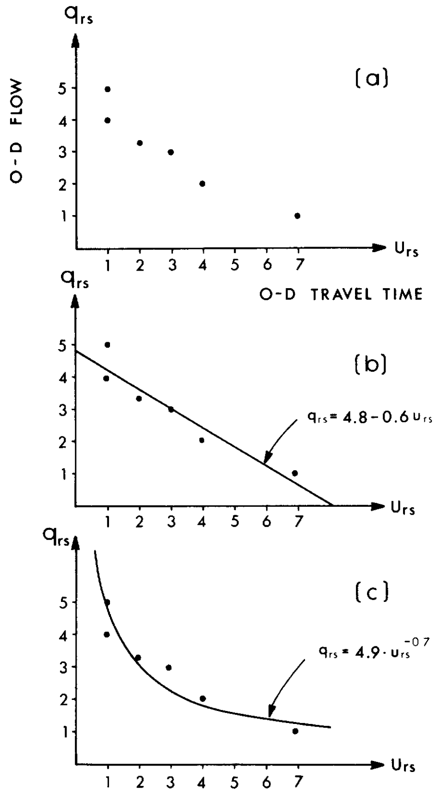


Figure 13.11 Curve-fitting example: (a) the data; (b) the least-squares linear fit; (c) the least-squares logarithmic fit.

between the observations and the line is minimized. Formally, the chosen parameters are the values that minimize the objective function $z(\phi_1, \phi_2)$, where

$$z(\phi_1, \phi_2) = \sum_{rs} (\phi_1 + \phi_2 \tilde{u}_{rs} - \tilde{q}_{rs})^2 \qquad [13.23]$$

and where $\{\tilde{u}_{rs}\}$ and $\{\tilde{q}_{rs}\}$ are the observations included in the data set. The sum in Eq. [13.23] goes over all the observations. The estimated line (which is known as the *regression line*) is shown in Figure 13.11b. As expected, ϕ_2 is negative; higher travel time means less trips in Eq. [13.22].

The same technique can be used for multivariate regression, in which the dependent variable (q_{rs} in this case) is a function of several explanatory ("independent") variables (such as travel time, u_{rs}, origin population, G_r, destination employment, E_s, etc.). The model then might be given by

$$q_{rs} = \phi_1 + \phi_2 u_{rs} + \phi_3 G_r + \phi_4 E_s \qquad [13.24]$$

Again, the set of parameters $(\phi_1, \phi_2, \phi_3, \phi_4)$ can be estimated from the data by using the least-squares method. Each observation (or data point) would include in this case the observed values of all the variables in the model (i.e., $\{\tilde{q}_{rs}\}$, $\{\tilde{u}_{rs}\}$, $\{\tilde{G}_r\}$, and $\{\tilde{E}_s\}$). As mentioned above, linear regression analysis is widely used and computer codes for estimating the parameters of multivariate regression models are readily available.

The line shown in Figure 13.11b may fit the data well. This fit does not mean, however, that the line shown is the correct model of the relationships between q_{rs} and u_{rs}. Theoretical considerations may suggest that the rate at which the travel time affects the O–D flow is not constant. This means that the travel demand model should be nonlinear. An example of such a demand curve is the function

$$q_{rs} = \phi_1 u_{rs}^{\phi_2} \tag{13.25}$$

Again, the parameters of this curve can be estimated by the least-squares method.† The resulting curve is shown in Figure 13.11c. Nonlinear equations can be estimated in the multidimensional case as well, and appropriate computer codes are available (even though it is more difficult to estimate a general nonlinear model than a linear one).

Maximum Likelihood

The statistical properties of the parameters estimated by the least-squares method are not always good. Better estimates can usually be obtained by using the method of maximum likelihood (ML), which provides more statistical information about the parameter estimates.

The idea behind the ML method is very simple. Given the data and a specified model, the estimated parameters are the values that are the most likely to generate the observed data. For example, consider the following set of realizations of a random number: 83, 72, 92, 77. If these were generated from a normal distribution, it is unlikely that the mean of this distribution was zero or 1000. It is more likely that these observations were generated from a distribution with a mean that is close to the observed values. In this case, it is most likely that these numbers were generated from a normal distribution with mean equal to 81 (which is the average of the observations). The mechanics of the ML approach can be explained by means of a simple example.

Consider a process of automobile arrivals at an intersection. In light traffic, these arrivals follow a Poisson process and the headway distribution between successive cars follows a negative exponential distribution. In other words, the density function of the (random) headway, H, is given by

$$f_H(h) = \lambda e^{-\lambda h} \tag{13.26}$$

where λ is a parameter of this distribution.

The data collected in order to estimate λ in a certain intersection approach would typically include N independent headway measurements h_1,

†Alternatively, it can be estimated by using a linear regression if a logarithmic transformation is used first on the data (i.e., the resulting ϕ_1 and ϕ_2 are those that minimize $\sum_{rs} (\ln \phi_1 + \phi_2 \ln \tilde{u}_{rs} - \ln \tilde{q}_{rs})^2$.

$h_2, \ldots, h_N$. The probability density function of a single headway is given by Eq. [13.26], so the joint probability density of observing the entire sample (assuming that λ is known) would be

$$f_{H_1, H_2, \ldots, H_n}(h_1, h_2, \ldots, h_n) = \lambda e^{-\lambda h_1} \lambda e^{-\lambda h_2} \cdots \lambda e^{-\lambda h_N}$$

$$= \prod_{n=1}^{N} \lambda e^{-\lambda h_n} \qquad [13.27]$$

This probability can now be looked upon as a function of λ since in Eq. [13.27] the data are known but λ is not. The resulting function, $L(\lambda)$, is known as the *likelihood function*. For the example under consideration this function is given by

$$L(\lambda) = \prod_{n=1}^{N} \lambda e^{-\lambda h_n} \qquad [13.28]$$

As mentioned before, the ML method is based on maximizing the probability of observing the data. Consequently, the value of λ that maximizes $L(\lambda)$ is the ML estimator for this problem.

Instead of maximizing $L(\lambda)$, it is usually easier to maximize $L'(\lambda) = \log L(\lambda)$, which is known as the log-likelihood function. Since the logarithm is a monotonic transformation, both the likelihood function and its logarithm will have a maximum at the same point. For the example under consideration,

$$L'(\lambda) = \sum_{n=1}^{N} (\log \lambda - \lambda h_n) \qquad [13.29]$$

The maximum of $L'(\lambda)$ is at the point where $dL'(\lambda)/d\lambda = 0$, or

$$\sum_{n=1}^{N} \left(\frac{1}{\lambda} - h_n\right) = 0$$

This equation can be rewritten as

$$\frac{N}{\lambda} - \sum_{n=1}^{N} h_n = 0$$

or

$$\lambda^* = \frac{N}{\sum_{n=1}^{N} h_n} = \frac{1}{\bar{h}} \qquad [13.30]$$

where $\bar{h} = (1/N) \sum_{n=1}^{N} h_n$. Equation [13.30] means that the value of λ which maximizes the likelihood function is the inverse of the sample mean, $\bar{h}$.

This example shows how the sample can be used to estimate the parameter of a negative exponential distribution. The maximum likelihood method is used in many statistical and econometric estimation problems. Interestingly, when the ML method is used to estimate the parameters of a regression line,† the result is identical to the least-squares method discussed at the beginning of this section.

Estimating the Parameters of Discrete Choice Models

Discrete choice models were reviewed in Chapter 10, which focused on the logit and probit models. These models give the share of travelers choosing a certain alternative as a function of the characteristics of all alternatives and of the individuals making the choice. As explained in Chapter 10, the measured (systematic) utility function is a function of the attributes of each alternative (and of the characteristics of the decision maker). As mentioned there, the measured part of the utility function also includes some parameters to be estimated. Expanding on the notation used in Chapter 10, let $V_k(\boldsymbol{\phi}, \mathbf{a}_k)$ denote the systematic utility of the kth alternative ($k \in \mathscr{K}$), where $\mathbf{a}_k$ is a vector of explanatory variables (or attributes) associated with the kth alternative and $\boldsymbol{\phi}$ is a vector of parameters. The choice function of this alternative is then given by $P_k(\boldsymbol{\phi}, \mathbf{a})$, where $\mathbf{a} = (\ldots, \mathbf{a}_k, \ldots)$ is a vector which includes the attributes of all the alternatives in the model. For example, the logit choice function is the following:

$$P_k(\boldsymbol{\phi}, \mathbf{a}) = \frac{e^{V_k(\boldsymbol{\phi}, \mathbf{a}_k)}}{\sum_{l \in \mathscr{K}} e^{V_l(\boldsymbol{\phi}, \mathbf{a}_l)}} \qquad [13.31]$$

$P_k(\boldsymbol{\phi}, \mathbf{a})$ is the probability that alternative k will be chosen if the attributes of the alternatives are characterized by a vector $\mathbf{a}$ and the model parameters are $\boldsymbol{\phi}$.

The data for estimating the parameters of a discrete choice model include a simple random sample of observations (where each observation corresponds to an individual making a choice). Each observation consists of the values of the explanatory variables and the observed choice, which is the dependent variable in this case (the focus is on estimating the probability of these choices). Let the observations be designated by n, where $n = 1, 2, \ldots, N$. Accordingly, denote the systematic utility of the kth alternative of the nth observation by $V_{kn}(\boldsymbol{\phi}, \mathbf{a}_{kn})$, where $\mathbf{a}_{kn}$ includes the set of attributes of the kth alternative and the nth person in the sample. Furthermore, let c_n designate the choice of the nth individual observed in the sample. In other words, $c_n = k$ indicates that the observed choice of the nth individual is alternative k.

