CÓDIGOS Fortran para NETWORK OTIMIZAÇÃO
por Dimitri P. Bertsekas

A maioria destes códigos são essencialmente aqueles listados no apêndice do livro didático
"Otimização da rede LINEAR: algoritmos e códigos",
MIT Press, 1991 por Dimitri P. Bertsekas

A teoria de muitos dos códigos é descrito no artigo do exame "leilão Algoritmos para problemas de fluxo de rede: Um Tutorial Introdução ", Computacional
Optimization and Applications, 1, 1992, 7-26, e no livro mais recente

"Otimização da rede: CONTÍNUA E modelos discretos",
ATHENA SCIENTIFIC de 1998 por Dimitri P. Bertsekas

Os códigos serão atualizadas periodicamente, assim que sua opinião será muito valiosa.
Enviar e-mail para

dimitrib@mit.edu

ou para USMail

Prof. Dimitri P. Bertsekas
MIT, Rm 35-210
Cambridge, Mass. 02139
EUA

****
****

USO:

Todos os códigos de usar aritmética inteira e pode ser compilado com o compilador Absoft em todos os computadores Macintosh (sujeito a
limitações de memória) .A faixa de custo do problema a ser resolvido deve ser
suficientemente limitado para evitar integer overflow. Ao alterar uma entrada e poucos
instruções de saída, os códigos podem ser facilmente adaptadas para outros computadores e
Compiladores Fortran. Os códigos não deve ser modificado, exceto para o mínimo
modificações necessárias para fazê-los funcionar em outros compiladores. Por Favor
indicar claramente as suas modificações ao relatar a investigação.

Todos os códigos podem ser utilizados para qualquer finalidade não comercial. Em particular, os códigos
não deve ser integrado em qualquer produto comercial ou ser usado para satisfazer qualquer
nao Governamentais ou deriverables industriais

****
****

CONTEÚDO:

O que se segue é um resumo dos códigos:

1) GRIDGEN: gera problemas de fluxo de custo mínimo com uma estrutura de grade.

2) NGCONVERT: Converte problemas desde o formato NETGEN para um formato padrão.

3) PAPE_ALL_DEST: o método de Pape para caminhos mais curtos a partir de uma origem a todos
destinos.

4) HEAP_ALL_DEST: Binary método heap para caminhos mais curtos a partir de uma origem
para todos os destinos.

5) HEAP_SELECT_DEST: Binary método heap para caminhos mais curtos a partir de uma origem
para destinos selecionados.

6) AUCTION_SELECT_DEST: algoritmo Leilão de caminhos mais curtos a partir de uma origem
para destinos selecionados, conforme o jornal "An Leilão Algoritmo para Shortest
Caminhos ", SIAM J. on Optimization, Vol. 1, 1991, pp. 425-447, por DP Bertsekas.

7) RELAX_QC: método de relaxamento para o fluxo de custo min, de acordo com o papel
"A estrutura unificada para Métodos primal-dual em
Fluxo de problemas de rede de Custo Mínimo ", Math. Programação, Vol. 32, pp. 125-145,
1985, pela DP Bertsekas.

8) LEILÃO: algoritmo leilão Avançado para problemas de atribuição simétricas, como por
o original 1.979 relatório do MIT "Um algoritmo distribuído para a Cessão
Problema "; ver também o jornal" The Auction Algoritmo: um relaxamento Distribuído
Método para o Problema, "Annals of Operations Research, Vol. 14, 1988, pp. 105-123, por DP Bertsekas.

9) AUCTION_FLP: Igual ao leilão, mas usa a aritmética de ponto flutuante para lidar
com problemas com a faixa de custo grande e / ou dimensão. Para tais problemas a
preços pode transbordar o intervalo inteiro no decurso do algoritmo.

10) AUCTION_AS: algoritmo Leilão de problemas de atribuição assimétricas.

11) AUCTION_FR: Atacante algoritmo leilão / reversa para atribuição simétrica
problemas.

12) NAUCTION_SP: Combinado leilão ingênuo e sequencial método caminho mais curto
para atribuição.

13) FORD_FULKERSON: Ford-Fulkerson algoritmo para max-flow.

14) E_RELAX_MF: algoritmo E-relaxamento para max-flow.

15) E_RELAX: algoritmo E-relaxamento para o fluxo de custo min de acordo com o papel
"Métodos de relaxamento distribuídas para problemas de fluxo de rede Linear", Proc. 25 IEEE
Conferência sobre a decisão e controle, Atenas, Grécia, 1986, por DP Bertsekas.

16) E_RELAX_N: algoritmo E-relaxamento para o fluxo de custo min; itera de ambos
nós excedentes positivos e negativos.

17) SLF: pequena etiqueta First (SLF) algoritmo para encontrar caminhos mais curtos de um
origem para todos os destinos, de acordo com o jornal "A etiqueta simples e rápida Corrigindo
Método para caminhos mais curtos, "Redes, Vol. 23, 1993, pp. 703-709, por DP
Bertsekas.

18) SLFT: Combinação de pequeno selo First (SLF) e algoritmos de limiar
para encontrar caminhos mais curtos de uma origem para todos os destinos, de acordo com o papel
"A etiqueta simples e rápida Corrigindo Método para caminhos mais curtos," Redes, Vol. 23,
703-709 1993, pp., Pela DP Bertsekas.

19) AUCTION_MF: algoritmo de Leilão para o problema de fluxo máximo, de acordo com o papel
"Um leilão Algoritmo para o Max-Flow Problem", JOTA, Vol. 87, 1995, pp. 69-101, pela DP
Bertsekas.

Um formato comum tem sido adoptada em que cada um dos códigos (excepto para o
códigos de atribuição) faz o seguinte:
(A)
Lê o problema em um modelo comum.
(B)
Converte o problema a uma forma adequada para o algoritmo usado.
(C)
Chama o algoritmo para resolver o problema.
(D)
Emite os resultados.

[Os códigos de atribuição incluir um gerador problema aleatório embutido.]

O formulário padrão adotado é o utilizado para o método simplex (cf. \ Seção
2.4). Em particular, o problema é especificado por:

O número de nós $ N $.

O número de arcos $ A $.

Os quatro $ A $ -Comprimento arrays $ \ TI começam $, $ \ it END $, $ \ custa US $, $ \ it CAPACIDADE $.

Os US $ N $ array -Comprimento $ \ it SUPPLY $. \ Endlist

Como na Seção 2.4, as matrizes representam o seguinte:

START (a): O nó de início de arco $ a $.

END (a): O nó final do arco $ a $.

COST (a): O coeficiente de custo do arco $ a $.

CAPACIDADE (a): O fluxo limite superior do arco $ a $.

SUPPLY (i): A oferta de nó $ i $.

A representação formulário padrão do problema é o mais simples. No entanto,
para evitar conversões desnecessários, é
mais eficiente para representar o problema com um formato que é adaptado
com o método em questão. Assim, para uso a longo prazo, o leitor pode desejar modificar o
a porção de entrada dos códigos referidos.

****
****

LISTAS:

****
****

Gerador de Problemas e Códigos de Conversão

****
****

GRIDGEN

Para fornecer ao leitor os meios para criar
problemas de teste nonbipartite, damos um gerador simples, chamado GRIDGEN, que
constrói problemas aleatórios
 com um
grade de duas dimensões com estrutura subjacente envolvente. Os arcos de grade formar um
`` Esqueleto, '' viabilidade problema garantindo. Arcos adicionais são adicionadas
entre nós selecionados aleatoriamente, conforme especificado pelo usuário.
 Os seguintes parâmetros de problema são necessários como entrada:

{\ Bf $ \ it DIM1 $,

$ \ Lo DIM2 $:} Estas são as duas dimensões da grelha; o número de nós está
$ \ It N = DIM1 \ times DIM2 $.

{\ Bf $ \ lo ADDARCS $:} Este é o número de arcos em adição a $ $ arcos que 6N
são gerados automaticamente. Em particular, existem quatro arcos por nó
formar o grid, dois arcos por nó que começam no nó e terminar a uma aleatoriamente
nó selecionado, e US $ \ it ADDARCS $ arcos adicionais que se conectam selecionados aleatoriamente
pares de nós.

{\ Bf $ \ it TOTSUPPLY $:} A oferta total, ou seja, a soma das prestações de
as fontes (nós que têm alimentação positiva).

{\ Bf $ \ it FONTES $, $ \ ele afunda $:} O número de nós que têm positivo
abastecimento, e o número de nós que têm de alimentação negativa, respectivamente. Se
tanto $ \ it FONTES $ e $ \ ele afunda $ são menos de US $ N / 4 $, a fonte
nós e nós sorvedouros são selecionados aleatoriamente de acordo com um uniforme
distribuição; caso contrário nós 1 a $ \ it FONTES $ são os nós de origem, e
nós são $ \ it (N + 1-PIAS) $ a $ N $ são os
nós pia. Em ambos os casos, o
fornecimento de uma fonte é
$$ \ It TOTSUPPLY \ relação às fontes de $$
e
o fornecimento de uma pia é $$ \ it - {TOTSUPPLY \ over PIAS} $$ (em maior
possível aproximação).

{\ Bf $ \ it MINCOST $, $ \ it MAXCOST $:} O coeficiente de custo dos $ 4N $ arcos de grade
é de R $ \ it MAXCOST $. Todos os outros arcos têm coeficiente de custo selecionado
de acordo com uma distribuição uniforme entre US $ \ it MINCOST $ e $ \ it MAXCOST $.
Assim, existe uma
incentivo para evitar os arcos da rede, tanto quanto possível, a uma solução óptima.

{\ Bf $ \ it MINCAP $, $ \ it MaxCap $:} A capacidade dos arcos da rede é de R $ \ it
TOTSUPPLY $. A capacidade dos outros arcos é seleccionado de acordo com uma uniforme
distribuição
entre US $ \ it MINCAP $ e $ \ it MaxCap $.
Os arcos da garantia grade `` esqueleto '' que o
problema é viável (qualquer fonte pode ser ligada a qualquer dissipador com um
caminho de `` capacidade '' $ \ it TOTSUPPLY $). No entanto, estes arcos de alto custo,
Assim, com o ótimo, eles
tendem a ter pouco fluxo (se houver). Assim, para gerar um problema significativo,

```
o leitor deve especificar um bastante grande valor para $ \ it ADDARCS
$. Note-se que o
problema gerado não pode ser ilimitada uma vez que cada arco tem um
fluxo superior reais
obrigado.

C ***********************************************
**********************
C
C MINCOST GRID PROBLEMA GERADOR
C por Dimitri P. Bertsekas
C dezembro 1990
C
C ***********************************************
**********************
C
C este código gera A GRID MINCOST PROBLEMA
C e saídas do problema em formato normalizado, o arquivo "FOR013.DAT".
C
C
      Implícita INTEGER (AZ)
      MARK LÓGICO
      STARTN DIMENSÃO (50000), ENDN (50000), C (50000), U (50000)
      DIMENSÃO B (10000)
      MARK DIMENSÃO (10000)


C Este código inclui UM UNIFORME Random Number Generator
C que retorna um valor na (0,1)

C gerador aleatório STUFF

      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      RAN REAIS

C INITIALIZE Gerador Aleatório

      ISEED = 13502460
      CHAME SETRAN (ISEED)

      PRINT *, 'Criar um 2-D GRID MINCOST FLUXO DE PROBLEMA GRÁFICO'
      PRINT *, '****************************************** '
5 PRINT *, "ENTER # de nós em cada uma das duas dimensões"
      LEIA *, DIM1, DIM2

      IF (DIM1.LE.1) THEN
        PRINT *, "Dimensão 1 deve ser maior que 1 '
               IR PARA 5
      END IF
      IF (DIM2.LE.1) THEN
        PRINT *, "Dimensão 2 deve ser maior que 1 '
               IR PARA 5
      END IF


      N = DIM1 * DIM2

      PRINT *, 'número de arcos SO FAR', 6 * N
      PRINT *, "ENTER # DE ARCS ADICIONAIS '
      LER *, ADDARCS
```

```
      PRINT *, 'oferta total, # fontes, # de sumidouros'
      LER *, TOTSUPPLY, fontes, sumidouros

      PRINT *, "ENTER mínimo e máximo CUSTO DO COEFICIENTE DE '
      LEIA *, MINCOST, MAXCOST

      PRINT *, "ENTER CAPACIDADE E MAX MIN ARC '
      LEIA *, MINCAP, MaxCap

      ARC = 0

      FAZER 10 J = 1, DIM2
      FAZER 15 I = 1, DIM1
        NÓ = I + (J-1) * DIM1

C cria ARCS HORIZONTAL

        IF ((I.GT.1) .E. (I.LT.DIM1)) THEN
          ARC = ARC + 1
          STARTN (ARC) = NÓ
          ENDN (ARC) = NODE + 1
          C (ARC) = MAXCOST
          U (ARC) = TOTSUPPLY

          ARC = ARC + 1
          STARTN (ARC) = NÓ
          ENDN (ARC) = node-1
          C (ARC) = MAXCOST
          U (ARC) = TOTSUPPLY
        MAIS
          IF (I.EQ.1) THEN
            ARC = ARC + 1
            STARTN (ARC) = NÓ
            ENDN (ARC) = NODE + 1
            C (ARC) = MAXCOST
            U (ARC) = TOTSUPPLY

            ARC = ARC + 1
            STARTN (ARC) = NÓ
            ENDN (ARC) = NODE + (DIM1-1)
            C (ARC) = MAXCOST
            U (ARC) = TOTSUPPLY
          MAIS
            ARC = ARC + 1
            STARTN (ARC) = NÓ
            ENDN (ARC) = Nó- (DIM1-1)
            C (ARC) = MAXCOST
            U (ARC) = TOTSUPPLY

            ARC = ARC + 1
            STARTN (ARC) = NÓ
            ENDN (ARC) = node-1
            C (ARC) = MAXCOST
            U (ARC) = TOTSUPPLY
          END IF
        END IF



C cria ARCS VERTICAIS
```

```
        IF ((J.GT.1) .E. (J.LT.DIM2)) THEN
          ARC = ARC + 1
          STARTN (ARC) = NÓ
          ENDN (ARC) = NODE + DIM1
          C (ARC) = MAXCOST
          U (ARC) = TOTSUPPLY

          ARC = ARC + 1
          STARTN (ARC) = NÓ
          ENDN (ARC) = NODE-DIM1
          C (ARC) = MAXCOST
          U (ARC) = TOTSUPPLY
        MAIS
          IF (J.EQ.1) THEN
            ARC = ARC + 1
            STARTN (ARC) = NÓ
            ENDN (ARC) = NODE + DIM1
            C (ARC) = MAXCOST
            U (ARC) = TOTSUPPLY

            ARC = ARC + 1
            STARTN (ARC) = NÓ
            ENDN (ARC) = NODE + DIM1 * (DIM2-1)
            C (ARC) = MAXCOST
            U (ARC) = TOTSUPPLY
          MAIS
            ARC = ARC + 1
            STARTN (ARC) = NÓ
            ENDN (ARC) = I
            C (ARC) = MAXCOST
            U (ARC) = TOTSUPPLY

            ARC = ARC + 1
            STARTN (ARC) = NÓ
            ENDN (ARC) = NODE-DIM1
            C (ARC) = MAXCOST
            U (ARC) = TOTSUPPLY
          END IF
        END IF


C criar 2 ARCS com a Random nó final

        FAZER 8 K = 1,2

          ARC = ARC + 1
          STARTN (ARC) = NÓ
7 ENDN (ARC) = 1 + INT (RAN () * N)
          C (ARC) = MINCOST + RAN () * (MAXCOST-MINCOST)
          U (ARC) = MINCAP + RAN () * (MaxCap-MINCAP)
          IF (ENDN (ARC) .EQ.NODE) GOTO 7
8 CONTINUAR

15 CONTINUAR
10 CONTINUAR

C cria ADDARCS NEW ARCS com a Random começar e terminar NÓ

        FAZER 12 K = 1, ADDARCS
          ARC = ARC + 1
```

```
          NÓ = 1 + INT (RAN () * N)
          STARTN (ARC) = NÓ
13 ENDN (ARC) = 1 + INT (RAN () * N)
          C (ARC) = MINCOST + RAN () * (MAXCOST-MINCOST)
          U (ARC) = MINCAP + RAN () * (MaxCap-MINCAP)
          IF (ENDN (ARC) .EQ.NODE) GOTO 13
12 CONTINUAR

      NA = ARC


C gerar as pias e FONTES

      FAZER 17 I = 1, N
        B (I) = 0
        MARK (I) =. FALSE.
17 CONTINUAR

      SUPL = 0
      Fazer 20 i = 1, FONTES
18 NODE = 1 + INT (RAN () * N)
          IF (MARK (nó)) THEN
            GOTO 18
          MAIS
            MARK (nó) =. TRUE.
            B (nó) = INT (TOTSUPPLY / Fontes)
            SUPL = SUPL + B (nó)
            LAST = NÓ
          END IF
20 CONTINUAR

      B (LAST) = B (LAST) + TOTSUPPLY-SUPL

      DEM = 0
      FAZER 25 I = 1, pias
22 NODE = 1 + INT (RAN () * N)
          IF (MARK (nó)) THEN
            GOTO 22
          MAIS
            MARK (nó) =. TRUE.
            B (nó) = - INT (TOTSUPPLY / PIAS)
            DEM = DEM-B (nó)
          END IF
25 CONTINUAR


C equalizar OFERTA E PROCURA

      IF (SUPL.GT.DEM) THEN
        Fazer 30 NODE = 1, N
          IF (B (Nó) .LT.0) THEN
            B (nó) = B (Nó) - (Supl-DEM)
            GOTO 38
          END IF
30 CONTINUAR
       MAIS
         IF (SUPL.LT.DEM) THEN
           FAZER 35 NODE = 1, N
             IF (B (Nó) .GT.0) THEN
               B (nó) = B (Nó) + (DEM-SUPL)
               GOTO 38
```

```
            END IF
35 CONTINUAR
         END IF
       END IF

38 CONTINUAR


C
      PRINT *, '******************************************** '
      PRINT *, 'Escrevendo o PROBLEMA NO ARQUIVO FOR013.DAT'
      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' Novo ')
      Rewind (13)

C ESCREVA número de nós e arcos

      ESCREVA (13,1010) N, NA

C inicial de gravação, END, custo e capacidade de cada ARC

      DO 40 I = 1, NA
        WRITE (13,1020) STARTN (I), ENDN (I), C (I), L (I)
40 CONTINUAR

C ESCREVA SUPPLY de cada nó

      FAZER 50 I = 1, N
        ESCREVA (13,1000) B (I)
50 CONTINUAR

      ENDFILE (13)
      Rewind (13)

      PRINT *, "fim da escrita"

1000 FORMATO (1I8)
1010 FORMATO (2I8)
1020 FORMATO (4I8)

      PARE
      FIM

      SUBROUTINE SETRAN (ISEED)
      IMPLICIT real * 8 (AH, OZ), Integer * 4 (IN)
C ***********************************************
**********************
C PORTABLE congruential (uniforme) RANDOM gerador de números:
NEXT_VALUE C = [(7 ** 5) * PREVIOUS_VALUE] MODULO [(2 ** 31) -1]
C
C Este gerador é composto por duas ROTINAS:
C (1) SETRAN - inicializa constantes e SEED
C (2) RRAN - gera um número aleatório REAIS
C
C O GERADOR DE RODA uma máquina com pelo menos 32 bits de precisão.
C DA SEED (ISEED) deve estar no intervalo (1, (2 ** 31 -1)).
C ***********************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      IF (ISEED.LT.1) PARADA 77
      MULT = 16807
      MODUL = 2147483647
```

```
      I15 = 2 ** 15
      I16 = 2 ** 16
      JRAN = ISEED
      RETURN
      FIM
C
      RAN função real ()
      IMPLICIT real * 4 (AH, OZ), Integer * 4 (IN)
C ***********************************************
**********************
C RAN gera um número aleatório real entre 0 e 1
C ***********************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      Ixhi = JRAN / I16
      IXLO = JRAN-Ixhi * I16
      IXALO = IXLO * MULT
      LEFTLO = IXALO / I16
      IXAHI = Ixhi * MULT
      IFULHI = IXAHI + LEFTLO
      IRTLO = IXALO-LEFTLO * I16
      IOVER = IFULHI / I15
      IRTHI = IFULHI-IOVER * I15
      JRAN = ((IRTLO-MODUL) + IRTHI * I16) + IOVER
      IF (JRAN.LT.0) JRAN = JRAN + MODUL
      RAN = FLOAT (JRAN) / FLOAT (MODUL)
      RETURN
      FIM
```

NGCONVERT: NETGEN a conversão de formato Stardard

Há no público
domínio de um gerador de problema rede aleatória popular chamado
NETGEN [KNS74]. Este
gerador utiliza um formato de representação problema que é diferente
a forma padrão descrito anteriormente. Para o benefício de um leitor
que tem acesso
para NETGEN, nós fornecemos o NGCONVERT código de conversão, que lê um
problema no
o formato NETGEN e grava-lo para o formato padrão.

```
C ******************* programa de conversão *****************
C
C Este programa irá ler um arquivo problema criado VIA
C a Random PROBLEMA GERADOR NETGEN OU QUALQUER gerador que
C usa o formato NETGEN, e convertê-lo para o padrão
C FORMATO.
C
C ********************************************** ***
C
      PROGRAMA NGCONVERT
      NONE IMPLICIT
      INTEGER START (45000), END (45000), COST (45000), CAP (45000)
      FORNECIMENTO inteiro (15000)
      INTEGER I, N, IA, NA, NSORC, NSINK
C

      PRINT *, "leitura NETGEN problema de dados '
```

```
      ABERTO (12, FILE = "FOR012.DAT ', STATUS =' velho ')
      Rewind (12)
      LER (12,1020) N, IA, NSORC, NSINK

      FAZER 10 I = 1, IA
        READ (12,1020) START (I), END (I), COST (I), CAP (I)
10 CONTINUAR

      FAZER 20 I = 1, N
        FORNECIMENTO (I) = 0
20 CONTINUAR

      NA = 0
      FAZER 25 I = 1, IA
        IF (START (I) .EQ.N + 1) THEN
          SUPPLY (END (I)) = CAP (I)
        MAIS
          IF (END (I) .EQ.N + 2) THEN
            SUPPLY (START (I)) = - CAP (I)
          MAIS
            NA NA + 1 =
            COST (NA) = CUSTO (I)
            CAP (NA) = CAP (I)
            START (NA) = START (I)
            END (NA) = END (I)
          END IF
        END IF
25 CONTINUAR

      PRINT *, 'Escrevendo o problema Em DISK na forma padrão'
      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' Novo ')
      Rewind (13)

C ESCREVA número de nós e arcos

      ESCREVA (13,1010) N, NA

C inicial de gravação, END, custo e capacidade de cada ARC

      FAZER 30 I = 1, NA
        ESCREVA (13,1020) START (I), END (I), COST (I), CAP (I)
30 CONTINUAR

C ESCREVA SUPPLY de cada nó

      DO 40 I = 1, N
        ESCREVA (13,1000) SUPPLY (I)
40 CONTINUAR

      ENDFILE (13)
      Rewind (13)

      PRINT *, "fim da escrita"
      PARE

1000 FORMATO (1I8)
1010 FORMATO (2I8)
1020 FORMATO (4I8)

      FIM
```

```
************************************************
************************
```

Curtos Path Codes

```
************************************************
************************
```