The maximum likelihood estimates of the parameters can be obtained by finding those parameter values that maximize the likelihood of the data. The probability of observing an individual, n, with a vector of characteristics $\mathbf{a}_n$ is given by some density function $f_{\mathbf{A}}(\mathbf{a}_n)$, where $\mathbf{a}_n = (\ldots, \mathbf{a}_{nk}, \ldots)$. The likelihood

†In that case the ML method assumes that the regression line is associated with a normally distributed error term which causes the variability in the data.

that the nth person observed in the sample chooses alternative c_n, given $\mathbf{a}_n$, is specified by $P_{c_n}(\boldsymbol{\phi}, \mathbf{a}_n)$. This probability, for the logit model for example, is given by

$$P_{cn}(\boldsymbol{\phi}, \mathbf{a}_n) = \frac{e^{V_k(\boldsymbol{\phi},\, \mathbf{a}_{nk})}}{\sum\limits_{l \in \mathscr{K}} e^{V_l(\boldsymbol{\phi},\, \mathbf{a}_{nl})}} \qquad [13.32]$$

The joint probability density that the observed individual will have a characteristics vector a_n *and* will exhibit the choice of c_n is

$$f(\mathbf{a}_n, c_n) = P_{c_n}(\boldsymbol{\phi}, \mathbf{a}_n) f_{\mathbf{A}}(\mathbf{a}_n) \qquad [13.33]$$

The likelihood of observing the sample is, then, the product (over all n) of these probabilities. The likelihood function is the same expression, but viewed as a function of $\boldsymbol{\phi}$ since given the data, the observed values of $\mathbf{a}_n$ and c_n are, in fact, known for all n. Thus

$$L(\boldsymbol{\phi}) = \prod_{n=1}^{N} P_{c_n}(\boldsymbol{\phi}, \mathbf{a}_n) f_{\mathbf{A}}(\mathbf{a}_n) \qquad [13.34a]$$

and

$$L'(\boldsymbol{\phi}) = \sum_{n=1}^{N} \log P_{c_n}(\boldsymbol{\phi}, \mathbf{a}_n) + \log f_{\mathbf{A}}(\mathbf{a}_n) \qquad [13.34b]$$

The values of $\boldsymbol{\phi}$ that maximize this log-likelihood function are the ML estimates. Note that the second term on the right-hand side of Eq. [13.34b] is not a function of $\boldsymbol{\phi}$. Consequently, it can be dropped from the log-likelihood function without affecting the maximization.

The maximization of the log-likelihood function can be carried out by using any of the methods discussed in Chapter 4. For the log-likelihood function to have a unique maximum, it is sufficient to show that the function is concave. Unfortunately, it is proven to be concave only for logit models in which the utility functions are specified to be linear in the parameters, that is, if in Eq. [13.31] $V_k(\boldsymbol{\phi}, \mathbf{a}_k) = \boldsymbol{\phi} \cdot \mathbf{a}_k^T$. Concavity is also proven for certain probit models in which this condition holds and, in addition, the terms in the covariance matrix are not a function of $\mathbf{a}$. This property is the reason for the widespread usage of linear specifications of the utility functions of discrete choice models.

A simple example can illustrate the maximum likelihood estimation of the parameters of a discrete choice model. Consider a logit model of mode choice between automobile and transit. To formulate the model, let V_{au} and V_{tr} denote the systematic utility of the automobile and transit alternative, respectively. Also, let t and $\hat{t}$ denote the respective travel times on the two modes. The specification of the model under consideration is

$$V_{\text{au}} = -\phi t \qquad [13.35a]$$

$$V_{\text{tr}} = -\phi \hat{t} \qquad [13.35b]$$

TABLE 13.1 **Data for Estimating a Mode Choice Model**

Observation Number	Observed Travel-Time Difference, $\hat{t} - t$ (min)	Observed Choice
1	3.0	Automobile
2	0.5	Transit
3	−2.0	Transit
4	1.0	Automobile

where ϕ is a parameter of the model. The choice function for transit in this case, $P_{\text{tr}}(\phi, t, \hat{t})$, is given by

$$P_{\text{tr}}(\phi, t, \hat{t}) = \frac{e^{-\phi\hat{t}}}{e^{-\phi\hat{t}} + e^{-\phi t}} = \frac{1}{1 + e^{\phi(\hat{t}-t)}} \qquad [13.36]$$

The automobile choice function, $P_{\text{au}}(\phi, t, \hat{t}) = 1 - P_{\text{tr}}(\phi, t, \hat{t})$.

The data for this problem include four observations of travel-time differences and choices, as shown in Table 13.1. The log-likelihood function for this problem is then

$$L(\phi) = \ln\left(1 - \frac{1}{1 + e^{3\phi}}\right) + \ln\frac{1}{1 + e^{0.5\phi}} + \ln\frac{1}{1 + e^{-2\phi}} + \ln\left(1 - \frac{1}{1 + e^{\phi}}\right)$$

Figure 13.12 depicts the values of the log-likelihood versus ϕ for this example. As can be seen from the figure, $L(\phi)$ has a maximum for $\phi^* = 1.44$. This value, then, is the ML estimate of ϕ for this problem. It can be used to predict the choice probability for values of $(\hat{t} - t)$ for which the choice is not observed (see Eq. [13.36]). Note that the values of the log-likelihood function itself are all negative, as would be the case for any other discrete choice model. (Why?)

Another type of estimation problem arises when the parameter of a stochastic network loading model needs to be estimated. (Using the notation of Chapter 11 this parameter may be θ for logit-based models or β for probit-based models.) This problem is similar to the estimation of any other discrete choice model parameter, but in this case it is not practical to use the maximum likelihood method described in this section. The reason is that the alternatives to be chosen include all the paths between each O–D pair. The number of these alternatives on even a modest-size network is extremely large, meaning that they cannot be enumerated and serve as a basis for an estimation procedure. Furthermore, in practice, the data usually comprise observed link flows rather than observed choices of individual motorists. An estimation technique that can be used in this case is a variant of the least-squares method mentioned in the first part of Section 13.2. The objective would be to minimize

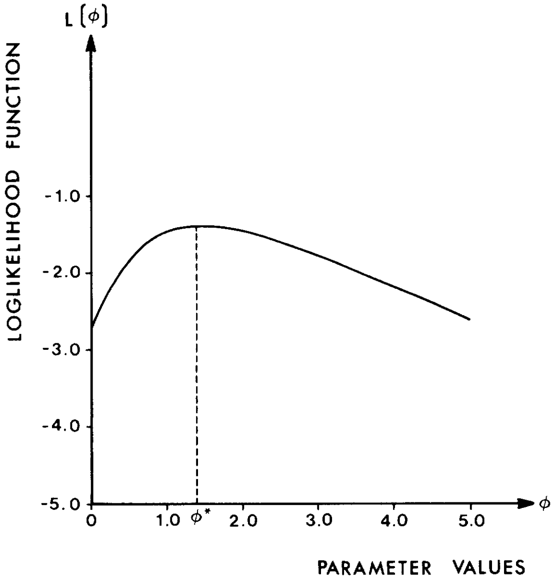


Figure 13.12 The log-likelihood function and its maximum for the binary logit example of modal split discussed in the text.

the function

$$z(\phi) = \sum_a (x_a(\phi) - \tilde{x}_a)^2 \qquad [13.37]$$

where $\{\tilde{x}_a\}$ is the set of observed network flows, and $\{x_a(\phi)\}$ is the set of flow variables generated by a network loading model with parameter ϕ, which is to be estimated. The objective function in Eq. [13.37], $z(\phi)$, can be minimized by using any line-search method. Such a line search is an iterative procedure in which a flow pattern, $\{x_a(\phi)\}$, is generated at each iteration. The procedure generating this flow pattern does not involve a stochastic *equilibration* problem but rather a stochastic network *loading* problem (with parameter ϕ); the link travel times used in the calibration process are the observed (measured) values.

13.3 ESTIMATION OF O-D MATRICES

The travel demand data needed for a complete equilibrium analysis can be collected by conducting a home interview survey. Typically, each interview will include a detailed account of all trips made by each member of a household. The data can then be used to estimate travel demand models such as trip generation, modal split, and trip distribution equations. These models are used, in turn, to generate the O-D matrix, which is the input to a traffic assignment process.

The data collected this way can be used to estimate the demand functions required for a joint analysis of several travel choice dimensions, as described in Part III. In many cases, however, the focus of the analysis is on traffic assignment with fixed O–D matrix. This type of analysis is applicable when the effects of the analyzed scenarios on the O–D matrix itself are negligible or when these effects are independent of the equilibrium travel times and can be calculated separately. In these cases, the O–D matrix can be estimated directly from link flows, without first estimating a series of travel demand models. This possibility may be very attractive since the cost of a home interview survey may reach millions of dollars, while the data needed for direct estimation may be readily available in many urban areas.

The criterion used to estimate the O–D matrix is the ability of the chosen matrix to reproduce the observed conditions. The solution of that problem, however, is not unique; there are many possible O–D matrices that satisfy this criterion. In other words, usually there are many O–D matrices which, when assigned to the network, can reproduce the observed conditions. The second part of this section then looks at the problem of choosing the appropriate O–D matrix among all those that satisfy the first criterion. The focus of this section is on the estimation of O–D matrices for the standard UE assignment. Some extensions are suggested in Problems 13.14 and 13.15.