PAPE-ALL-DEST

Este código implementa uma variação do D'Esopo-Pape
origem single / método todos os destinos rotular correção
(Cf. \ Seção 1.3.3), dada em [Pal84] e [GaP88]. O código é adaptado a
partir de
o código de L2QUEUE [GaP88].

```
C ***** AMOSTRA PROGRAMA CHAMADA PARA A SUBROUTINE L2QUEUE *****
C *** (menor caminho) ***
C *** ***
C *** O PROGRAMA É com base no Livro ***
C *** G. GALLO, S. Pallottino "algoritmos de menor caminho", ***
C *** ANAIS DO Operations Research 7, 1988. ***
C *** ***
C *** ENTRADA E SAÍDA DE PARTES modificado e ***
C *** A ESTRUTURA FOUT dados inseridos pela ***
C *** DIMITRI P. Bertsekas ***
C *** ***
C *** perguntas e comentários devem ser dirigidas ao ***
C *** G. Gallo e S. Pallottino ***
C *** DEPT. DE COMP. SCI., UNIV. De Pisa, Italia. ***
C *** FAX 39-50-510247 ***
C *** EMAIL: PALLO@DIPISA.DI.UNIPI.IT ***
C ********************************************** ***********

C ********************************************** ***********
C ENCONTRA Shortest Path DE UMA ORIGEM para todos os destinos
C ********************************************** ***********
C
C Todos os parâmetros são INTEIRO
C
C a única máquina CONSTANTE dependente usada É INF
C
C ***********************************************
**********************
```

```
      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      INTEGER FOUT, NXTOUT, D, P, Q, R, HP, Y, X, T2, DEST
      TT1 REAL, TT2, TCOST
      DIMENSÃO FOUT (maxNodes), D (maxNodes), P (maxNodes), HP
(maxNodes)
      DIMENSÃO Q (maxNodes)
      DIMENSÃO LNGT (MAXARCS), ND (MAXARCS), NXTOUT (MAXARCS)
      DATA INF / 999999999 /

      CHAMADA READ (N, M, FOUT, NXTOUT, ND, LNGT)

      FAZER 10 I = 1, N
```

```
        Q (I) = 0
        P (I) = 0
10 CONTINUAR
C
C REINICIAR ORIGEM E FILA DE DESTINOS
C

      PRINT *, "o número de nós IS = ', N
      PRINT *, "ENTER o nó origem»
      LEIA *, R



      PRINT *, "CHAMANDO DESOPO-PAPE para resolver o problema"
      PRINT *, "****************************************** '
C
C GET hora de início para o Mac II
C
      TT1 = longa (362) /60.0
C

        CHAMAR L2QUE (FOUT, NXTOUT, ND, LNGT, D, P, Q, N, INF, R)
C
C GET TÉRMINO TEMPO PARA O MAC II
C
      TT2 = longa (362) /60.0 - TT1
      PRINT *, "acabado --- tempo de CPU TOTAL ', TT2,' s '
      PRINT *, "****************************************** '
      PRINT *, "O nó ORIGEM É ', R
C

      PRINT *, 'SH. DIST. De destino », N, 'é', D (N)

45 PRINT *, "entrar em outros nó de destino (0 se for feito)"
      LEIA *, DEST
      IF ((DEST.LT.0) .OR. (DEST.GT.N)) GOTO 45
      IF (DEST.EQ.R) GOTO 45
      IF (DEST.NE.0) THEN
        PRINT *, 'distância mais curta para', DEST, '=', D (DEST)
        GOTO 45
      END IF



      PARE
      FIM
C
C
      SUBRUTINA L2QUE (FOUT, NXTOUT, ND, LNGT, D, P, Q, N, INF, R)
C ********************************************** ******************
C
C L2QUE ROTINA
C
C 1) encontra um MENOR caminho árvore com raiz em NODE R eo menor
C DISTÂNCIAS
C 2) é baseado no método D'ESOPO-PAPE com o conjunto Q REALIZAR
C como um duplo FILA Q (.)
C
C significado dos parâmetros de entrada:
C
C FOUT (I) = POINTER de Arco-LIST do nó i, i = 1,2, ..., N + 1
```

```
C ND (J) = TÉRMINO NÓ DE ARC J, J = 1,2, ..., M
C LNGT (J) = comprimento do arco J, J = 1,2, ..., M
C NMAX = dimensão do ARRAYS A (.), D (.), P (.), Q (.)
C Mmáx = dimensão do ARRAYS ND (.), LNGT (.)
CN = número de nós
C INF = MUITO GRANDE VALOR INTEIRO (INFINITY)
CR = ROOT
C
C significado dos parâmetros de saída:
C
CD (I) = distância mais curta de R a I, I = 1,2, ..., N
CP (I) = antecessor nó I no menor caminho árvore, I = 1,2, ..., N
C
C DOS PARÂMETROS interna principal:
C
CQ (I) = lista de nós candidatos; Q (I) = -1 SE EU NÃO ESTÁ EM Q e tem
C JÁ sido digitalizados
C = 0 SE EU NÃO ESTÁ EM Q e tem
C não foi digitalizado
C = J IF I PRECEDE NÓ NA J
LISTA C
C = NN N + 1
CU = nó atual
CV = ENDING do nó da ARC CURRENT
C INIT = START-ponteiro para a ARC-LIST do nó atual
C IFIN = END-ponteiro para a ARC-LIST do nó atual
C DV = LABEL TENTATIVA DE NÓ V
C LAST = ponteiro para o último nó Q (.)
C PNTR = ponteiro para o último nó da fila PRIMEIRO DE Q (.)
C
C Todos os parâmetros são INTEIRO
C
C ********************************************** ******************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      INTEGER FOUT, NXTOUT, D, P, Q, R, U, V, DV, PNTR
      DIMENSÃO FOUT (maxNodes), D (maxNodes), P (maxNodes), Q
(maxNodes)
      DIMENSÃO ND (MAXARCS), LNGT (MAXARCS), NXTOUT (MAXARCS)
C
C INITIALIZE
C
      FAZER 10 I = 1, N
        Q (I) = 0
        D (I) = INF
  10 CONTINUAR
      Q (R) = - 1
      D (R) = 0
      P (R) = 0
      NN = N + 1
      Q (NN) = NN
      LAST = NN
      PNTR = NN
      U = R
C
C
C EXPLORAR A ESTRELA DE FRENTE U
C
20 CONTINUAR
```

```
      J = FOUT (L)
25 IF (J.GT.0) THEN
        V = ND (J)
        DV = D (L) + LNGT (J)
C
C Verifique se o rótulo de V PODE SER MELHORADO
C
        IF (D (V) .gt. DV) THEN
          D (V) = DV
          P (V) = L
          IF (Q (V)) 30,40,50
C
C IF V NÃO ESTÁ EM Q E já foi digitalizado, ele é inserido no
C POSIÇÃO apontado por PNTR
C
30 Q (V) = Q (PNTR)
          Q (PNTR) = V
          IF (LAST .EQ. PNTR) LAST = V
          V = PNTR
          IR PARA 50
C
C IF V NÃO ESTÁ EM Q e foi NUNCA digitalizado, ele é inserido na cauda
C DO Q
C
40 Q (LAST) = V
          Q (V) = NN
          LAST = V
50 CONTINUAR
        END IF
        J = NXTOUT (J)
        GOTO 25
       END IF
C
C Remova a nova corrente NODE U
C
60 U = Q (NN)
       Q (NN) = Q (U)
       Q (U) = - 1
       IF (LAST .EQ. U) LAST = NN
       IF (PNTR .EQ. U) PNTR = NN
C
C Verifique se a lista estiver vazia
C
       IF (U .LE. N) Vá para 20

       RETURN
       FIM

      SUBROUTINE READ (N, M, FOUT, NXTOUT, END, LNGT)
C ***********************************************
******************
C lê os dados do gráfico (armazenado como uma lista adjacence) e as
origens
LISTA C.
C ********************************************** ******************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      Implícita INTEGER (AZ)
```

```
      DIMENSÃO FOUT (maxNodes), LAST (maxNodes)
      DIMENSÃO START (MAXARCS), END (MAXARCS), LNGT (MAXARCS)
      DIMENSÃO NXTOUT (MAXARCS)
C
      PRINT *, "leitura mais curto DATA PATH problema"
      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' velho ')
      Rewind (13)

C LEIA número de nós e arcos

      LER (13,1010) N, M

C LEIA ENDNODE eo comprimento de cada ARC

      FAZER 20 I = 1, M
        READ (13,1020) START (I), END (I), LNGT (I), MANEQUIM
20 CONTINUAR

      FAZER 30 I = 1, N
        READ (13,1000) MANEQUIM
30 CONTINUAR

      ENDFILE (13)
      Rewind (13)

      PRINT *, 'FIM DE LEITURA'

1000 FORMATO (1I8)
1010 FORMATO (2I8)
1020 FORMATO (4I8)


      PRINT *, "reestruturar os dados '
C
C GERAR A FRENTE E ESTRELAS com versões anteriores para cada nó
C
      FAZER 60 I = 1, N
        FOUT (I) = 0
        LAST (I) = 0
60 CONTINUAR

      FAZER 65 ARC = 1, M
        NXTOUT (ARC) = 0
        NÓ = START (ARC)
        IF (FOUT (nó) .NE.0) THEN
          NXTOUT (LAST (nó)) = ARC
        MAIS
          FOUT (nó) = ARC
        END IF
        LAST (nó) = ARC
65 CONTINUAR

      RETURN

      FIM
```

MONTÃO-ALL-DEST

Esta é uma implementação da origem única / all

destinos rotular método de definição, usando um heap binário para manter a
lista de candidatos (cf. \ Seção 1.3.2). Este código é adaptado do sheap
código de Gallo e Pallotino [GaP88].

```
C ******* programa de amostra CHAMADA PARA SUBROUTINE sheap *******
C *** (menor caminho) ***
C *** ***
C *** O PROGRAMA É com base no Livro ***
C *** G. GALLO, S. Pallottino "algoritmos de menor caminho", ***
C *** ANAIS DO Operations Research 7, 1988. ***
C *** ***
C *** ENTRADA E SAÍDA DE PARTES modificado e ***
C *** A ESTRUTURA FOUT dados inseridos pela ***
C *** DIMITRI P. Bertsekas ***
C *** ***
C *** perguntas e comentários devem ser dirigidas ao ***
C *** G. Gallo e S. Pallottino ***
C *** DEPT. DE COMP. SCI., UNIV. De Pisa, Italia. ***
C *** FAX 39-50-510247 ***
C *** EMAIL: PALLO@DIPISA.DI.UNIPI.IT ***
C ********************************************** ***********

C ********************************************** ***********
C ENCONTRA Shortest Path DE UMA ORIGEM para todos os destinos
C ********************************************** ***********
C
C Todos os parâmetros são INTEIRO
C
C a única máquina CONSTANTE dependente usada É INF
C
C **********************************************
**********************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      INTEGER FOUT, NXTOUT, D, P, Q, R, HP, Y, X, T2, DEST
      TT1 REAL, TT2, TCOST
      DIMENSÃO FOUT (maxNodes), D (maxNodes), P (maxNodes)
      DIMENSÃO HP (maxNodes)
      DIMENSÃO Q (maxNodes)
      DIMENSÃO LNGT (MAXARCS), ND (MAXARCS), NXTOUT (MAXARCS)
      DATA INF / 999999999 /

      CHAMADA READ (N, M, FOUT, NXTOUT, ND, LNGT)

      FAZER 10 I = 1, N
        Q (I) = 0
        P (I) = 0
10 CONTINUAR
C
C REINICIAR ORIGEM E FILA DE DESTINOS
C

      PRINT *, "o número de nós IS = ', N
      PRINT *, "ENTER o nó origem»
      LEIA *, R
```

```
      PRINT *, "CHAMANDO DIJKSTRA / MONTÃO para resolver o problema"
      PRINT *, "****************************************** '
C
C GET hora de início para o Mac II
C
      TT1 = longa (362) /60.0
C

      CHAME sheap (FOUT, NXTOUT, ND, LNGT, D, P, Q, HP, N, INF, R)
C
C GET TÉRMINO TEMPO PARA O MAC II
C
      TT2 = longa (362) /60.0 - TT1
      PRINT *, "acabado --- tempo de CPU TOTAL ', TT2,' s '
      PRINT *, "****************************************** '
      PRINT *, "O nó ORIGEM É ', R
C


      PRINT *, "a distância mais curta até ao destino", N, '=', D (N)

45 PRINT *, "entrar em outros nó de destino (0 se for feito)"
      LEIA *, DEST
      IF ((DEST.LT.0) .OR. (DEST.GT.N)) GOTO 45
      IF (DEST.EQ.R) GOTO 45
      IF (DEST.NE.0) THEN
        PRINT *, 'distância mais curta para', DEST, '=', D (DEST)
        GOTO 45
      END IF


      PARE
      FIM
C
C
      SUBRUTINA sheap (FOUT, NXTOUT, ND, LNGT, D, P, Q, HP, N, INF, R)

C ***********************************************
**********************
C
C ROTINA sheap
C
C 1) encontra um MENOR caminho árvore com raiz em NODE R eo menor
C DISTÂNCIAS
C 2) é baseado no MÉTODO DIJKSTRA, COM prioridade da fila Q REALIZAR
C COMO UM MONTÃO BINÁRIO
C
C significado dos parâmetros de entrada:
C
C FOUT (I) = POINTER de Arco-LIST do nó i, i = 1,2, ..., N + 1
C ND (J) = TÉRMINO NÓ DE ARC J, J = 1,2, ..., M
C LNGT (J) = comprimento do arco J, J = 1,2, ..., M
CN = número de nós
C INF = MUITO GRANDE VALOR INTEIRO (INFINITY)
CR = ROOT
C
C significado dos parâmetros de saída:
C
CD (I) = distância mais curta de R a I, I = 1,2, ..., N
CI no menor caminho árvore, I = 1,2, ..., N
```

```
C
C significado dos parâmetros MAIN INTERNOS:
C
CQ (I) = DICIONÁRIO DO MONTÃO: Q (I) dá a posição do nodo
CI no heap HP (.), I = 1,2, ..., N
C HP (I) = nó i-TH no heap, I = 1,2, ..., NHP
C NHP = número de nós na pilha (NHP <= N)
C = NN N + 1
CU = nó atual
CV = ENDING do nó da ARC CURRENT
C INIT = START-ponteiro para a ARC-LIST do nó atual
C IFIN = END-ponteiro para a ARC-LIST do nó atual
C DV = LABEL TENTATIVA DE NÓ V
C
C Todos os parâmetros são INTEIRO
C
C ***********************************************
**********************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      INTEGER FOUT, D, P, Q, R, U, V, DV, HP, DP1
      INTEGER HP1, HP2, HP3
      DIMENSÃO FOUT (maxNodes), D (maxNodes), P (maxNodes)
      DIMENSÃO Q (maxNodes), HP (maxNodes)
      DIMENSÃO ND (MAXARCS), LNGT (MAXARCS), NXTOUT (MAXARCS)
C
C INITIALIZE
C
      FAZER 10 I = 1, N
        D (I) = INF
10 CONTINUAR
      NHP = 0
      D (R) = 0
      P (R) = 0
      NN = N + 1
      U = R
C
C EXPLORAR A ESTRELA DE FRENTE U
C
20 CONTINUAR

      J = FOUT (L)
25 IF (J.GT.0) THEN
        V = ND (J)
        DV = D (L) + LNGT (J)
C
C Verifique se o rótulo de V PODE SER MELHORADO
C
        IF (D (V) .gt. DV) THEN
          D (V) = DV
          P (V) = L
          IF (Q (V) .NE. 0) Ir para 30
C
C Inserir nó V na pilha
C
          NHP = NHP + 1
          Q (V) = NHP
C
C atualizar o MONTÃO
```

```
C
30 K = Q (V)
40 K2 = K / 2
           IF (K2 .LE. 0) Ir para 50
           HP2 = HP (K2)
           IF (DV .GE. D (HP2)) Vá para 50
           HP (K) = HP2
           Q (HP2) = K
           K = K2
           GOTO 40
50 HP (K) = V
           Q (V) = K
         END IF
         J = NXTOUT (J)
         GOTO 25
        END IF
C
C Remova a nova corrente NODE U do heap
C Verifique se é o destino
C
70 U = HP (1)
      Q (U) = 0
      NHP = NHP - 1
C
C Verifique se o MONTÃO ESTÁ VAZIO
C
      IF (NHP) 130,20,80
C
C atualizar o MONTÃO
C
80 HP1 = HP (NHP + 1)
      DP1 = D (HP1)
      K = 1
90 K2 = 2 * K
      HP2 = HP (K2)
      IF (K2-PNC) 100110120
100 HP3 = HP (K2 + 1)
      IF (D (HP2) .lt. D (HP3)) Vá para 110
      HP2 = HP3
      K2 K2 + 1 =
110 IF (DP1 .LE. D (HP2)) Vá para 120
      HP (K) = HP2
      Q (HP2) = K
      K = K2
      IR PARA 90
120 HP (K) = HP1
      Q (HP1) = K
      IR PARA 20
130 CONTINUAR
      RETURN
      FIM


      SUBROUTINE READ (N, M, FOUT, NXTOUT, END, LNGT)
C ***********************************************
******************
C lê os dados do gráfico (armazenado como uma lista adjacence) e as
origens
LISTA C.
C ********************************************** ******************
```

```
      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      Implícita INTEGER (AZ)
      DIMENSÃO FOUT (maxNodes), LAST (maxNodes)
      DIMENSÃO START (MAXARCS), END (MAXARCS), LNGT (MAXARCS)
      DIMENSÃO NXTOUT (MAXARCS)
C
      PRINT *, "leitura mais curto DATA PATH problema"
      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' velho ')
      Rewind (13)

C LEIA número de nós e arcos

      LER (13,1010) N, M

C LEIA ENDNODE eo comprimento de cada ARC

      FAZER 20 I = 1, M
        READ (13,1020) START (I), END (I), LNGT (I), MANEQUIM
20 CONTINUAR

      FAZER 30 I = 1, N
        READ (13,1000) MANEQUIM
30 CONTINUAR

      ENDFILE (13)
      Rewind (13)

      PRINT *, 'FIM DE LEITURA'

1000 FORMATO (1I8)
1010 FORMATO (2I8)
1020 FORMATO (4I8)


      PRINT *, "reestruturar os dados '
C
C GERAR A FRENTE E ESTRELAS com versões anteriores para cada nó
C
      FAZER 60 I = 1, N
        FOUT (I) = 0
        LAST (I) = 0
60 CONTINUAR

      FAZER 65 ARC = 1, M
        NXTOUT (ARC) = 0
        NÓ = START (ARC)
        IF (FOUT (nó) .NE.0) THEN
          NXTOUT (LAST (nó)) = ARC
        MAIS
          FOUT (nó) = ARC
        END IF
        LAST (nó) = ARC
65 CONTINUAR

      RETURN

      FIM
```

MONTÃO-SELECT-DEST

Este código resolve o problema do caminho mais curto com uma
origem única e um número selecionado de destinos. É o mesmo que o
código anterior MONTÃO-ALL-DEST, exceto que ele pára quando todos os
destinos têm
foi etiquetada de forma permanente.

```
C ******* programa de amostra CHAMADA PARA SUBROUTINE sheap *******
C *** (menor caminho) ***
C *** ***
C *** O PROGRAMA É com base no Livro ***
C *** G. GALLO, S. Pallottino "algoritmos de menor caminho", ***
C *** ANAIS DO Operations Research 7, 1988. ***
C *** ***
C *** ENTRADA E SAÍDA DE PARTES modificado e ***
C *** A ESTRUTURA FOUT dados inseridos pela ***
C *** DIMITRI P. Bertsekas ***
C *** ***
C *** perguntas e comentários devem ser dirigidas ao ***
C *** G. Gallo e S. Pallottino ***
C *** DEPT. DE COMP. SCI., UNIV. De Pisa, Italia. ***
C *** FAX 39-50-510247 ***
C *** EMAIL: PALLO@DIPISA.DI.UNIPI.IT ***
C ********************************************** ***********

C ********************************************** ***********
C ENCONTRA Shortest Path DE UMA ORIGEM para vários destinos
C ********************************************** ***********
C
C Todos os parâmetros são INTEIRO
C
C a única máquina CONSTANTE dependente usada É INF
C
C **********************************************
**********************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      INTEGER FOUT, D, P, Q, R, HP, Y, X, T2, DEST_COUNT, SH_DIST,
DEST
      TT1 REAL, TT2, TCOST
      DIMENSÃO FOUT (maxNodes), D (maxNodes), P (maxNodes)
      DIMENSÃO HP (maxNodes)
      DIMENSÃO Q (maxNodes), SH_DIST (maxNodes)
      DIMENSÃO LNGT (MAXARCS), ND (MAXARCS), NXTOUT (MAXARCS)
      DATA INF / 999999999 /

      CHAMADA READ (N, M, FOUT, NXTOUT, ND, LNGT)

      FAZER 10 I = 1, N
        Q (I) = 0
        P (I) = 0
        SH_DIST (I) = - 1
10 CONTINUAR
C
C REINICIAR ORIGEM E FILA DE DESTINOS
C
```

```
      PRINT *, "o número de nós IS = ', N
      PRINT *, "ENTER o nó origem»
      LEIA *, R


42 PRINT *, 'entre primeiro nó de destino'
      LEIA *, DEST

      IF ((DEST.LE.0) .OR. (DEST.GT.N)) GOTO 42
      IF (DEST.EQ.R) GOTO 42


      IF ((DEST.LE.0) .OR. (DEST.GE.N)) DEST = N
      SH_DIST (DEST) = 0
      DEST_COUNT = 1

45 PRINT *, "ENTER nó de destino NEXT (0 se for feito)"
      LEIA *, DEST
      IF ((DEST.LT.0) .OR. (DEST.GT.N) .OR. (DEST.EQ.R)) GOTO 45
      IF (DEST.NE.0) THEN
        IF (SH_DIST (DEST) .LT.0) THEN
          DEST_COUNT = + 1 DEST_COUNT
          SH_DIST (DEST) = 0
        END IF
        GOTO 45
      END IF

50 CONTINUAR

      PRINT *, 'NÚMERO TOTAL DE DESTINOS =', DEST_COUNT


      PRINT *, "CHAMANDO DIJKSTRA / MONTÃO para resolver o problema"
      PRINT *, "****************************************** '
C
C GET hora de início para o Mac II
C
      TT1 = longa (362) /60.0
C

      CHAMAR sheap (FOUT, NXTOUT, ND, LNGT, D, P, Q, HP, N, INF, R,
     & DEST_COUNT, SH_DIST)
C
C GET TÉRMINO TEMPO PARA O MAC II
C
      TT2 = longa (362) /60.0 - TT1
      PRINT *, "acabado --- tempo de CPU TOTAL ', TT2,' s '
      PRINT *, "****************************************** '
      PRINT *, "O nó ORIGEM É ', R
C

        DO 200 I = 1, N
          IF (SH_DIST (I) .GE.0) THEN
            IF (SH_DIST (I) .NE.D (I)) THEN
              PRINT *, 'ERRO'
            END IF
            PRINT *, 'SH. DIST. De destino », I, 'é', SH_DIST (I)
          END IF
200 CONTINUAR
```

```
      PRINT *, "****************************************** '
      PRINT *, 'programa terminou; PRESS <CR> '
      PAUSA
      PARE
      FIM
C
C
      SUBRUTINA sheap (FOUT, NXTOUT, ND, LNGT, D, P, Q, HP, N, INF, R,
     & DEST_COUNT, SH_DIST)
C ***********************************************
**********************
C
C ROTINA sheap
C
C 1) encontra um MENOR caminho árvore com raiz em NODE R eo menor
C DISTÂNCIAS
C 2) é baseado no MÉTODO DIJKSTRA, COM prioridade da fila Q REALIZAR
C COMO UM MONTÃO BINÁRIO
C
C significado dos parâmetros de entrada:
C
C FOUT (I) = POINTER de Arco-LIST do nó i, i = 1,2, ..., N + 1
C ND (J) = TÉRMINO NÓ DE ARC J, J = 1,2, ..., M
C LNGT (J) = comprimento do arco J, J = 1,2, ..., M
CN = número de nós
C INF = MUITO GRANDE VALOR INTEIRO (INFINITY)
CR = ROOT
C
C significado dos parâmetros de saída:
C
CD (I) = distância mais curta de R a I, I = 1,2, ..., N
CI no menor caminho árvore, I = 1,2, ..., N
C
C significado dos parâmetros MAIN INTERNOS:
C
CQ (I) = DICIONÁRIO DO MONTÃO: Q (I) dá a posição do nodo
CI no heap HP (.), I = 1,2, ..., N
C HP (I) = nó i-TH no heap, I = 1,2, ..., NHP
C NHP = número de nós na pilha (NHP <= N)
C = NN N + 1
CU = nó atual
CV = ENDING do nó da ARC CURRENT
C INIT = START-ponteiro para a ARC-LIST do nó atual
C IFIN = END-ponteiro para a ARC-LIST do nó atual
C DV = LABEL TENTATIVA DE NÓ V
C
C Todos os parâmetros são INTEIRO
C
C ***********************************************
**********************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      INTEGER FOUT, D, P, Q, R, U, V, DV, HP, DP1
      INTEGER HP1, HP2, HP3, DEST_COUNT, SH_DIST
      DIMENSÃO FOUT (maxNodes), D (maxNodes), P (maxNodes)
      DIMENSÃO Q (maxNodes), HP (maxNodes)
      DIMENSÃO ND (MAXARCS), LNGT (MAXARCS), NXTOUT (MAXARCS)
      DIMENSÃO SH_DIST (maxNodes)
C
```

```
C INITIALIZE
C
      FAZER 10 I = 1, N
        D (I) = INF
10 CONTINUAR
      NHP = 0
      D (R) = 0
      P (R) = 0
      NN = N + 1
      U = R
C
C EXPLORAR A ESTRELA DE FRENTE U
C
20 CONTINUAR

      J = FOUT (L)
25 IF (J.GT.0) THEN
        V = ND (J)
        DV = D (L) + LNGT (J)
C
C Verifique se o rótulo de V PODE SER MELHORADO
C
        IF (D (V) .gt. DV) THEN
          D (V) = DV
          P (V) = L
          IF (Q (V) .NE. 0) Ir para 30
C
C Inserir nó V na pilha
C
          NHP = NHP + 1
          Q (V) = NHP
C
C atualizar o MONTÃO
C
30 K = Q (V)
40 K2 = K / 2
          IF (K2 .LE. 0) Ir para 50
          HP2 = HP (K2)
          IF (DV .GE. D (HP2)) Vá para 50
          HP (K) = HP2
          Q (HP2) = K
          K = K2
          GOTO 40
50 HP (K) = V
          Q (V) = K
        END IF
        J = NXTOUT (J)
        GOTO 25
       END IF
C
C Remova a nova corrente NODE U do heap
C verifique se é um destino
C
70 U = HP (1)
      IF (SH_DIST (U) .GE.0) THEN
        SH_DIST (L) = D (L)
        DEST_COUNT = DEST_COUNT-1
        IF (DEST_COUNT.EQ.0) GOTO 130
      END IF
      Q (U) = 0
      NHP = NHP - 1
```

```
C
C Verifique se o MONTÃO ESTÁ VAZIO
C
      IF (NHP) 130,20,80
C
C atualizar o MONTÃO
C
80 HP1 = HP (NHP + 1)
      DP1 = D (HP1)
      K = 1
90 K2 = 2 * K
      HP2 = HP (K2)
      IF (K2-PNC) 100110120
100 HP3 = HP (K2 + 1)
      IF (D (HP2) .lt. D (HP3)) Vá para 110
      HP2 = HP3
      K2 K2 + 1 =
110 IF (DP1 .LE. D (HP2)) Vá para 120
      HP (K) = HP2
      Q (HP2) = K
      K = K2
      IR PARA 90
120 HP (K) = HP1
      Q (HP1) = K
      IR PARA 20
130 CONTINUAR
      RETURN
      FIM


      SUBROUTINE READ (N, M, FOUT, NXTOUT, END, LNGT)
C ***********************************************
******************
C lê os dados do gráfico (armazenado como uma lista adjacence) e as
origens
LISTA C.
C ********************************************** ******************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      Implícita INTEGER (AZ)
      DIMENSÃO FOUT (maxNodes), LAST (maxNodes)
      DIMENSÃO START (MAXARCS), END (MAXARCS), LNGT (MAXARCS)
      DIMENSÃO NXTOUT (MAXARCS)
C
      PRINT *, "leitura mais curto DATA PATH problema"
      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' velho ')
      Rewind (13)

C LEIA número de nós e arcos

      LER (13,1010) N, M

C LEIA ENDNODE eo comprimento de cada ARC

      FAZER 20 I = 1, M
        READ (13,1020) START (I), END (I), LNGT (I), MANEQUIM
20 CONTINUAR

      FAZER 30 I = 1, N
```

```
        READ (13,1000) MANEQUIM
30 CONTINUAR

      ENDFILE (13)
      Rewind (13)

      PRINT *, 'FIM DE LEITURA'

1000 FORMATO (1I8)
1010 FORMATO (2I8)
1020 FORMATO (4I8)


      PRINT *, "reestruturar os dados '
C
C GERAR A FRENTE E ESTRELAS com versões anteriores para cada nó
C
      FAZER 60 I = 1, N
        FOUT (I) = 0
        LAST (I) = 0
60 CONTINUAR

      FAZER 65 ARC = 1, M
        NXTOUT (ARC) = 0
        NÓ = START (ARC)
        IF (FOUT (nó) .NE.0) THEN
          NXTOUT (LAST (nó)) = ARC
        MAIS
          FOUT (nó) = ARC
        END IF
        LAST (nó) = ARC
65 CONTINUAR

      RETURN

      FIM



LEILÃO-SELECT-DEST

Este código também resolve o problema do caminho mais curto
com uma única origem e um número seleccionado de destinos. Ele
implementa o
combinado para a frente / trás algoritmo leilão da Seção 4.3.

C ********************************************** ***********
C *** AMOSTRA PROGRAMA CHAMADA PARA A SUBROUTINE AUCTION_SP ***
C *** (PARA A FRENTE E VERSÃO PARA TRÁS) ***
C *** MENOR PROBLEMA caminho de um ORIGEM A UMA SELECIONADOS ***
C *** JOGO DE DESTINOS ***
C *** assumindo todos os comprimentos de arco são não negativos ***
C *** ***
C *** por Dimitri Bertsekas, agosto '90 ***
C *** lê arquivos problema no FORMULÁRIO DE ***
C *** ***
C ********************************************** ***********


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)
```

```
      NONE IMPLICIT
      INTEGER FOUT, P, LNGT, PRD, END, R, DEST, INICIAR, T2, grande,
I, N, NA
      INTEGER PRDARC, COUNT, ITER, TERM, NUM_ORIG, NUM_EXT, NUM_SUCC
      INTEGER DPRDARC, FIN, NXTIN, NXTOUT, DTERM, DCOUNT
      INTEGER EXTEND_ARC, DEXTEND_ARC, NODE, ARC
      INTEGER FRSTQUEUE, NXTQUEUE, SH_DIST, CUR_DEST
      INTEGER PrevNode, DEST_COUNT, MANEQUIM
      Real * 8 TT1, TT2

      Common / escalares / N, grande, R, DEST
      Common / STATS / ITER, NUM_ORIG, NUM_EXT, NUM_SUCC
      COMUM / BLK1 / START
      COMUM / BLK2 / END
      COMUM / comprimento / LNGT
      COMUM / FRSTOUT / FOUT