O–D Matrices That Reproduce Observed Conditions

As mentioned above, the first problem in estimating O–D matrices is to make sure that the estimated matrix, when assigned to the network, reproduces the observed conditions. The procedure described here assumes that the observed network is at equilibrium. In other words, the observed link flows and travel times represent the (observed) UE conditions. The objective of the solution approach is to find an O–D matrix such that when this matrix is assigned to the network (using any UE method), the resulting O–D travel times will be equal to the observed O–D travel times. The solution method is based on the equivalent minimization concept used throughout this book.

To describe the method, let $\tilde{u}_{rs}$ denote the observed travel time between origin r and destination s. Now consider the following minimization program:

$$\min z(\mathbf{x}, \mathbf{q}) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega - \sum_{rs} \tilde{u}_{rs} q_{rs} \qquad [13.38a]$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\ r, s \qquad [13.38b]$$

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \qquad [13.38c]$$

The link–path incidence relationships are also assumed to hold, as usual. The first-order conditions for a minimum of this program can be calculated from the derivative of the Lagrangian of program [13.38] with respect to the path flows and with respect to the O–D flow. The derivative of the Lagrangian with respect to f_k^{rs} is $(c_k^{rs} - u_{rs})$, where u_{rs} is the dual variable associated with the flow conservation constraint (Eq. [13.38b]). The associated first-order conditions are therefore

$$(c_k^{rs} - u_{rs})f_k^{rs} = 0 \qquad \forall\ k, r, s \tag{13.39a}$$

$$c_k^{rs} - u_{rs} \geq 0 \qquad \forall\ k, r, s \tag{13.39b}$$

The second type of first-order conditions is based on the derivative of the Lagrangian of [13.38] with respect to q_{rs}. This derivative is equal to $(-\tilde{u}_{rs} + u_{rs})$, meaning that the associated first-order conditions are

$$\tilde{u}_{rs} = u_{rs} \qquad \forall\ r, s \tag{13.39c}$$

The last of the first-order conditions are, of course, the original constraints, that is,

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall\ r, s \tag{13.39d}$$

$$f_k^{rs} \geq 0 \qquad \forall\ k, r, s \tag{13.39e}$$

The first-order conditions (Eqs. [13.39]) mean that the solution of program [13.38] is a set of O–D trip rates, which when assigned to the network, result in a special UE flow pattern. This UE solution is associated with O–D travel times which are equal to the observed O–D times.

Program [13.38] can be solved by using the convex combinations algorithm. The application of this algorithm to this program is very similar to the application of this algorithm to the solution of the variable-demand program analyzed in Section 6.1. As described there, an upper bound constraint such as

$$q_{rs} \leq \bar{q}_{rs} \qquad \forall\ r, s \tag{13.40}$$

has to be added to the program. The direction-finding step of the algorithm includes the calculation of the minimum path travel time between every O–D pair (at the current flow level). If the travel time on that path at the nth iteration, u_{rs}^n, is larger than the observed time, $\tilde{u}_{rs}$, then the auxiliary O–D flow variable (corresponding to q_{rs}^n), v_{rs}^n, is set to zero. If on the other hand, $u_{rs}^n < \tilde{u}_{rs}$, then v_{rs}^n is set to $\bar{q}_{rs}$. The condition $u_{rs}^n = \tilde{u}_{rs}$ indicates convergence.

Choosing the Appropriate Matrix

Unfortunately, program [13.38] is not strictly convex in $\{q_{rs}\}$. [Note that $\partial^2 z(\mathbf{x}, \mathbf{q})/\partial q_{rs}^2 = 0$, $\forall\ r, s$.] Therefore, it does not have a unique solution in terms of the O–D variables $\{q_{rs}\}$. This nonuniqueness is not surprising since the underlying problem, that of finding an O–D matrix that produces the observed link flows, is underspecified. In other words, there are many O–D matrices that can produce a given set of observed travel times.† The choice between these matrices can be based on several criteria. The criterion assumed here is the closeness of the O–D matrix to some *target matrix.*

The target matrix may be available from past studies in the same area, from a crude demand analysis, or from a limited set of direct ("mailback") questionnaires. In each of these cases the target O–D matrix is not sufficiently accurate to serve as the matrix upon which the study can proceed. The methodology described in this section can be viewed as a method for enhancing the target matrix by using observed conditions. The criterion, then, is to choose the O–D matrix that is closest to the target matrix, yet when assigned to the network reproduces the observed O–D travel times.‡

Let η^* denote the optimal (minimum) value of objective function [13.38a]. Any of the optimal solutions of program [13.38] satisfies constraints [13.38b] and [13.38c] as well as

$$\sum_a \int_0^{x_a^*} t_a(\omega)\, d\omega - \sum_{rs} \tilde{u}_{rs} q_{rs}^* = \eta^* \qquad [13.41]$$

In practice, η^* is calculated by solving Eq. [13.38] with the use of an iterative descent method. The true minimum value of objective function [13.38a] will always be less than η^*, meaning that constraint [13.41] should be expressed as $\leq \eta^*$.

Let $\{\tilde{q}_{rs}\}$ represent the entries of the target O–D matrix. The objective function of the problem is then to choose a set of O–D flows, $\{q_{rs}\}$, which is closest to $\{\tilde{q}_{rs}\}$ yet minimizes program [13.38]. This problem can be formulated as the following minimization program:

$$\min z(\mathbf{x}, \mathbf{q}) = \sum_{rs} (q_{rs} - \tilde{q}_{rs})^2 \qquad [13.42a]$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \quad \forall\, r, s \qquad [13.42b]$$

$$f_k^{rs} \geq 0 \quad \forall\, k, r, s \qquad [13.42c]$$

$$\sum_a \int_0^{x_a} t_a(\omega)\, d\omega - \sum_{rs} \tilde{u}_{rs} q_{rs} \leq \eta^* \qquad [13.42d]$$

†One trivial solution to this problem would be to let r be the upstream node and s be the downstream node of each network link a. The O–D trip rates would then be $q_{rs} = \tilde{x}_a$, where $\tilde{x}_a$ is the observed flow on link a (leading from node r to node s). Such an O–D matrix may be completely erroneous, yet it satisfies the criterion of reproducing the observed conditions. This "solution" highlights the need for selecting the best O–D matrix from among all the possible ones. (Note that in actual cases the particular trivial solution mentioned here will not be feasible since the set of origins and destinations is a small subset of the set of all nodes.)

‡This statement has to be understood as applicable within some convergence criterion set for program [13.38]. In other words, this program is solved by using an iterative descent method and thus its first order conditions (which require that the observed travel times will be reproduced) are met only within some convergence tolerance, rather than exactly.

The objective function in this program minimizes the difference between the O–D trip rates and their target values. Constraints [13.42b] and [13.42c] ensure that the solution is feasible in terms of program [13.38], while constraint [13.42d] ensures that the O–D trip rates which solve program [13.42] are *optimal solutions* of program [13.38].

Program [13.42] can be solved by using the method of partial dualization.† This method is based on forming the (partial) Lagrangian of a given mathematical program, by adding only some of the constraints to the objective function. In this case, let γ denote the dual variable associated with the (single) constraint [13.42d]. The partial Lagrangian with respect to this constraint is

$$\min L(\mathbf{x}, \mathbf{q}, \gamma) = \sum_{rs} (q_{rs} - \tilde{q}_{rs})^2 + \gamma\left[\sum_a \int_0^{x_a} t_a(\omega)\, d\omega - \sum_{rs} \tilde{u}_{rs} q_{rs} - \eta^*\right] \qquad [13.43a]$$

subject to

$$\sum f_k^{rs} = q_{rs} \quad \forall\, r, s \qquad [13.43b]$$

$$f_k^{rs} \geq 0 \quad \forall\, k, r, s \qquad [13.43c]$$

$$\gamma \geq 0 \qquad [13.43d]$$

Constraint [13.43d] is added since Eq. [13.42d] is not an equality constraint and γ is therefore restricted to be nonnegative.‡

The minimum of the original program is at the stationary point of the Lagrangian, which is (as mentioned in Chapter 2) a saddle point. The stationary point is at the *minimum* of $L(\mathbf{x}, \mathbf{q}, \gamma)$ with respect to $\mathbf{x}$ and $\mathbf{q}$ but at the *maximum* of $L(\mathbf{x}, \mathbf{q}, \gamma)$ with respect to γ.

To solve this program, assume that the value of γ is given. In this case, Lagrangian [13.43a] can be minimized with respect to $\mathbf{x}$ and $\mathbf{q}$ (subject to constraints [13.43b] through [13.43d]) by using the convex combinations algorithm or any other suitable method. This is no more difficult than solving program [13.38]. Let the optimal value of Lagrangian [13.43a], given γ, be denoted by $L(\gamma)$ [i.e., $L(\gamma) = L(\mathbf{x}^*, \mathbf{q}^*, \gamma)$]. The correct value of γ that should be used in program [13.43] to minimize $L(\mathbf{x}, \mathbf{q}, \gamma)$ is γ^* [i.e., the value of γ that maximizes $L(\gamma)$]. At this point $L(\gamma^*) = L(\mathbf{x}^*, \mathbf{q}^*, \gamma^*)$ is a saddle point, as required. The maximization of $L(\gamma)$ is a one-dimensional maximization problem that can be accomplished by any of the relevant line-search methods. In particular, a method based on the derivative of $L(\gamma)$ with respect to γ can be used.