      COMUM / FRSTIN / FIN
      COMUM / NEXTOUT / NXTOUT
      COMUM / NEXTIN / NXTIN
      Common / PREÇOS / P
      COMUM / PREDARC / PRDARC
      COMUM / PRECARC / DPRDARC
      COMUM / EXTARC / EXTEND_ARC
      COMUM / DEXTARC / DEXTEND_ARC
      COMUM / DTERM / FRSTQUEUE, PrevNode, DEST_COUNT
      COMUM / NXTQUEUE / NXTQUEUE
      COMUM / SH_DIST / SH_DIST
      DIMENSÃO FOUT (maxNodes), NXTOUT (MAXARCS), P (maxNodes), LNGT
(MAXARCS)
      DIMENSÃO START (MAXARCS), END (MAXARCS), PRDARC (maxNodes)
      DIMENSÃO EXTEND_ARC (maxNodes), DEXTEND_ARC (maxNodes)
      DIMENSÃO FIN (maxNodes), NXTIN (MAXARCS), DPRDARC (maxNodes)
      DIMENSÃO NXTQUEUE (maxNodes), SH_DIST (maxNodes)

      PRINT *, 'LEILÃO / short. CAMINHOS (1 orig., Vários DEST). '
      PRINT *, "****************************************** '
      PRINT *, 'READING problema de dados'

      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' velho ')
      Rewind (13)

C LEIA número de nós e arcos

      LER (13,1010) N, NA

C LEIA início, fim, custo e capacidade de cada ARC

      FAZER 20 I = 1, NA
        READ (13,1020) START (I), END (I), LNGT (I), MANEQUIM
20 CONTINUAR

C LEIA SUPPLY de cada nó

      FAZER 30 I = 1, N
        READ (13,1000) MANEQUIM
30 CONTINUAR

      ENDFILE (13)
      Rewind (13)

      PRINT *, 'FIM DE LEITURA'
```

```
1000 FORMATO (1I8)
1010 FORMATO (2I8)
1020 FORMATO (4I8)


      PRINT *, "reestruturar os dados '
C
C GERAR A FRENTE E ESTRELAS com versões anteriores para cada nó
C
      FAZER 60 I = 1, N
        PRDARC (I) = 0
        FOUT (I) = 0
60 CONTINUAR

      FAZER 65 ARC = 1, NA
        NXTOUT (ARC) = 0
        NÓ = START (ARC)
        IF (FOUT (nó) .NE.0) THEN
          NXTOUT (PRDARC (nó)) = ARC
        MAIS
          FOUT (nó) = ARC
        END IF
        PRDARC (nó) = ARC
65 CONTINUAR


      DO 125 = I 1, N
        FIN (I) = 0
        PRDARC (I) = 0
125 CONTINUAR
      FAZER 130 ARC = 1, NA
        NXTIN (ARC) = 0
        NÓ = END (ARC)
        IF (FIN (nó) .NE.0) THEN
          NXTIN (PRDARC (nó)) = ARC
        MAIS
          FIN (nó) = ARC
        END IF
        PRDARC (nó) = ARC
130 CONTINUAR

C SET grande para um inteiro para a sua máquina
C
      GRANDE = 200000000
C

C REINICIAR PREÇOS E ARCS antecessor

      FAZER 140 I = 1, N
        P (I) = 0
        PRDARC (I) = - 1
        DPRDARC (I) = - 1
        EXTEND_ARC (I) = - 1
        DEXTEND_ARC (I) = - 1
        NXTQUEUE (I) = - 1
        SH_DIST (I) = - 1
140 CONTINUAR


C
```

```
C REINICIAR ORIGEM E FILA DE DESTINOS
C

      PRINT *, "o número de nós IS = ', N
150 PRINT *, "ENTER o nó origem»
      LEIA *, R
      IF ((R.LE.0) .OR. (R.GT.N)) GOTO 150

41 CONTINUAR


42 PRINT *, 'entre primeiro nó de destino'
      LEIA *, DEST

      IF ((DEST.LE.0) .OR. (DEST.GT.N)) GOTO 42
      IF (DEST.EQ.R) GOTO 42
      CUR_DEST = DEST
      FRSTQUEUE = DEST
      DEST_COUNT = 1
      NXTQUEUE (DEST) = 0

45 PRINT *, "ENTER nó de destino NEXT (0 se for feito)"
      LEIA *, DEST
      IF ((DEST.LT.0) .OR. (DEST.GT.N)) GOTO 45
      IF (DEST.EQ.R) GOTO 45
      IF (DEST.NE.0) THEN
        IF (NXTQUEUE (DEST) .LT.0) THEN
          NXTQUEUE (CUR_DEST) = DEST
          CUR_DEST = DEST
          DEST_COUNT = + 1 DEST_COUNT
          NXTQUEUE (DEST) = 0
        END IF
        GOTO 45
      MAIS
        NXTQUEUE (CUR_DEST) = FRSTQUEUE
        PrevNode = CUR_DEST
      END IF



50 CONTINUAR

      PRINT *, 'NÚMERO TOTAL DE DESTINOS =', DEST_COUNT

      PRINT *, "CHAMANDO MENOR LEILÃO PATH '
      PRINT *, "****************************************** '

C
C GET hora de início para o Mac II
C
      TT1 = longa (362) /60.0
      CHAMADA AUCTION_SP
C
C GET TÉRMINO TEMPO PARA O MAC II
C
      TT2 = longa (362) /60.0 - TT1
      PRINT *, "acabado --- tempo de CPU TOTAL ', TT2,' s '
      PRINT *, "****************************************** '
      PRINT *, "O nó ORIGEM É ', R
C
        DO 200 I = 1, N
```

```
          IF (SH_DIST (I) .GE.0) THEN
            PRINT *, 'SH. DIST. De destino », I, 'é', SH_DIST (I)
          END IF
200 CONTINUAR

      PRINT *, "****************************************** '

      COUNT = 0
      DCOUNT = 0
      FAZER 100 I = 1, N
        IF (PRDARC (I) .GT.0) COUNT = COUNT + 1
        IF (DPRDARC (I) .GT.0) DCOUNT = DCOUNT + 1
100 CONTINUAR

      PRINT *, '# nó terminal alcançado a partir de ORIGIN =', COUNT
      PRINT *, '# nó terminal alcançado a partir DESTINOS =', DCOUNT
X PRINT *, '# de extensões na origem =', NUM_ORIG
X PRINT *, '# DE TENTATIVAS DE EXTENSÃO ESPECULATIVO =', NUM_EXT
X PRINT *, '# DE SUCESSOS DE EXTENSÃO ESPECULATIVO =', NUM_SUCC
X PRINT *, '# DE NÓ SCANS =', ITER

      PRINT *, "****************************************** '
      PRINT *, "ENTER <0> para parar; <1> PARA FUNCIONAR COM OUTROS
DESTINOS '
      LEIA *, DEST
      IF (DEST.EQ.1) THEN
        DO 400 = I 1, N
          NXTQUEUE (I) = - 1
          SH_DIST (I) = - 1
400 CONTINUAR
        GOTO 41
      END IF


      PARE

      FIM
C
C


      AUCTION_SP SUBROUTINE
C ********************************************
**********************
C
C AUCTION_SP ROTINA
C
C encontra um caminho mais curto do nó R TO vários nós
C assumindo todos os comprimentos de arco são não negativos
C
C Significado das variáveis VÁRIOS:
C
C FOUT (I) = Primeiro OUTGOING ARC do nó I
C FIN (I) = Primeiro ENTRANTE ARC ao nó I
C NXTOUT (J) = PRÓXIMO ARC com o nó FIM MESMO AS J (J = 0 se é a
última)
C NXTIN (J) = PRÓXIMO ARC com o nó FIM MESMO AS J (J = 0 se é a
última)
C START (J) = COMEÇAR NÓ DE ARC J
C END (J) = TÉRMINO NÓ DE ARC J
C LNGT (J) = comprimento do arco J
```

```
CN = número de nós
CM = número de arcos
C GRANDE = MUITO GRANDE VALOR INTEIRO (INFINITY)
CR = ORIGEM
C = DEST destino actual
CP (I) = PREÇO DE I
C PRDARC (I) = antecessor do I NO CAMINHO A SEGUIR EM CURSO
C TERMO NODE = TERMINAL DE CAMINHO A SEGUIR EM CURSO
C DPRDARC (I) = antecessor do I NO CAMINHO PARA TRÁS DE CORRENTE
C DTERM = TERMINAL nó trajeto inverso CURRENT
C = ENDNODE ENDING do nó do ARC CURRENT
C = FRSTQUEUE primeiro nó NA FILA DE DESTINOS
C NXTQUEUE (I) = DESTINO AO LADO DO I NA FILA DE DESTINOS
C SH_DIST (I) = menor distância da origem ao nó i
C EXTEND_ARC (I) = provável candidato à ARC de saída do I ON caminho
mais curto
C DEXTEND_ARC (I) = provável candidato ENTRANTE ARC TO I ON caminho
mais curto
C
C todas as variáveis são INTEIRO
C
C ************************************************
**********************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      NONE IMPLICIT
      INTEGER FOUT, P, LNGT, PRD, SB, W, R, DEST, Grande, N, M
      TERMO INTEIRO, ENDNODE, INIT, IFIN, LEVEL, NEW_LEVEL
      INTEGER I, J, início, fim, ITER, NUM_ORIG, NUM_EXT, NUM_SUCC
      INTEGER EXTEND_ARC, BSTLEVEL, PTERM, PRDARC, EXTARC
      INTEGER DEXTEND_ARC, DTERM, DBSTLEVEL, DPTERM, DPRDARC, DEXTARC
      INTEGER FIN, NXTIN, NXTOUT, StartNode
      INTEGER FRSTQUEUE, NXTQUEUE, SH_DIST, PrevNode, DEST_COUNT

      Common / escalares / N, grande, R, DEST
      Common / STATS / ITER, NUM_ORIG, NUM_EXT, NUM_SUCC
      COMUM / BLK1 / START
      COMUM / BLK2 / END
      COMUM / comprimento / LNGT
      COMUM / FRSTOUT / FOUT
      COMUM / FRSTIN / FIN
      COMUM / NEXTOUT / NXTOUT
      COMUM / NEXTIN / NXTIN
      Common / PREÇOS / P
      COMUM / PREDARC / PRDARC
      COMUM / PRECARC / DPRDARC
      COMUM / EXTARC / EXTEND_ARC
      COMUM / DEXTARC / DEXTEND_ARC
      COMUM / DTERM / FRSTQUEUE, PrevNode, DEST_COUNT
      COMUM / NXTQUEUE / NXTQUEUE
      COMUM / SH_DIST / SH_DIST
      DIMENSÃO FOUT (maxNodes), NXTOUT (MAXARCS), P (maxNodes), LNGT
(MAXARCS)
      DIMENSÃO START (MAXARCS), END (MAXARCS), PRDARC (maxNodes)
      DIMENSÃO FIN (maxNodes), NXTIN (MAXARCS), DPRDARC (maxNodes)
      DIMENSÃO EXTEND_ARC (maxNodes), DEXTEND_ARC (maxNodes)
      DIMENSÃO NXTQUEUE (maxNodes), SH_DIST (maxNodes)
```

```
X ITER = 0
X = 0 NUM_ORIG
X = 0 NUM_EXT
X = 0 NUM_SUCC
       DEST = FRSTQUEUE
       R = TERMO
       DTERM = DEST
       GOTO 110

C ********************************************** ***
C
C ************ MAIN algoritmo Forward **********
C
C ********************************************** ***


C INÍCIO DE UM NOVO ITERATION

20 CONTINUAR

      PTERM = P (TERM)
      EXTARC = EXTEND_ARC (TERM)

      IF (EXTARC.LT.0) THEN
        EXTARC = DPRDARC (TERM)
        IF (EXTARC.LT.0) ir para 100
      END IF

C PATH ESPECULATIVO EXTENSÃO TENTATIVA

X = NUM_EXT NUM_EXT + 1
      ENDNODE = END (EXTARC)
      BSTLEVEL = LNGT (EXTARC) + P (ENDNODE)
      IF (PTERM.EQ.BSTLEVEL) THEN
X = NUM_SUCC NUM_SUCC + 1
         TERM = ENDNODE
         PRDARC (TERM) = EXTARC

C Se um destino novo é encontrado, alterne para a ALGORITHM TRÁS

         IF (NXTQUEUE (TERM) .GE.0) THEN
           IF (SH_DIST (TERM) .LT.0) THEN
             SH_DIST (TERM) = P (R) -P (TERM)
             GOTO 110
           END IF
         END IF

C voltar para outra ITERATION

         IR PARA 20
      END IF
C
C TENTATIVA extensão foi vencida, SO SCAN nodo terminal
C

100 CONTINUAR
X = ITER ITER + 1
       BSTLEVEL = GRANDE
       J = FOUT (TERM)
200 IF (J.GT.0) THEN
        NEW_LEVEL = P (FIM (J)) + LNGT (J)
```

```
         IF (NEW_LEVEL.LT.BSTLEVEL) THEN
           BSTLEVEL = NEW_LEVEL
           EXTARC = J
         END IF
         J = NXTOUT (J)
         GOTO 200
        END IF
        EXTEND_ARC (TERM) = EXTARC



       IF (BSTLEVEL.GT.PTERM) THEN

C CONTRAÇÃO PATH

           P (TERM) = BSTLEVEL
           IF (TERM.NE.R) THEN
             TERM = START (PRDARC (TERM))

C voltar para outra ITERATION mas ignorar o TENTATIVA DE EXTENSÃO
ESPECULATIVO

             PTERM = P (TERM)
             Ir para 100
           MAIS

C Origem obtido; CHAVE ao algoritmo TRÁS

X = NUM_ORIG NUM_ORIG + 1
             IF (PTERM.GE.LARGE) THEN
               PRINT *, 'NO caminho para o destino'
               PAUSA
               PARE
             END IF
             IR PARA 110
           END IF
       MAIS

C EXTENSÃO PATH

          TERM = END (EXTARC)
          PRDARC (TERM) = EXTARC

C Se um destino novo é encontrado, alterne para a ALGORITHM TRÁS

          IF (NXTQUEUE (TERM) .GE.0) THEN
            IF (SH_DIST (TERM) .LT.0) THEN
              SH_DIST (TERM) = P (R) -P (TERM)
              GOTO 110
            END IF
          END IF


C voltar para outra ITERATION

          IR PARA 20
        END IF


C ********************************************** ***
C
```

```
C ************ ALGORITHM TRÁS MAIN **********
C
C ********************************************** ***

C LIMPEZA DA FILA

110 CONTINUAR

115 IF (SH_DIST (DEST) .GE.0) THEN
          DTERM = NXTQUEUE (DEST)
          NXTQUEUE (DEST) = - 1
          DEST = DTERM
          NXTQUEUE (PrevNode) = DEST
          DEST_COUNT = DEST_COUNT-1

RETURN C SE OS DESTINOS foram esgotados

          IF (DEST_COUNT.EQ.0) THEN
            RETURN
          MAIS
            GOTO 115
          END IF
        END IF


C INÍCIO DE UM NOVO ITERATION

120 CONTINUAR

       DPTERM = P (DTERM)
       DEXTARC = DEXTEND_ARC (DTERM)

       IF (DEXTARC.LT.0) THEN
         DEXTARC = PRDARC (DTERM)
         IF (DEXTARC.LT.0) IR PARA 1100
       END IF

C EXTENSÃO PATH

X = NUM_EXT NUM_EXT + 1
       StartNode = START (DEXTARC)
       DBSTLEVEL = P (StartNode) -LNGT (DEXTARC)
       IF (DPTERM.EQ.DBSTLEVEL) THEN
X = NUM_SUCC NUM_SUCC + 1
          DTERM = StartNode
          DPRDARC (DTERM) = DEXTARC

C se a origem for encontrado, alterne para a ALGORITHM FRENTE

          IF (DTERM.EQ.R) THEN
            SH_DIST (DEST) = P (R) -P (DEST)
            DEST_COUNT = DEST_COUNT-1

RETURN C SE OS DESTINOS foram esgotados

            IF (DEST_COUNT.EQ.0) THEN
              RETURN
            MAIS
              DTERM = NXTQUEUE (DEST)
              NXTQUEUE (DEST) = - 1
              DEST = DTERM
```

```
              NXTQUEUE (PrevNode) = DEST
              GOTO 20
            END IF
          END IF

C voltar para outra ITERATION

          IR PARA 120
       END IF
C
C SCAN TERMINAL NÓ
C
1100 CONTINUAR
X = ITER ITER + 1
       DBSTLEVEL = -Grande
       J = FIN (DTERM)
1200 IF (J.GT.0) THEN
        StartNode = START (J)
        NEW_LEVEL = P (startNode) - LNGT (J)
        IF (NEW_LEVEL.GT.DBSTLEVEL) THEN
          DBSTLEVEL = NEW_LEVEL
          DEXTARC = J
        END IF
        J = NXTIN (J)
        GOTO 1200
       END IF
       DEXTEND_ARC (DTERM) = DEXTARC


       IF (DBSTLEVEL.LT.DPTERM) THEN

C CONTRAÇÃO PATH

          P (DTERM) = DBSTLEVEL
          IF (DTERM.NE.DEST) THEN
            DTERM = END (DPRDARC (DTERM))

C voltar para outra SCAN

            DPTERM = P (DTERM)
            IR PARA 1100
          MAIS

C DESTINO atingido; CHAVE PARA A ORIGEM

X = NUM_ORIG NUM_ORIG + 1
            IF (DPTERM.LE.-GRANDE) THEN
              PRINT *, 'NO CAMINHO PARA A ORIGEM'
              PAUSA
              PARE
            END IF
            PrevNode = DEST
            DEST = NXTQUEUE (DE ST)
            DTERM = DEST
            IR PARA 20
          END IF
       MAIS

C EXTENSÃO PATH

         DTERM = START (DEXTARC)
```

```
          DPRDARC (DTERM) = DEXTARC

C se a origem for encontrado, alterne para a ALGORITHM FRENTE

          IF (DTERM.EQ.R) THEN
            SH_DIST (DEST) = P (R) -P (DEST)
            DEST_COUNT = DEST_COUNT-1
            IF (DEST_COUNT.EQ.0) THEN
              RETURN
            MAIS
              DTERM = NXTQUEUE (DEST)
              NXTQUEUE (DEST) = - 1
              DEST = DTERM
              NXTQUEUE (PrevNode) = DEST
              GOTO 20
            END IF
          END IF

C voltar para outra ITERATION

          IR PARA 120
         END IF



        FIM



**************************************************
************************

Código de Relaxamento

**************************************************
************************

Este código (RELAX_QC) implementa o método de relaxamento de
Seção 3.3. O código foi escrito pelo autor em colaboração com Paul
Tseng (com contribuições de Jon Eckstein na parte de inicialização do
código). Existem algumas diferenças entre o código dadas aqui e
semelhante
códigos da família RELAX que foram lançados para a comunidade de
pesquisa
desde 1987 ([BeT88], [BeT90]).

AMOSTRA C ******* PROGRAMA CHAMADA PARA A RELAXT_QC *******
C
C Este programa vai LER UM PROBLEMA EM UM ARQUIVO
C forma padrão, e resolvê-lo USANDO RELAXT_QC
C
C relaxt-QC é diferente do código relaxt-III E OUTROS MAIS CEDO
CÓDIGOS C relaxamento em que o mantém
C o conjunto de nós com déficit diferente de zero EM UMA FILA FIFO.
C TI tem também a opção da inicialização USANDO UM
C CRASH PROCESSO; Esta inicialização É RECOMENDADO
C para os problemas que ser difícil usar o padrão
C INICIAÇÃO.
C
C ********************************************** ***
```

```
C
      Programa principal
      Implícita INTEGER (AZ)
      Real * 8 TCOST, TT, TIMER
      INTEGER C (20,000), X (20000), L (20000), RC (20000), B (8000)
      INTEGER STARTN (20000), ENDN (20000)
      INTEGER I1 (8000), I2 (8000), I3 (8000), I4 (20000), I5 (8000)
      INTEGER I6 (20000), I7 (20000)
      INTEGER TFSTOU (8000), TNXTOU (20000), TFSTIN (8000), TNXTIN
(20000)
      INTEGER NXTQUEUE (8000)
      LÓGICO * 1 L1 (8000), L2 (8000)
      COMUNS / matrizes / STARTN / ARRAYE / ENDN / ARRAYU / U / ARRAYX
/ X
     $ / ARRAY9 / RC / arrayb / B / BLK1 / I1 / BLK2 / I2 / BLK3 / I3
/ BLK4 / I4 / BLK5 / I5 / BLK6
     $ / I6 / BLK7 / I7 / BLK8 / L1 / BLK9 / L2
     $ / L / N, NA, GRANDE, NMULTINODE, ITER, NUM_AUGM, NUM_ASCNT,
NUM_SP, CRASH
      COMUM / BLK10 / TFSTOU / BLK11 / TNXTOU / BLK12 / TFSTIN / BLK13
/ TNXTIN
      COMUM / BLK14 / NXTQUEUE


C SET grande para um inteiro para a sua máquina


      GRANDE = 500000000
      DANGER_THRESH = GRANDE / 10


      PRINT *, 'método de relaxamento PARA MIN CUSTO DE FLUXO'
      PRINT *, '**********************************'
      PRINT *, 'READING problema de dados'
      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' velho ')
      Rewind (13)

C LEIA número de nós e arcos

      LER (13,1010) N, NA

C LEIA início, fim, custo e capacidade de cada ARC

      FAZER 20 I = 1, NA
        READ (13,1020) STARTN (I), ENDN (I), C (I), L (I)
20 CONTINUAR

      FLAG1 = 0
      Flag2 = 0
      Flag3 = 0
      FAZER 25 I = 1, NA
        IF (ABS (C (I)). GT.LARGE) = 1 FLAG1
        IF (L (I) .GT.LARGE) = 1 flag2
        IF (ABS (C (I)). GT.DANGER_THRESH) = 1 Flag3
25 CONTINUAR

      IF (FLAG1.EQ.1) THEN
         PRINT *, "alguns custos excedam do intervalo permitido '
         PRINT * PROGRAMA, "não pode ser executado; PRESS <CR> PARA
SAIR '
         PAUSA
```

```
         PARE
      END IF

      IF (FLAG2.EQ.1) THEN
         PRINT *, 'algumas capacidades ARC ULTRAPASSAR A faixa
permitida "
         PRINT * PROGRAMA, "não pode ser executado; PRESS <CR> PARA
SAIR '
         PAUSA
         PARE
      END IF

      IF (FLAG3.EQ.1) THEN
         PRINT *, '*******************************'
         PRINT *, "alguns custos são perigosamente grande '
         PRINT *, "programa pode não funcionar corretamente '
         PRINT *, '*******************************'
      END IF

C LEIA SUPPLY de cada nó; Convertê-lo para EXIGIR

      FAZER 30 I = 1, N
        READ (13,1000) B (I)
        B (I) = - B (I)
30 CONTINUAR

      ENDFILE (13)
      Rewind (13)

      PRINT *, 'FIM DE LEITURA'

1000 FORMATO (1I8)
1010 FORMATO (2I8)
1020 FORMATO (4I8)


      PRINT *, "reestruturar os dados '
      CHAMADA INIDAT

C SET redução de custos iguais aos custos (preços iniciais são 0)

      DO 40 I = 1, NA
       RC (I) = C (I)
40 CONTINUAR

C ************* parâmetro de inicialização ************

C SET CRASH igual a 1 para ativar um PROCEDIMENTO ESPECIAL PARA CRASH
C o par de preço inicial-FLOW. Isto é recomendado para DIFÍCIL
C problemas onde a inicialização padrão de zero PREÇOS
C resulta em tempos de execução longa.

      CRASH = 1

      PRINT *, '****************************************** ** '
      PRINT *, "RESOLUÇÃO DO PROBLEMA '

TEMPO C LEIA MAC II

      TIMER = Long (362) /60.0
```

```
C CHAMADA MAIN rotina de relaxamento

      CHAMADA RELAXT_QC

TEMPO C LEIA MAC II

      TT = longa (362) /60.0 - TIMER
      PRINT *, 'TIME TOTAL =', TT, 'SECS.'

C Verifica a exactidão da ANSWER

      FAZER 80 NODE = 1, N
        IF (B (Nó) .NE.0) THEN
          PRINT *, "excedente diferente de zero AT NODE ', NODE
        ENDIF
80 CONTINUAR
      FAZER 90 ARC = 1, NA
        IF (X (ARC) .GT.0) THEN
          IF (RC (ARC) .GT.0) THEN
            PRINT *, 'CS VIOLADOS AT ARC ", ARC
          ENDIF
        ENDIF
        IF (U (ARC) .GT.0) THEN
          IF (RC (ARC) .LT.0) THEN
            PRINT *, 'CS VIOLADOS AT ARC ", ARC
          ENDIF
        ENDIF
90 CONTINUAR

      TCOST = DBLE (float (0))
      FAZER 100 I = 1, NA
        TCOST = TCOST + DBLE (FLOAT (X (I) C * (I)))
100 CONTINUAR

      ESCREVA (9,1100) TCOST
1100 FORMATO ('', 'custo ótimo =', F14.2)

      PRINT *, 'NÚMERO DE SH inicial. Espólios PATH = ', NUM_SP
      PRINT *, 'número de iterações =', ITER
      PRINT *, 'NÚMERO DE Multinode ITERATIONS =', NMULTINODE
      PRINT *, 'número de passos Multinode ASCENT =', NUM_ASCNT
      PRINT *, "número de aumentos = ', NUM_AUGM
      PRINT *, '****************************************** ** '
      PRINT *, 'programa terminou; PRESS <CR> PARA SAIR '

      PAUSA
      FIM




C ********************************************** ***

      SUBROUTINE INIDAT

C Esta sub-rotina usa o STARTN matrizes de dados e ENDN
C para a construção de matrizes de dados auxiliares FOU, NXTOU, FIN, e