†The underlying theory of partial dualization methods is beyond the scope of this text. The application of the method to this problem is, however, quite simple and may be intuitive.

‡Note that both the inequality sign in Eq. [13.42d] and the last term in Eq. [13.43a] have opposite signs as compared to the notational convention used in Chapter 2.

This derivative is given by

$$\frac{dL(\lambda)}{d\lambda} = \sum_a \int_0^{x_a^m(\gamma)} t_a(x_a) - \sum_{rs} \tilde{u}_{rs}\, q_{rs}^m(\lambda) - \eta^* \qquad [13.44]$$

where $\{x_a^m(\gamma)\}$ and $\{q_{rs}^m(\gamma)\}$ denote the optimal solution of the Lagrangian problem [13.43] at the *m*th iteration of the line search.

Note that since η^* denotes the (almost) optimal value of the objective function of program [13.38], $dL(\gamma)/d\gamma$ in Eq. [13.44] will be positive for small values of γ. The reason is that the problem of minimizing Lagrangian [13.43] (given γ) involves striking a balance between minimizing $\sum_{rs} (q_{rs} - \tilde{q}_{rs})^2$ and minimizing $\sum_a \int_0^{x_a} t_a(\omega)\, d\omega - \sum_{rs} \tilde{u}_{rs} q_{rs}$, where the balancing factor is γ. For very small values of γ the emphasis of the Lagrangian minimization will be on the first term and the solution will involve an O–D matrix (i.e., a set $\{q_{rs}\}$) which is very close to the target O–D matrix. This solution may be far from optimal in terms of program [13.38] (hence $dL(\gamma)/d\gamma$ will be positive). If, however, γ is very large, the solution of the Lagrangian minimization problem (given γ) will be very close to the solution of program [13.38]. It may, however, be further from the target O–D matrix.

The solution procedure here, then, is to start with a low value for γ and solve the Lagrangian program [13.43]. The value of γ is then increased and program [13.43] solved again. The process continues in this fashion until $dL(\gamma)/d\gamma$ is small enough (or negative). The series of solutions of program [13.43] (given γ) which is generated in this process will start from one that is very close to the target O–D matrix. As γ is increased, more and more weight is given to having a solution (in terms of an O–D matrix) that when assigned to the network will generate O–D travel times that are close to the observed ones. The process will terminate when the smallest value of γ for which $dL(\gamma)/d\gamma \leq k$ (where k is a positive convergence measure) is found. At this point, the O–D matrix is closest to the target O–D matrix of any of the matrices that solve program [13.38] (having a solution better than $\eta^* + k$). In other words, it is closest to the target matrix among all those matrices that, when assigned to the network, reproduce the observed O–D travel times.

13.4 SUMMARY

This chapter describes some of the background necessary to derive the inputs for a network equilibrium analysis of an urban network. The first section focused on the derivation of link performance functions.

The time spent traveling along a network link includes both the "running" time along the road segment represented by the link, and the time spent queueing at intersections. These two components of the performance functions can be modeled by using traffic flow theory to model the running time as a function of the link flow, and queueing theory to model intersection delays. At signalized intersections the delay will be influenced by the signal timing, while at unsignalized intersections, it is influenced by the cross-street flows. These derivations generate complicated formulas which, for the most part, are not used in practice. In applications, these functions are usually approximated by BPR-type (or other simple) curves.

Section 13.2 deals with the estimation of travel demand functions. It reviews the least-squares method for estimating the parameters of both linear and nonlinear regression curves. Then the maximum likelihood method is reviewed with particular emphasis on the application of this method to the estimation of discrete choice models. It is interesting to note that the ML method can be applied to such models, in which the dependent variable does not have a numerical value. (Instead, it represents a choice.) These demand functions can be used as part of a global equilibrium analysis, or they can be used to generate an O–D matrix for a standard (fixed-demand) equilibrium assignment.

Another approach to the derivation of O–D matrices is outlined in Section 13.3. It describes a direct estimation method in which the O–D entries are developed from observed O–D travel times and a given target O–D matrix. The estimated O–D matrix is the closest possible to the target O–D matrix, yet it closely reproduces the observed O–D travel times.

13.5 ANNOTATED REFERENCES

The material of traffic flow theory given in the first part of Section 13.1 can be found in any traffic engineering book, such as Pignataro (1973). More advanced treatments of the subject are offered by Drew (1968) and Haberman (1977). The speed/density line mentioned in conjunction with Eq. [13.2] was estimated by Greenshields (1934). The race track example mentioned in Section 13.1 was suggested by Daganzo. The material on delay at signalized intersections can be found in Homburger (1982), which is based on the works of May (1965), Webster (1958), Allsop (1972), and others. The problem of delay at unsignalized intersections was investigated by Weiss and Maradudin (1962) and Herman and Weiss (1961), who found delay expressions based on assumptions that were more general than the ones used in Section 13.1. The estimation of the critical gap by using probit models was investigated by many researchers, including Miller (1972), Daganzo (1981), and Mahmassani and Sheffi (1981).

The BPR curves were suggested by the U.S. Bureau of Public Roads (1964), while Davidson (1966) based the derivation of his performance function on queueing theory. Taylor (1977) described a simple technique for estimating the parameters of Davidson's curve under various conditions. Branston (1976) surveyed many of the performance functions used in practice, including the practice of approximating complex functions with BPR-type curves and other simple functions. Such approximations were used, for example, by Florian and

Nguyen (1978) in the study of the city of Winnipeg. In several other studies, the performance functions were correlated directly with link characteristics, such as speed limit, frequency of intersections along the link, and the urbanization level. Such relationships were used, for example, in the Metropolitan Toronto Regional Transportation Study (1967) and by Freeman Fox and Associates (1972).

The econometric issues mentioned in Section 13.2 can be found in any statistics or econometric textbook. The application of the maximum likelihood method to the calibration of discrete choice models is now a standard technique. The theory and the methods used in this context are described in the texts by Daganzo (1979), Hensher and Johnson (1981), and Kanafani (1983).

The method for estimating O–D matrices from link data described in Section 13.3 is based on the work of LeBlanc and Farhangian (1982). Their work, in turn, follows Nguyen (1977) who suggested the use of program [13.38] for deriving O–D matrices which, when assigned to the network, generates the observed O–D travel times. Other researchers suggested different criteria for choosing among all the O–D matrices that reproduce the observed link flows. For example, Jornsten and Nguyen (1980) suggested selecting the O–D matrix with the greatest entropy. The partial dualization method used in that section is used quite often in large-scale mathematical programming problems. The related theory and algorithmic approaches are described, for example, in the textbook by Lasdon (1970).

13.6 PROBLEMS

13.1. Show that the fundamental relationships of traffic flow [Eq. 13.1] hold. Use a simple traffic stream at constant speed to develop your argument. Why is the space mean speed used in these relationships?

13.2. **(a)** Consider a set of speed measurements $u_1, u_2, \ldots, u_n$ of n vehicles passing a stationary observer during some period, T. Show that the space mean speed is given by

$$\bar{u}_s = \frac{1}{(1/n)\sum_{i=1}^{n}(1/u_i)}$$

What is the time mean speed?

(b) Show that $\bar{u}_s \leq \bar{u}_t$.

13.3. The speed–density relationship for a certain road segment is

$$\bar{u}_s = \begin{cases} u_f & \text{if } K \leq K_0 \\ u_f \dfrac{K_j - K}{K_j - K_0} & \text{if } K_0 < K < K_j \\ 0 & \text{if } K \geq K_j \end{cases}$$

where K is the density, K_j is the jam density, and K_0 is the maximum unconjested density. Plot the fundamental diagram (i.e., flow versus density) for this segment. Find the road's capacity and the speed at capacity flow.

13.4. Develop the expression for the road's capacity, the density at which flow equals capacity, and the speed at that point, for the linear model in Eq. [13.2].

13.5. Some driving instructors advocate the "2-second rule," encouraging drivers to keep at least 2 seconds behind the car in front. Assuming that all drivers follow this advice, the spacing between cars is given by

$$\Delta = l + t\bar{u}_s$$

where l is car length in meters, $t = 2$ seconds, and $\bar{u}_s$ is the space mean speed (in meters per second). Based on this, find **(a)** the capacity of the road (in veh/hr), **(b)** the speed at capacity (in km/hr), and **(c)** the jam density (in veh/km). (*Hint:* The spacing is the inverse of the density.) **(d)** Are your results realistic? Explain.

13.6. **(a)** Verify the expression for g_0 in Eq. [13.4].
(b) Explain why the total delay in a deterministic system is given by the area of the triangle in Figure 13.6. Then derive the delay expression in Eq. [13.6].

13.7. The service rate of a given toll booth is 60 veh/min. Suppose that the arrival rate to this booth is 80 veh/min between 5 and 6 P.M. and 40 veh/min thereafter. Assuming that no queue exists before 5 P.M., use a deterministic input/output diagram to answer the following questions:
(a) When does the queue start, and when does it dissipate?
(b) What is the maximum queue length, and when does it happen?
(c) What is the maximum waiting time and when does it occur?
(d) What is the total delay?
(e) What is the average delay during the rush period (which is defined as the time in which there is a queue in front of the booth)?
(f) What is the average delay as a function of the arrival rate?