C NXTIN que são obrigados por relaxt. Neste nós sub-rotina
C arbitrariamente encomendar os arcos deixando cada nó e loja
```

```
C Este informações em FOU e NXTOU. Da mesma forma, nós arbitragem
C rilly encomendar os arcos inserir cada nó e armazenar esta
C informações em FIN e NXTIN. Após a conclusão da
C construção, temos que

C FOU (I) = Primeiro arco deixando nó I.
C NXTOU (J) = Próximo arco deixando o nó principal do arco J.
C FIN (I) = Primeiro arco nó entrar I.
C NXTIN (J) = Próximo arco entrar no nó cauda do arco J.

      COMUM / matrizes / STARTN / ARRAYE / ENDN / BLK1 / TEMPIn / BLK2
/ TEMPOU
     $ / BLK3 / FOU / BLK4 / NXTOU / BLK5 / FIN / BLK6 / NXTIN
     $ / L / N, NA
      INTEGER STARTN (1), ENDN (1), TEMPIn (1), TEMPOU (1), fou (1)
      INTEGER NXTOU (1), FIN (1), NXTIN (1)
      * 1 L LÓGICO

      FAZER 25 I = 1, N
        FIN (I) = 0
        Fou (I) = 0
        TEMPIn (I) = 0
25 TEMPOU (I) = 0

      FAZER 27 I = 1, NA
        NXTIN (I) = 0
        NXTOU (I) = 0
        I1 = STARTN (I)
        I2 = ENDN (I)
        IF (FOU (I1) .NE.0) THEN
          NXTOU (TEMPOU (I1)) = I
        MAIS
          FOU (I1) = I
        END IF
        TEMPOU (I1) = I
        IF (FIN (I2) .NE.0) THEN
          NXTIN (TEMPIn (I2)) = I
        MAIS
          FIN (I2) = I
        END IF
27 TEMPIn (I2) = I
      RETURN
      FIM



C ***********************************************
**********************
C
C RELAXT_QC SUBROUTINE
C VERSÃO DO novembro 1991
C
C relaxt-QC é diferente do código relaxt-III E OUTROS MAIS CEDO
CÓDIGOS C relaxamento em que o mantém
C o conjunto de nós com déficit diferente de zero EM UMA FILA FIFO.
C TI tem também a opção da inicialização USANDO UM
C CRASH PROCESSO; Esta inicialização É RECOMENDADO
C PARA problemas difíceis.
C
C ***********************************************
**********************
```

```
C
C Esta sub-rotina resolve o (linear) custo mínimo comum
C problema de fluxo de rede.
C
C A rotina implementa o método de relaxamento de
C
C Bertsekas, DP, "A Unified quadro para os métodos primal-dual ..."
C Math. Programação, Vol. 32, 1985, pp. 125-145
C Bertsekas, DP, e Tseng, P., "Métodos de relaxamento para o custo
mínimo .."
C Research Operations J., Vol. 36, 1988, pp. 93-114
C
C O método de relaxamento é também descrita nos livros:
c
C Bertsekas, DP, "Rede Linear Optimization: Algoritmos e Códigos"
C MIT Press, 1991
C Bertsekas, DP, e Tsitsiklis, JN, "Paralelo e Distribuído
C Computação: Métodos Numéricos "Prentice-Hall, 1989
C
C A rotina foi escrita por Dimitri Bertsekas e Paul Tseng com
Contribuições C por Jonathan Eckstein na inicialização padrão
Rotina C.
C
C Uma diferença entre esta rotina eo código semelhante RELAX é
C que se mantém uma estrutura de dados que dá a todos os arcos
equilibradas
C na rede. Essa estrutura é chamada de "árvore" para histórico
Razões C, mesmo que ele descreve uma sub-rede que irá geralmente
C ser nem acíclico nem ligado. Além disso, a árvore pode conter alguns
Arcos em C que não estão em equilíbrio: o que acaba por ser mais
barato para purga
Arcos em C que se tornaram apenas desequilibrado quando seus nós
finais estão sendo
C digitalizada, ao contrário de manter sempre um conjunto exato de
arcos equilibradas.
C
C ********************************************** ***************
C
C Todos os dados devem estar em Integer * 4. Para executar em sistemas
de memória limitada
C o STARTN matrizes, ENDN, NXTIN, NXTOU, SAVE, FIN, FOU, LABEL,
C PRDCSR, NXTQUEUE podem ser declaradas como INTEGER * 2.
C
CN (o número de nós)
C NA (número de arcos)
C GRANDE (um grande número inteiro positivo para representar o
infinito.
C Todos os dados do problema deve ser inferior a LARGE em grandeza,
C e grandes deve ser menor do que, digamos, 1/4 o maior inteiro * 4
C da máquina utilizada. Isso irá proteger principalmente contra
C estouro em problemas incapacitado sempre que as capacidades de arco
C são levados finita, mas muito grande).
C STARTN (NA) (a matriz nó principal)
C ENDN (NA) (a matriz nó cauda)
C RC (NA) (a matriz de custo reduzido)
CX (NA) (a matriz de fluxo de arco)
CU (NA) (a matriz de capacidade de fluxo de arco)
C DFCT (N) (a matriz déficit)
C FOU (N) (o primeiro arco fora array)
C FIN (N) (o primeiro arco em array)
C NXTOU (NA) (o próximo arco fora array)
```

```
C NXTIN (NA) (o próximo arco em array)
C NXTQUEUE (N) (o próximo nó na fila para ser examinado)
C
C Esta sub-rotina coloca o fluxo ideal na matriz X
C e o custo reduzido do vetor correspondente na matriz RC.
C
C Se matrizes favoráveis X, U, e RC são conhecidos que podem
C ser passado para a rotina diretamente. Isto requer que o
C porção de inicialização da rotina (até a linha 70) é ignorado.
C
C ************************************************
******************

      RELAXT_QC SUBROUTINE
      Implícita INTEGER (AZ)
      LÓGICO * 1 FEASBL, parado, SCAN, SWITCH, MARK, POSIT, pChange
      Real * 8 TT, temporizador, TIMER1, CAPFACTOR
      COMUNS / matrizes / STARTN / ARRAYE / ENDN / ARRAYU / U / ARRAYX
/ X / ARRAY9 / RC
     $ / Arrayb / DFCT / BLK1 / LABEL / BLK2 / PRDCSR / BLK3 / FOU /
BLK4 / NXTOU / BLK5 / FIN
     $ / BLK6 / NXTIN / BLK7 / SAVE / BLK8 / SCAN / BLK9 / MARK
     $ / L / N, NA, GRANDE, NMULTINODE, ITER, NUM_AUGM, NUM_ASCNT,
NUM_SP, CRASH
      COMUM / BLK10 / TFSTOU / BLK11 / TNXTOU / BLK12 / TFSTIN / BLK13
/ TNXTIN
      COMUM / BLK14 / NXTQUEUE

C Cada bloco comum contém apenas um array, para que as matrizes em
RELAXT_QC
C pode ser dimensionado para um elemento e tomar a sua dimensão a
partir da
C rotina de chamada principal. Com este RELAXT_QC truque não precisa
ser recompilados
C se a dimensão do problema muda. Se o seu FORTRAN não suporta
C, esta característica alterar as dimensões abaixo de ser o mesmo que
aqueles
C declarou em seu programa de chamada principal.

      DIMENSÃO TFSTOU (1), TNXTOU (1), TFSTIN (1), TNXTIN (1)
      DIMENSÃO STARTN (1), ENDN (1), L (1), X (1), V (1), DFCT (1)
      LABEL DIMENSÃO (1), PRDCSR (1), SCAN (1), FOU (1), NXTOU (1)
      DIMENSÃO FIN (1), NXTIN (1), SAVE (1), Mark (1)
      DIMENSÃO NXTQUEUE (1)

C DDPOS e DDNEG são matrizes que dão as derivadas direcionais para
C variações de preços de nó simples todos positivos e negativos. Estes
são utilizados
C, apenas na fase inicial do algoritmo, antes de os dados de "árvore"
Estrutura de C entra em jogo. Portanto, eles são equivalenced para
C TFSTOU e TFSTIN, que são do mesmo tamanho (número de nós) e são
C apenas usado depois que a árvore entra em uso.

      DIMENSÃO DDPOS (1), DDNEG (1)
      CORRESPONDÊNCIA (DDPOS (1), TFSTOU (1)), (DDNEG (1), TFSTIN (1))

C reduzir a capacidade de arco, tanto quanto possível w / out mudando
o problema
C Se esta é uma sensibilidade executar via SENSTV rotina pular o
Inicialização C, definindo REPEAT para .TRUE. no programa de chamada.
```

```
A inicialização C (a partir daqui até à linha 70)
C pode ser ignorado se o programa de chamada
Lugares C em valores escolhidos pelo usuário comuns para os fluxos de
arco, arco residual
Capacidades C, déficits nodais e custos reduzidos. É obrigatório que
os fluxos de arco
C e os custos residuais satisfazer slackess complementar, em cada
arco,
C e que a matriz DFCT corresponder adequadamente para flui o arco
inicial


       NDEF1 = 0
       DO 40 NODE = 1, N
        NODE_DEF = DFCT (nó)
        IF (NODE_DEF.LT.0) THEN
          NDEF1 = + 1 NDEF1
          SAVE (NDEF1) = NÓ
        END IF

C Note que nós também configurar o DDPOS inicial e DDNEG para cada nó.
C (isso não é necessário em RELAX)

        DDPOS (nó) = NODE_DEF
        DDNEG (nó) = - NODE_DEF

        MaxCap = 0
        SCAPOU = 0
        ARC = FOU (nó)
41 IF (ARC.GT.0) THEN
          IF (SCAPOU.LE.LARGE-U (ARC)) THEN
            SCAPOU = SCAPOU + U (ARC)
          MAIS
            IR PARA 40
          END IF
          ARC = NXTOU (ARC)
          IR PARA 41
        END IF
        IF (SCAPOU.LE.LARGE-NODE_DEF) THEN
          CAPOUT = SCAPOU + NODE_DEF
        MAIS
          IR PARA 40
        END IF
        IF (CAPOUT.LT.0) THEN

C ** PROBLEMA é inviável - SAIR

       PRINT *, "Sair durante a inicialização"
       PRINT *, "fluxo exógeno em NODE ', NODE,' excede OUT CAPACIDADE
'
       CHAMADA PRINTFLOWS (nó)
       IR PARA 400
        END IF

        SCAPIN = 0
        ARC = FIN (nó)
43 IF (ARC.GT.0) THEN
          U (ARC) = MIN0 (U (ARC), CAPOUT)
          IF (MAXCAP.LT.U (ARC)) MaxCap = U (ARC)
          IF (SCAPIN.LE.LARGE-U (ARC)) THEN
            SCAPIN = SCAPIN + U (ARC)
```

```
          MAIS
            IR PARA 40
          END IF
          ARC = NXTIN (ARC)
          IR PARA 43
        END IF
        IF (SCAPIN.LE.LARGE + NODE_DEF) THEN
          Capin = SCAPIN-NODE_DEF
        MAIS
          IR PARA 40
        END IF
        IF (CAPIN.LT.0) THEN

C *** PROBLEMA é inviável - SAIR

       PRINT *, "Sair durante a inicialização"
       PRINT *, "exógenas fluir de NODE ', NODE,' excede na capacidade
de '
       CHAMADA PRINTFLOWS (nó)
       IR PARA 400
       END IF

        ARC = FOU (nó)
45 IF (ARC.GT.0) THEN
          U (ARC) = MIN0 (U (ARC), Capin)
          ARC = NXTOU (ARC)
          IR PARA 45
        END IF
40 CONTINUAR




C ******* inicializar os fluxos de arco e os déficits nodais ********
C *** note que U (ARC) é redefinida como a capacidade residual da ARC
***

C Agora, calcule as derivadas direcionais para cada coordenada
exatamente.
C, bem como o cálculo dos défices. L (ARC) é o resíduo
Capacidade C em ARC, e X (ARC) é o fluxo. Estes sempre adicionar até o
Capacidade total C.



      FAZER 48 ARC = 1, NA
         X (ARC) = 0
         IF (RC (ARC) .LE. 0) THEN
            T = U (ARC)
            T1 = STARTN (ARC)
            T2 = ENDN (ARC)
            DDPOS (T1) = DDPOS (T1) + T
            DDNEG (T2) = DDNEG (T2) + T