13.8. A traffic light is planned for the intersection shown in Figure 13.8, where the flows are $\lambda = 1000$ veh/hr and $q = 500$ veh/hr. The green splits are set in direct proportion to the flow, thus $g_1/g_2 = \lambda/q$, where g_1 is the green split on the major road and g_2 is the green split on the minor one. The problem is to set the cycle length, c, so that the total delay is minimized. Assume further that 10 seconds are taken from each cycle for pedestrian crossing, amber periods, and "all-red" period (the sum of these times is known as the lost cycle time by traffic engineers), and that it takes 2.1 seconds for each car to go through the light. Using Webster's formula, plot the total delay (on both approaches) as a function of the cycle length and determine the optimal cycle length. Explain your considerations.

***13.9.** Derive explicitly the expression for $E[W^2]$ under the Poisson assumptions used in conjunction with Eq. [13.19]. Then develop explicitly the expression for the mean delay in the minor approach queue, using Eq. [13.18].

13.10. Fit a BPR curve and a Davidson curve to Webster's delay formula for $c = 90$ seconds, $g = 50$ seconds, $1/\mu = 2.1$ seconds, and $r = 35$ seconds. Explain your considerations and compare (graphically) the fit of both models (in a plot of the delay versus λ).

13.11. Consider the process of maximum likelihood estimation of a discrete-choice model.

(a) Why are the log-likelihood values negative?

(b) Can you think of a goodness-of-fit measure that can be used to characterize the "quality of fit" of a discrete choice model?

13.12. **(a)** Explain in detail the direction-finding step involved in solving program [13.38] by using the convex combinations method.

(b) Repeat part (a) for program [13.43].

13.13. Derive and explain the first-order conditions of program [13.43].

13.14. Analyze the possibility of using the O–D matrix estimation technique explained in Section 13.3 for a model of joint mode split and traffic assignment. Discuss the possibility of a joint estimation of the O–D volumes and the parameter of the modal split function. Use the supernetwork approach in your discussion.

13.15. Discuss the applicability of the O–D matrix estimation method of Section 13.3 to SUE. In this case, the stochastic network loading parameter has to be estimated as well, as part of the process.

Appendix: Normal Distribution and Probit Analysis

This appendix reviews some of the characteristics of multivariate normal distribution functions which are important for probit analysis. It also reviews two of the methods mentioned in Chapter 10 for evaluating the probit choice function: Clark's numerical approximation and the Monte Carlo simulation method.

A.1 MULTIVARIATE NORMAL DISTRIBUTION

The multivariate normal (MVN) density function can be characterized by a K-vector of means $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$ and a (nonsingular, symmetric, positive definite), $K \times K$ covariance matrix, $\boldsymbol{\Sigma}$, where

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1K} \\ \sigma_{21} & & & \\ \vdots & & & \vdots \\ \sigma_{K1} & & \cdots & \sigma_K^2 \end{pmatrix}$$

The covariance matrix of a multivariate normal distribution is often expressed in a standard form as the correlation matrix, $\boldsymbol{\rho}$. The entries of this matrix are given by $\rho_{kl} = \sigma_{kl}^2/\sigma_k \sigma_l$, $\forall\ k, l$ where ρ_{kl} is known as the correlation coefficient.

A random K-vector, $\mathbf{X}$, is MVN distributed if its density function $f_{\mathbf{X}}(\mathbf{x})$ is given by

$$f_{\mathbf{X}}(\mathbf{x}) = (2\pi |\boldsymbol{\Sigma}|)^{-K/2} \exp\left[-\tfrac{1}{2}(\mathbf{x} - \boldsymbol{\mu}) \cdot \boldsymbol{\Sigma}^{-1} \cdot (\mathbf{x} - \boldsymbol{\mu})^T\right] \qquad \text{[A.1]}$$

This statement is usually abbreviated by $\mathbf{X} \sim \text{MVN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} = E[\mathbf{X}]$ and $\boldsymbol{\Sigma} = \text{cov}\,[\mathbf{X}]$. The diagonal entries of $\boldsymbol{\Sigma}$ are the variances of the elements of $\mathbf{X}$, while the off-diagonal elements express the covariance between these elements.

An important property of the MVN distribution is that it is closed under linear transformation. This property means that a linear transformation of a normally distributed random vector generates a new normally distributed random vector. To see the mechanics of the transformation, let $\mathbf{Y} = (Y_1, \ldots, Y_L)$ be a random L-vector which is related to $\mathbf{X}$ by a linear transformation. In other words,

$$\mathbf{Y} = \mathbf{X} \cdot \mathbf{m} + \mathbf{b} \qquad [A.2]$$

where $\mathbf{m}$ is a $(K \times L)$ matrix of constants and $\mathbf{b}$ is an L-vector of constants. The vector $\mathbf{Y}$ is a random variable. The mean vector of $\mathbf{Y}$, $E[\mathbf{Y}]$, is given by

$$E[\mathbf{Y}] = \boldsymbol{\mu} \cdot \mathbf{m} + \mathbf{b} \qquad [A.3]$$

where $\boldsymbol{\mu} = E[\mathbf{X}]$. The covariance matrix of $\mathbf{Y}$, cov $[\mathbf{Y}]$, is given by

$$\text{cov}\,[\mathbf{Y}] = \mathbf{m}^T \cdot \boldsymbol{\Sigma} \cdot \mathbf{m} \qquad [A.4]$$

where $\boldsymbol{\Sigma} = \text{cov}\,[\mathbf{X}]$. Note that $E[\mathbf{Y}]$ is an L-vector and cov $[\mathbf{Y}]$ is an $L \times L$ matrix, as expected.

If X is a scalar, it is normally distributed if its density function, $f_X(x)$, is given by

$$f_X(x) = \frac{1}{2\pi\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \qquad [A.5]$$

where μ and σ^2 are the mean and variance, respectively, of X. The linear transformation

$$Z = \frac{1}{\sigma} X - \frac{\mu}{\sigma} \qquad [A.6]$$

yields a normally distributed random variable Z, with mean zero and variance equal to one. The variable Z is known as a *standard normal* variate.

The cumulative distribution function of a normal variate cannot be expressed in a closed form. Using the transformation in Eq. [A.6], this cumulative distribution can be expressed in terms of the distribution of a standard normal variate, known as the standard normal curve:

$$\Phi(z) = \int_{-\infty}^{z} \frac{1}{2\pi} \exp\left(\frac{-\omega^2}{2}\right) d\omega$$

where z is the value that the random variable Z can obtain in a particular experiment. The values of $\Phi(z)$ for various values of $-\infty < z < \infty$ have been tabulated and are readily available.

A.2 MULTINOMIAL PROBIT MODEL

Assume, as in Section 10.1, that a decision maker is faced with a set, $\mathscr{K}$, of alternatives numbered 1, 2, ..., K. Let the utility of the kth alternative be denoted by U_k. Assume further that

$$U_k = V_k + \xi_k \qquad \forall\, k \in \mathscr{K} \qquad [A.7]$$

where V_k is the systematic utility of the kth alternative and ξ_k is the random disturbance term associated with that alternative. The assumption leading to the probit model is that the vector $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_K)$ is MVN distributed with mean vector zero and covariance matrix $\boldsymbol{\Sigma}$.

The probability that the kth alternative is chosen, P_k, is given by

$$P_k = \Pr\,(U_k \geq U_l, \forall\, l \in \mathscr{K}) \qquad [A.8]$$

Substituting Eq. [A.7], the choice probability becomes

$$P_k = \Pr\,(\xi_l \leq V_k - V_l + \xi_k, \forall\, l \in \mathscr{K}) \qquad [A.9]$$

This expression involves the calculation of the cumulative MVN distribution function, which cannot be evaluated in closed form; it involves a multiple integral (of dimension $K - 1$) which is difficult to evaluate by standard numerical methods (especially for large values of K). The remainder of this appendix describes the two methods mentioned in Section 10.1 for evaluating the probit choice probability. These are the analytical approximation method and the Monte Carlo simulation method.

Approximation Method

The formulas upon which the approximation method is based were suggested by Clark (see Section 10.1) for the approximation of the first two moments of the distribution of max $(U_1, \ldots, U_K)$, where the K random variables follow a joint normal density.

Let U_1, U_2, and U_3 be MVN distributed with means V_1, V_2, and V_3, variances σ_1^2, σ_2^2, and σ_3^2, and correlation coefficients ρ_{12}, ρ_{13}, and ρ_{23}. Then, if v_i is the ith moment about zero of the random variable, max (U_1, U_2), and $\rho[U_3$, max $(U_1, U_2)]$ is the correlation coefficient between the new variable and U_3, Clark showed that

$$v_1 = V_1\Phi(\gamma) + V_2\Phi(-\gamma) + a\phi(\gamma) \qquad [A.10a]$$

$$v_2 = (V_1^2 + \sigma_1^2)\Phi(\gamma) + (V_2^2 + \sigma_2^2)\Phi(-\gamma) + (V_1 + V_2)a\phi(\gamma) \qquad [A.10b]$$

and

$$\rho[U_3, \max\,(U_1, U_2)] = \frac{\sigma_1\rho_{13}\,\Phi(\gamma) + \sigma_2\,\rho_{23}\,\Phi(-\gamma)}{(v_2 - v_1^2)^{1/2}} \qquad [A.10c]$$

where

$$\phi(\omega) = (2\pi)^{-1/2} \exp\left(\frac{-\omega^2}{2}\right) \qquad \text{(standard normal distribution)}$$

$$\Phi(\omega) = \int_{-\infty}^{\omega} \phi(t)\,dt \qquad \text{(standard cumulative normal curve)}$$

$$a^2 = \sigma_1^2 + \sigma_2^2 - 2\sigma_1\sigma_2\rho_{12} \qquad \text{(variance of the difference } V_1 - V_2\text{)}$$

and

$$\gamma = \frac{V_1 - V_2}{a}$$

The distribution of the maximum of U_1 and U_2 can now be approximated by a normal distribution by using the moments in Eqs. [A.10]. In other words,

$$\max(U_1, U_2) \sim N(v_1, v_2 - v_1^2) \qquad \text{[A.11]}$$

This approximation can be used recursively to obtain the approximate distribution of the maximum of any number of normal variables, by calculating the mean vector and covariance matrix of $[U_1, \ldots, U_{K-2}, \max(U_{K-1}, U_K)]$ and repeating the process to calculate them for $[U_1, \ldots, \max(U_{K-2}, \max(U_{K-1}, U_K))]$, $[U_1, \ldots, \max(U_{K-3}, \max(U_{K-2}, \max(U_{K-1}, U_K)))]$, and so on. If U_K is the last variable to be considered, we have after $K - 1$ iterations the approximate mean of the maximum of the $K - 1$ variables considered (denoted V_{-k}), the approximate variance of the maximum (σ^2_{-k}), and the correlation between this maximum and U_K (denoted $\rho_{-k,k}$). In other words, the following parameters are known at this point.

$$V_{-k} = E\,[\max(U_1, \ldots, U_{k-1}, U_{k+1}, \ldots, U_K)] \qquad \text{[A.12a]}$$

$$\sigma^2_{-k} = \text{var}\,[\max(U_1, \ldots, U_{k-1}, U_{k+1}, \ldots, U_K)] \qquad \text{[A.12b]}$$

$$\rho_{-k,k} = \text{corr}\,[U_k, \max(U_k, \ldots, U_{k-1}, U_{k+1}, \ldots, U_K)] \qquad \text{[A.12c]}$$

It is now possible to calculate the probability that the kth variate is actually the largest (i.e., the probability that the kth alternative is chosen), P_k, as follows:

$$\begin{aligned} P_k &= \Pr\left\{U_k \geq \max_{\forall l \neq k} [U_l]\right\} \\ &= \Pr\{[\max(U_1, \ldots, U_{k-1}, U_{k+1}, \ldots, U_k)] - U_k \leq 0\} \\ &= \Phi\left(\frac{V_k - V_{-k}}{\sqrt{\sigma_k^2 + \sigma^2_{-k} - 2\sigma_k\sigma_{-k}\rho_{-k,k}}}\right) \end{aligned} \qquad \text{[A.13]}$$

Due to the error introduced by Eq. [A.11], Eq. [A.13] is only approximate. The accuracy of this approximation was investigated by Daganzo et al. (1977), Lerman and Manski (1978), and Horowitz et al. (1982).

Simulation Method

As explained in Section 10.1, the simulation method for computing the choice probability is based on the following iteration:

1. Draw a realization of a random MVN distributed vector.
2. Record the largest component of this vector.

The choice probability of each component is then approximated by the frequency with which this component was recorded as the largest. The accuracy of this method for computing the choice probability increases as the number of iterations grows.

The only difficulty with this method, when applied to a general MVN distribution is the Monte Carlo drawing process. Even though it is easy to generate a single realization from a normally distributed random variable, it is more difficult to generate a normal random vector with known covariance matrix. The process can be accomplished with a factorization of the covariance matrix, a process known as *Choleski factorization.*

A factorization consists of finding the "root" matrix, **R**, of a given matrix, $\boldsymbol{\Sigma}$, such that

$$\mathbf{R} \cdot \mathbf{R}^T = \boldsymbol{\Sigma} \qquad [A.14]$$

The Choleski factorization finds a triangular matrix, **R**, which satisfies Eq. [A.14].† In order to generate a realization, **x**, from a random vector, **X**, where

$$\mathbf{X} \sim \text{MVN}\,(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad [A.15]$$

a vector of standard normal independent random variates, **s**, can easily be generated. The distribution of the random variable **S** (from which **s** is drawn) is given by

$$\mathbf{S} \sim \text{MVN}\,(\mathbf{0}, \mathbf{I}) \qquad [A.16]$$

where **I** is the identity matrix. Now consider the transformation

$$\mathbf{X} = \mathbf{R} \cdot \mathbf{S} + \boldsymbol{\mu} \qquad [A.17]$$

Using the relationship mentioned at the beginning of this appendix (see Eqs. [A.3] and [A.4]), the distribution of **X** can be obtained from the distribution of **S**, that is,

$$E[\mathbf{X}] = \mathbf{R} \cdot \mathbf{0} + \boldsymbol{\mu} = \boldsymbol{\mu} \qquad [A.18a]$$

$$\text{cov}\,[\mathbf{X}] = \mathbf{R}^T \cdot \mathbf{I} \cdot \mathbf{R} = \boldsymbol{\Sigma} \qquad [A.18b]$$

Thus the transformation

$$\mathbf{x} = \mathbf{R} \cdot \mathbf{s} + \boldsymbol{\mu} \qquad [A.19]$$

generates a realization, **x**, from the distribution given in Eq. [A.15].

†The mechanics of the factorization process can be found in any linear algebra book. It is also available as a part of many computer libraries.

References

Aashtiani, H. Z., and T. L. Magnanti (1981). Equilibria on a Congested Transportation Network. *SIAM Journal on Algebraic and Discrete Methods 2*(3), pp. 213–226.

Abdulaal, M., and L. J. LeBlanc (1979). Methods for Combining Modal Split and Equilibrium Assignment Models. *Transportation Science 13*(4), pp. 292–314.

Allsop, R. E. (1972). Delay at Fixed Time Traffic Signals I: Theoretical Analysis. *Transportation Science 6*(2), pp. 260–285.

Almond, J. (1967). Traffic Assignment with Flow-Dependent Journey Times. In L. C. Edie, R. Herman, and R. Rothery (Eds.) *Vehicular Traffic Science*, Proceedings, 3rd International Symposium on the Theory of Traffic Flow, American Elsevier, New York, pp. 222–234.

Andersen, J. (1977). A Method for the Analysis of Transit Networks. In M. Roubens (Ed.), *Advances in Operations Research*, pp. 1–8. North-Holland, Amsterdam.

Beckmann, M. J., C. B. McGuire, and C. B. Winsten (1956). *Studies in the Economics of Transportation.* Yale University Press, New Haven, Conn.

Ben-Akiva, M., and S. R. Lerman (1978). Disaggregate Travel and Mobility Choice Models and Measures of Accessibility. *Proceedings, 3rd International Conference on Behavioral Travel Modelling*, Adelaide, Australia.

Blum, J. R. (1954). Multidimensional Stochastic Approximation Methods. *Annals of Mathematical Statistics 25*, pp. 737–744.

Bolland, J. D., M. D. Hall, and D. Van Vliet (1979). SATURN: A Model for the Evaluation of Traffic Management Schemes. *Working Paper 106*, Institute for Transportation Studies, University of Leeds, England.

Bouthelier, F., and C. F. Daganzo (1979). Aggregation with Multinomial Probit and Estimation of Disaggregate Models with Aggregate Data: A New Methodological Approach. *Transportation Research 13B*(2), pp. 133–146.

BOVY, P. H. L., AND G. R. M. JANSEN (1981). Network Modelling Effects in Equilibrium Assignment: An Empirical Investigation. *Proceedings, International Symposium on Frontiers in Transportation Equilibrium and Supply Models*, Transportation Research Center, University of Montreal, Montreal.

BOYCE, D. E. (1981). Editorial Note. *Environment and Planning A 13*(4), pp. 395–398.

BRADLEY, S. P., A. C. HAX, AND T. L. MAGNANTI (1977). *Applied Mathematical Programming*. Addison-Wesley, Reading, Mass.

BRAESS, D. (1968). Über ein Paradox der Verkehrsplanung. *Unternehmenstorchung 12*, pp. 258–268.

BRANSTON, D. (1976). Link Capacity Functions: A Review. *Transportation Research 10*(4), pp. 223–236.

BRUYNOOGHE, M., A. GILBERT, AND M. SAKAROVICH (1968). Une Méthode d'affectation du traffic. *Proceedings, 4th International Symposium on the Theory of Road Traffic Flow*, Karlsruhe, West Germany.

BURRELL, J. E. (1968). Multipath Route Assignment and Its Applications to Capacity Restraint. *Proceedings, 4th International Symposium on the Theory of Road Traffic Flow*, Karlsruhe, West Germany.

BURRELL, J. E. (1976). Multipath Route Assignment: A Comparison of Two Methods. In M. Florian (Ed.), *Traffic Equilibrium Methods*, Lecture Notes in Economics and Mathematical Systems 118, Springer-Verlag, New York, pp. 229–239.

CEDAR, A. (1978). *Network Theory and Selected Topics in Dynamic Programming*. Dekel Academic Publications, Tel Aviv.

CLARK, C. E. (1961). The Greatest of a Finite Set of Random Variables. *Operations Research 9*(2), pp. 145–162.

DAFERMOS, S. C. (1971). An Extended Traffic Assignment Model with Application to Two-Way Traffic. *Transportation Science 5*(4), pp. 366–389.

DAFERMOS, S. C. (1976). Integrated Equilibrium Flow Models for Transportation Planning. In M. Florian (Ed.), *Traffic Equilibrium Methods*, Lecture Notes in Economics and Mathematical Systems 118, Springer-Verlag, New York, pp. 106–118.

DAFERMOS, S. C. (1980). Traffic Equilibrium and Variational Inequalities. *Transportation Science 14*(1), pp. 42–54.

DAFERMOS, S. C. (1982). Relaxation Algorithms for the General Asymmetric Traffic Equilibrium Problem. *Transportation Science 16*(2), pp. 231–240.

DAGANZO, C. F. (1977a). Some Statistical Problems in Connection with Traffic Assignment. *Transportation Research 11*(6), pp. 385–389.

DAGANZO, C. F. (1977b). Some Research on Traffic Assignment Methodology Selection. *Working Paper 7703*, Institute of Transportation Studies, University of California, Berkeley.

DAGANZO, C. F. (1977c). On the Traffic Assignment Problem with Flow Dependent Costs—I. *Transportation Research 11*(6), pp. 433–438.

DAGANZO, C. F. (1979). *Multinomial Probit: The Theory and Its Application to Demand Forecasting*. Academic Press, New York.

DAGANZO, C. F. (1981). Estimation of Gap Acceptance Parameters Within and Across the Population from Direct Roadside Observation. *Transportation Research 15B*(1), pp. 1–15.

DAGANZO, C. F. (1982). Unconstrained Extremal Formulation of Some Transportation Equilibrium Problems. *Transportation Science 16*(3), pp. 332–360.

DAGANZO, C. F., AND Y. SHEFFI (1977). On Stochastic Models of Traffic Assignment. *Transportation Science 11*(3), pp. 253–274.

DAGANZO, C. F., F. BOUTHELIER, AND Y. SHEFFI (1977). Multinomial Probit and Qualitative Choice: A Computationally Efficient Algorithm. *Transportation Science 11*(4), pp. 338–358.

DANTZIG, G. B., S. F. MAIER, AND Z. F. LANDSDOWNE (1976). The Application of Decomposition to Transportation Networks Analysis. *Report DOT-TSC-OST-76-26*, U.S. Department of Transportation, Washington, D.C.

DAVIDSON, K. B. (1966). A Flow-Travel Time Relationship for Use in Transportation Planning. *Proceedings, Australian Road Research Board*, Melbourne, Vol. 3, pp. 183–194.

DIAL, R. B. (1971). A Probabilistic Multipath Traffic Assignment Algorithm Which Obviates Path Enumeration. *Transportation Research 5*(2), pp. 83–111.

DIAL, R. B., AND R. E. BUNYAN (1968). Public Transit Planning System. *Socio-Economic Planning Sciences 1*, pp. 345–362.

DIAL, R. B., F. GLOVER, D. KARNEY, AND D. KLINGMAN (1977). A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees. *Research Report CCS 291*, Center for Cybernetics Studies, University of Texas, Austin.

DOMENCICH, T. A., AND D. MCFADDEN (1975). *Urban Travel Demand: A Behavioral Analysis.* American Elsevier, New York.

DREW, D. R. (1968). *Traffic Flow Theory and Control.* McGraw-Hill, New York.

EASH, R. W., B. N. JANSON, AND D. E. BOYCE (1979). Equilibrium Trip Assignment: Advantages and Implication for Practice. *TRB Transportation Research Record 728*, pp. 1–7.

EVANS, S. P. (1976). Derivation and Analysis of Some Models for Combining Trip Distribution and Assignment. *Transportation Research 10*(1), pp. 37–57.

FEDERAL HIGHWAY ADMINISTRATION (1977). *Computer Programs for Urban Transportation Planning: PLANPAC/BACKPAC General Information Manual.* U.S. Department of Transportation, Washington, D.C.

FERLAND, J. A., M. FLORIAN, AND C. ACHIM (1975). On Incremental Methods for Traffic Assignment. *Transportation Research 9*(4), pp. 237–239.

FINNEY, D. (1964). *Probit Analysis.* Cambridge University Press, Cambridge, England.

FISK, C. (1980). Some Developments in Equilibrium Traffic Assignment. *Transportation Research 14B*(3), pp. 243–255.

FISK, C., AND S. NGUYEN (1982). Solution Algorithms for Network Equilibrium Models with Asymmetric User Costs. *Transportation Science 16*(3), pp. 361–381.

FLORIAN, M. (1977). A Traffic Equilibrium Model of Travel by Car and Public Transit Modes. *Transportation Science 11*(2), pp. 166–179.

FLORIAN, M. (1981). The Convergence of Diagonalization Algorithms for Fixed Demand Asymmetric Network Equilibrium Problems. *Publication 198*, Center of Transportation Research, University of Montreal, Montreal.

FLORIAN, M., AND B. FOX (1976). On the Probabilistic Origin of Dial's Multipath Traffic Assignment Model. *Transportation Research 10*(5), pp. 339–341.

Florian, M., and S. Nguyen (1976). An Application and Validation of Equilibrium Trip Assignment Methods. *Transportation Science 10*(4), pp. 374–390.

Florian, M., and S. Nguyen (1978). A Combined Trip Distribution Modal Split and Trip Assignment Model. *Transportation Research 12*(4), pp. 241–246.

Florian, M., S. Nguyen, and J. Ferland (1975). On the Combined Distribution-Assignment of Traffic. *Transportation Science 9*(1), pp. 43–53.

Frank, M., and P. Wolfe (1956). An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly 3*(1-2), pp. 95–110.

Freeman Fox and Associates (1972). Speed/Flow Relationships on Suburban Main Roads. *Road Research Laboratory, Report of the Department of Environment*, London.

Gartner, N. H. (1980). Optimal Traffic Assignment with Elastic Demands: A Review; Part II: Algorithmic Approaches. *Transportation Science 14*(2), pp. 192–208.

Glover, F., D. Karney, and D. Klingman (1974). Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems. *Networks 4*(3), pp. 191–212.

Greenshields, B. D. (1934). A Study of Traffic Capacity. *Highway Research Board Proceedings*, Vol. 14, part I, pp 448–478.

Gunnarson, S. (1972). An Algorithm for Multipath Traffic Assignment. *Proceedings, PTRC Urban Traffic Model Research Seminar*, London.

Haberman, R. (1977). *Mathematical Models: Mechanical Vibrations, Populations Dynamics and Traffic Flow.* Prentice-Hall, Englewood Cliffs, New Jersey.

Han, A. F., and N. H. Wilson (1982). The Allocation of Buses in Heavily Utilized Networks with Overlapping Routes. *Transportation Research 16B*(3), pp. 221–232.

Hausman, J. A., and D. A. Wise (1978). A Conditional Probit Model for Qualitative Choice: Discrete Decisions Recognizing Interdependence and Heterogeneous Preferences. *Econometrica 46*(2), pp. 403–426.

Hensher, D. A., and L. W. Johnson (1981). *Applied Discrete-Choice Modelling.* Croom Helm, London; John Wiley & Sons, New York.

Herman, R., and G. H. Weiss (1961). Comments on the Highway Crossing Problem. *Operations Research 9*(6), pp. 828–840.

Homburger, W. S. (ed.) (1982). *Transportation and Traffic Engineering Handbook.* Prentice-Hall, Englewood Cliffs, New Jersey.

Horowitz, J. L. (1983). Stability of Equilibrium in a Two-Link Transportation Network. *62nd Annual TRB Conference*, Washington, D.C., January 1983.

Horowitz, J. L., J. M. Sparmann, and C. F. Daganzo (1982). An Investigation of the Accuracy of the Clark Approximation for the Multinomial Probit Model. *Transportation Science 16*(3), pp. 382–401.

Hutchinson, B. G. (1974). *Principles of Urban Transport Systems Planning.* McGraw-Hill, New York.

Jornsten, K., and S. Nguyen (1980). On the Estimation of a Trip Matrix from Network Data. *Report NITH-MAT-R-79-36*, Linkoping Institute of Technology, Sweden.

Kanafani, A. (1983). *Transportation Demand Analysis.* McGraw-Hill, New York.

Knight, F. H. (1924). Some Fallacies in the Interpretation of Social Costs. *Quarterly Journal of Economics 38*, pp. 582–606.

Knödel, W. (1969). *Graphentheoreticishe Methoden und ihre Anwendungen.* Springer-Verlag, Berlin.

Koppelman, F. S. (1976). Guidelines for Aggregate Travel Predictions Using Disaggregate Choice Models. *TRB Transportation Research Record 610*, pp. 19–24.

Lasdon, L. S. (1970). *Optimization Theory for Large Systems.* Macmillan, New York.

LeBlanc, L. J., and M. Abdulaal (1982). Combined Mode Split-Assignment and Distribution-Modal Split-Assignment Models with Multiple Groups of Travelers. *Transportation Science 16*(4), pp. 430–442.

LeBlanc, L. J., and K. Farhangian (1981). Efficient Algorithms for Solving Elastic Demand Traffic Assignment Problems and Mode Split-Assignment Problems. *Transportation Science 15*(4), pp. 306–317.

LeBlanc, L. J., and K. Farhangian (1982). Selection of Trip Table Which Reproduces Observed Link Flows. *Transportation Research 16B*(2), pp. 83–88.

LeBlanc, L. J., E. K. Morlok, and W. Pierskalla (1975). An Efficient Approach to Solving the Road Network Equilibrium Traffic Assignment Problem. *Transportation Research 9*(5), pp. 309–318.

Lerman, S. R., and C. F. Manski (1978). An Estimator for the Generalized Multinomial Probit Choice Model. *56th Annual TRB Meeting*, Washington, D.C.

Luenberger, D. G. (1973). *Introduction to Linear and Nonlinear Programming.* Addison-Wesley, Reading, Mass.

Magnanti, T. L., and B. L. Goldin (1985). *Network Optimization.* John Wiley & Sons, New York.

Mahmassani, H., and Y. Sheffi (1981). Using Gap Sequences to Estimate Gap Acceptance Functions. *Transportation Research 15B*(3), pp. 143–148.

Manheim, M. L. (1979). *Fundamentals of Transportation Systems Analysis.* MIT Press, Cambridge, Mass.

Manheim, M. L., and E. R. Ruiter (1970). DODOTRANS I: A Decision Oriented Computer Language for Analysis of Multimodal Transportation Systems. *HRB Highway Research Record 314*, pp. 135–163.

May, A. D. (1965). Traffic Flow Theory—The Traffic Engineer's Challenge. *Proceedings, Institute of Traffic Engineers*, pp. 290–303.

Mayberry, J. P. (1970). Structural Requirements for Abstract-Mode Models of Passenger Transportation. In R. Quandt (Ed.), *The Demand for Travel: Theory and Measurement.* Heath, Lexington, Mass.

McFadden, D. (1973). Conditional Logit Analysis of Qualitative Choice Behavior. In P. Zarembka (Ed.), *Frontiers in Econometrics*, Academic Press, New York, pp. 105–142.

McFadden, D., and F. Reid (1975). Aggregate Travel Demand Forecasting from Disaggregate Behavioral Models. *TRB Transportation Research Board 534*, pp. 24–37.

Metropolitan Toronto Regional Transportation Study (1967). *Calibration of Regional Traffic Prediction Model for the A.M. Peak Period.* Toronto, Ontario.

Miller, A. J. (1972). Nine Estimators of Gap-Acceptance Parameters. G. F. Newell (Ed.), *Proceedings, 5th International Symposium on the Theory of Traffic Flow and Transportation*, American Elsevier, New York, pp. 215–235.

Mimis, S. (1984). Equilibrium Traffic Assignment Models for Urban Networks. S. M. Thesis, Center for Transportation Studies, M.I.T., Cambridge, Mass.

Moore, E. (1957). The Shortest Path through a Maze. *Proceedings of the International*

Symposium on the Theory of Switching. Harvard University Press, Cambridge, Mass, pp. 285–292.

Morlok, E. K. (1978). *Introduction to Transportation Engineering and Planning.* McGraw-Hill, New York.

Murchland, J. D. (1969). Road Traffic Distribution in Equilibrium. Conference Paper, *Mathematical Methods in the Economic Sciences,* Mathematiches Forschungsinstitut, Oberwolfack, West Germany.

Murchland, J. D. (1970). Braess's Paradox of Traffic Flow. *Transportation Research 4*(4), pp. 391–394.

Nagurney, A. B. (1983). Comparative Tests of Multimodal Traffic Equilibrium Methods. *U.S. DOT, Program of University Research Report,* Brown University, Providence, Rhode Island.

Newell, G. F. (1980). *Traffic Flow on Transportation Networks.* MIT Press, Cambridge, Mass.

Nguyen, S. (1974a). A Unified Approach to Equilibrium Methods for Traffic Assignment. In M. Beckmann and H. P. Künzi (Eds.), *Traffic Equilibrium Methods,* Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, New York.

Nguyen, S. (1974b). An Algorithm for the Traffic Assignment Problem. *Transportation Science 8*(3), pp. 203–216.

Nguyen, S. (1977). Estimating an OD Matrix from Network Data: A Network Equilibrium Approach, *Publication No. 87, Centre de recherche sur les transports,* University of Montreal, Montreal.

Pape, U. (1974). Implementation and Efficiency of Moore-Algorithms for the Shortest Route Problem. *Mathematical Programming 7*(2), pp. 212–222.

Pignataro, L. J. (1973). *Traffic Engineering: Theory and Practice.* Prentice-Hall, Englewood Cliffs, New Jersey.

Potts, R. B., and R. M. Oliver (1972). *Flows in Transportation Networks.* Academic Press, New York.

Powell, W. B., and Y. Sheffi (1982). The Convergence of Equilibrium Algorithms with Predetermined Step Sizes. *Transportation Science 16*(1), pp. 45–55.

Safwat, K. N. A., and T. L. Magnanti (1982). A Combined Trip Generation, Trip Distribution, Modal Split and Traffic Assignment Model. *Working Paper OR112-82,* Operations Research Center, M.I.T., Cambridge, Mass.

Schneider, M. (1973). Probability Maximization in Networks. *Proceedings, International Conference on Transportation Research,* Bruges, Belgium, pp. 748–755.

Sheffi, Y. (1979). A Note on the Turn and Arrival Likelihood Algorithms of Traffic Assignment. *Transportation Research 13B*(2), pp. 147–150.

Sheffi, Y. (1981). Aggregation and Equilibrium with Multinomial Logit Models. *Proceedings, International Symposium on Equilibrium and Supply Models,* University of Montreal, Montreal.

Sheffi, Y., and C. F. Daganzo (1978a). Another "Paradox" of Traffic Flow. *Transportation Research 12*(1), pp. 43–46.

Sheffi, Y., and C. F. Daganzo (1978b). Hypernetworks and Supply Demand Equilibrium with Disaggregate Demand Models. TRB *Transportation Research Record 673,* pp. 113–121.

Sheffi, Y., and C. F. Daganzo (1980). Computation of Equilibrium over Transportation Networks: The Case of Disaggregate Demand Models. *Transportation Science 14*(2), pp. 155–173.

SHEFFI, Y., AND W. B. POWELL (1981). A Comparison of Stochastic and Deterministic Traffic Assignment over Congested Networks. *Transportation Research 15B*(1), pp. 53–64.

SHEFFI, Y., AND W. B. POWELL (1982). An Algorithm for the Equilibrium Assignment Problem with Random Link Times. *Networks 12*(2), pp. 191–207.

SHEFFI, Y., R. HALL, AND C. F. DAGANZO (1982). On the Estimation of the Multinomial Probit Model. *Transportation Research 16A*(5-6), pp. 447–456.

SIMMONS, D. M. (1975). *Nonlinear Programming for Operations Research.* Prentice-Hall, Englewood Cliffs, New Jersey.

SMITH, M. J. (1979). The Existence, Uniqueness and Stability of Traffic Equilibria. *Transportation Research 13B*(4), pp. 295–304.

STOPHER, P. R., AND A. H. MEYBURG (1975). *Urban Transportation Modeling and Planning.* Lexington Books, Lexington, Mass.

TAYLOR, M. A. P. (1977). Parameter Estimation and Sensitivity of Parameter Values in a Flow-Rate/Travel-Time Relation. *Transportation Science 11*(3), pp. 275–292.

TOBIN, R. L. (1977). An Extension of Dial's Algorithm Utilizing a Model of Tripmaker's Perceptions. *Transportation Research 11*(5), pp. 337–342.

TYE, W. B., L. SHERMAN, M. KINNUCAN, D. NELSON, AND T. TARDIFF (1982). Application of Disaggregate Travel Demand Models. *NCHRP Report 253*, Transportation Research Board, Washington, D.C.

URBAN MASS TRANSPORTATION ADMINISTRATION (1977). *UMTA Transportation Planning System Reference Manual.* U.S. Department of Transportation, Washington, D.C.

U.S. BUREAU OF PUBLIC ROADS (1964). *Traffic Assignment Manual.* U.S. Department of Commerce, Washington, D.C.

VON FALKENHAUSEN, H. (1966). Traffic Assignment by a Stochastic Model. *Proceedings, 4th International Conference on Operational Science*, pp. 415–421.

VAN VLIET, D. (1976). Road Assignment—I, II. *Transportation Research 10*(3), pp. 137–149.

WAGNER, H. M. (1975). *Principles of Operations Research with Applications to Managerial Decisions*, 2nd ed. Prentice-Hall, Englewood Cliffs, New Jersey.

WARDROP, J. G. (1952). Some Theoretical Aspects of Road Traffic Research. *Proceedings, Institution of Civil Engineers II*(1), pp. 325–378.

WEBSTER, F. V. (1958). Traffic Signal Setting. *Road Research Laboratory Report 39*, Crowthorne, Berkshire, England.

WEISS, G. H., AND A. A. MARADUDIN (1962). Some Problems in Traffic Delay. *Operations Research 10*(1), pp. 74–104.

WIGAN, M. R. (1971). Benefit Assessment for Network Traffic Models and Application to Road Pricing. *Road Research Laboratory Report LR 417*, Crowthorne, Berkshire, England.

WILDE, D. J. (1964). *Optimum Seeking Methods.* Prentice-Hall, Englewood Cliffs, New Jersey.

WILKIE, D. F., AND R. G. STEFANEK (1971). Precise Determination of Equilibrium in

Travel Forecasting Problems Using Numerical Optimization Techniques. HRB *Highway Research Record 369*, pp. 239–252.

WILLIAMS, H. C. W. L. (1977). On the Formulation of Travel Demand Models and Economic Evaluation Measures of User Benefit. *Environment and Planning A 9*(3), pp. 285–344.

WILSON, A. G. (1967). A Statistical Theory of Spatial Distribution Models. *Transportation Research 1*(3), pp. 253–270.

WILSON, A. G. (1970). *Entropy in Urban and Regional Modelling*. Pion, London.

WISMER, D. A., AND R. CHATTERGY (1978). *Introduction to Nonlinear Optimization: A Problem Solving Approach*. Elsevier/North-Holland, New York.

ZANGWILL, W. I. (1969). *Nonlinear Programming: A Unified Approach*. Prentice-Hall, Englewood Cliffs, New Jersey.

Index