            IF (RC (ARC) .lt. 0) THEN
               X (ARC) = T
               U (ARC) = 0
               DFCT (T1) = DFCT (T1) + T
               DFCT (T2) = DFCT (T2) - t
               DDNEG (T1) = DDNEG (T1) - t
               DDPOS (T2) = DDPOS (T2) - t
            END IF
```

```
         END IF
48 CONTINUAR

INITIALIZATION C ver com ITERATIONS RELAXAMENTO nó único
C Uma estratégia adaptativa é utilizada: o número de nó única iteração
Ciclos C tentativas é uma função da densidade média da rede.

      IF (NA.GT.N * 10) THEN
        NumPasses = 2
      MAIS
        NumPasses = 3
      END IF

C Vamos agora fazer 2 ou 3 passes através de todos os nós. Este é o
inicial
Fase C: se uma única iteração nó não é possível, basta ir para
C o próximo nó.

      FAZER 390 passes = 1, NumPasses

      FAZER 300 NÓ = 1, N

       IF (DFCT (nó) .NE. 0) THEN

C aumento de preços ou queda de preço? (Nota: é impossível ter os
dois.)

        IF (DDPOS (nó) .LE. 0) THEN

Aumento C Price. Loop sobre pontos de interrupção no + Preço (nó)
direção.
C Em arcos de saída, a tensão vai subir e custo reduzido vai cair
C - por isso, da próxima pausa vem em menor custo reduzido positivo.
C Em arcos de entrada, a tensão vai cair e custo reduzido vai subir
C - por isso, da próxima pausa vem em menor custo reduzido negativo.

           DELPRC = GRANDE

           ARC = FOU (nó)
303 IF (ARC.GT.0) THEN
              TRC = RC (ARC)
              IF ((TRC.GT. 0) .E. (TRC.LT.DELPRC)) THEN
                 DELPRC = TRC
              END IF
              ARC = NXTOU (ARC)
              GOTO 303
           END IF

           ARC = FIN (nó)
304 IF (ARC.GT.0) THEN
              TRC = RC (ARC)
              IF ((TRC.LT.0) .E. (TRC.GT.-DELPRC)) THEN
                 DELPRC = -TRC
              END IF
              ARC = NXTIN (ARC)
              GOTO 304
           END IF

C Se não há pontos de interrupção para a esquerda e subida ainda é
possível, o problema
C é inviável.
```

```
            IF (DELPRC.GE.LARGE) THEN
               IF (DDPOS (nó) .EQ.0) GOTO 300
               GOTO 400
            ENDIF

C Temos um breakpoint real. Aumento de preço por essa quantidade.
C Primeiro verifique o efeito sobre todos os arcos de saída, o que
terá um
Aumento da tensão de C e gota custo reduzido.

350 NXTBRK = GRANDE
            ARC = FOU (nó)
305 IF (ARC.GT.0) THEN
               TRC = RC (ARC)
               IF (TRC .EQ. 0) THEN
                  T1 = ENDN (ARC)
                  T = U (ARC)
                  IF (T.GT.0) THEN
                     DFCT (nó) = DFCT (nó) + T
                     DFCT (T1) = DFCT (T1) - t
                     X (ARC) = T
                     U (ARC) = 0
                  MAIS
                     T = X (ARC)
                  END IF
                  DDNEG (nó) = DDNEG (nó) - T
                  DDPOS (T1) = DDPOS (T1) - t
               END IF

C Para todos os arcos de saída tensão aumenta, e reduzir a queda de
custos.

               TRC = TRC - DELPRC
               IF ((TRC.GT.0) .E. (TRC.LT.NXTBRK)) THEN
                  NXTBRK = TRC
               Else if (TRC.EQ.0) THEN

Arco C vai desde a inactiva equilibrada. Basta alterar a tensão
C de aumento derivados, e verificar se há mudança de status na outra
extremidade.

                  DDPOS (nó) = DDPOS (nó) + U (ARC)
                  DDNEG (ENDN (ARC)) = DDNEG (ENDN (ARC)) + U (ARC)
               END IF
               RC (ARC) = TRC
               ARC = NXTOU (ARC)
               GOTO 305
            END IF

C Tempo para verificar os seus arcos de entrada para o nó.
C Estes arcos terá uma diminuição da tensão e um reduzido aumento de
custos.

            ARC = FIN (nó)
306 IF (ARC.GT.0) THEN
               TRC = RC (ARC)
               IF (TRC.EQ.0) THEN
                  T1 = STARTN (ARC)
                  T = X (ARC)
                  IF (T.GT.0) THEN
```

```
                  DFCT (nó) = DFCT (nó) + T
                  DFCT (T1) = DFCT (T1) - t
                  U (ARC) = T
                  X (ARC) = 0
               MAIS
                  T = U (ARC)
               END IF
               DDPOS (T1) = DDPOS (T1) - t
               DDNEG (nó) = DDNEG (nó) - T
            END IF

C Nota o aumento de custo reduzido para cada arco.

            TRC = TRC + DELPRC
            IF ((TRC.LT.0) .E. (TRC.GT.-NXTBRK)) THEN
               NXTBRK = -TRC
            Else if (TRC.EQ.0) THEN

C Agora vá para o movimento de ativo para equilibrada.
C Em caso afirmativo, os derivados de diminuir a tensão aumente.

               DDNEG (STARTN (ARC)) = DDNEG (STARTN (ARC)) + X (ARC)
               DDPOS (nó) = DDPOS (nó) + X (ARC)
            END IF
            RC (ARC) = TRC
            ARC = NXTIN (ARC)
            GOTO 306
         END IF

C Nós agora são feitas com a iteração. Se a direção da corrente
C ainda é um (degenerada) em direcção a subida, empurrar para a
frente.

         IF ((DDPOS (nó) .LE.0) .E. (NXTBRK.LT.LARGE)) THEN
            DELPRC = NXTBRK
            GOTO 350
         END IF


C Agora vem o código para uma descida de preços do nó.
C Em arcos de saída, a tensão vai cair e reduzido custo vai aumentar
C - por isso, da próxima pausa vem em menor custo reduzido negativo.
C Em arcos de entrada, a tensão vai aumentar e custo reduzido vai cair
C - por isso, da próxima pausa vem em menor custo reduzido positivo.

      Else if (DDNEG (nó) .LE.0) THEN

         DELPRC = GRANDE

         ARC = FOU (nó)
307 IF (ARC.GT.0) THEN
            TRC = RC (ARC)
            IF ((TRC.LT.0) .E. (TRC.GT.-DELPRC)) THEN
               DELPRC = -TRC
            ENDIF
            ARC = NXTOU (ARC)
            GOTO 307
         ENDIF

         ARC = FIN (nó)
308 IF (ARC.GT.0) THEN
```

```
            TRC = RC (ARC)
            IF ((TRC.GT.0) .E. (TRC.LT.DELPRC)) THEN
               DELPRC = TRC
            END IF
            ARC = NXTIN (ARC)
            GOTO 308
         END IF

C Se não houver nenhum ponto de interrupção, o problema não é
exequível,
C, a menos que nós estamos fazendo uma etapa degenerada.

         IF (DELPRC.EQ.LARGE) THEN
            IF (DDNEG (nó) .EQ.0) GOTO 300
            GOTO 400
         END IF

C Agora vamos dar o passo para o próximo ponto de interrupção.
Começamos com o
C arcos de saída. Estes têm uma queda de tensão e custo reduzido
C ascensão. Portanto, as transições são possíveis de ser equilibrado
para
C inativo ou ativo para equilibrada.

360 NXTBRK = GRANDE
         ARC = FOU (nó)
309 IF (ARC.GT.0) THEN
            TRC = RC (ARC)
            IF (TRC.EQ.0) THEN
               T1 = ENDN (ARC)
               T = X (ARC)
               IF (T.GT.0) THEN
                  DFCT (nó) = DFCT (nó) - T
                  DFCT (T1) = DFCT (T1) + T
                  U (ARC) = T
                  X (ARC) = 0
               MAIS
                  T = U (ARC)
               END IF
               DDPOS (nó) = DDPOS (nó) - T
               DDNEG (T1) = DDNEG (T1) - t
            END IF

C Login ascensão custo reduzido para todos os arcos.

            TRC = TRC + DELPRC
            IF ((TRC.LT.0) .E. (TRC.GT.-NXTBRK)) THEN
               NXTBRK = -TRC
            Else if (TRC.EQ.0) THEN

C Ativa para equilibrada. Derivs diminuir a tensão subir.

               DDNEG (nó) = DDNEG (nó) + X (ARC)
               DDPOS (ENDN (ARC)) = DDPOS (ENDN (ARC)) + X (ARC)
            END IF
            RC (ARC) = TRC
            ARC = NXTOU (ARC)
            GOTO 309
         END IF

C Agora fazer os arcos de entrada. Estes têm um aumento de tensão e
```

```
C, por conseguinte, uma gota de custo reduzido. As transições são
possíveis
C de inativo para equilibrado e de ser equilibrado para ativo.

         ARC = FIN (nó)
310 IF (ARC.GT.0) THEN
            TRC = RC (ARC)
            IF (TRC.EQ.0) THEN
               T1 = STARTN (ARC)
               T = U (ARC)
               IF (T.GT.0) THEN
                  DFCT (nó) = DFCT (nó) - T
                  DFCT (T1) = DFCT (T1) + T
                  X (ARC) = T
                  U (ARC) = 0
               MAIS
                  T = X (ARC)
               END IF
               DDNEG (T1) = DDNEG (T1) - t
               DDPOS (nó) = DDPOS (nó) - T
            END IF
            TRC = TRC - DELPRC
            IF ((TRC.GT.0) .E. (TRC.LT.NXTBRK)) THEN
               NXTBRK = TRC
            Else if (TRC.EQ.0) THEN
               DDPOS (STARTN (ARC)) = DDPOS (STARTN (ARC)) + U (ARC)
               DDNEG (nó) = DDNEG (nó) + U (ARC)
            END IF
            RC (ARC) = TRC
            ARC = NXTIN (ARC)
            GOTO 310
         END IF

C OK. O movimento é feito. É neste sentido ainda um (degenerado)
C direção subida. Se assim for, continue.

         IF ((DDNEG (nó) .LE.0) .E. (NXTBRK.LT.LARGE)) THEN
            DELPRC = NXTBRK
            GOTO 360
         END IF

       END IF

      END IF

300 CONTINUAR

390 CONTINUAR

C STORE os nós que agora têm DEFICIT NEGATIVO

      NDEF = NDEF1

      FAZER 372 I = 1, N
        IF (DFCT (I) .LT.0) THEN
          NDEF = NDEF + 1
          SAVE (NDEF) = I
        END IF
372 CONTINUAR

C RETURN se tiver terminado
```

```
      IF (NDEF.EQ.NDEF1) THEN
         RETURN
      END IF

C Esta versão do relaxt USOS SELETIVAMENTE mais curto ITERATIONS PATH
C para inicialização. ARCS de pequena capacidade não forem tomadas
C EM CONTA no menor cálculos PATH, conforme especificado
C PELO CAPFACTOR PARÂMETROS E CAPTHRESH que são fixadas ABAIXO.

C se o acidente parâmetro é definido como 1, o
C PROCEDIMENTO Shortest Path inicialização é mais extensa
CRASH C = 1 é recomendada para problemas difíceis


      THRESH = 15
      IF (NDEF-NDEF1.LE.THRESH) THEN
        Limit1 = NDEF1 + 1
        Limite2 = NDEF
      MAIS
        Limit1 = 1
        Limite2 = NDEF1
      END IF

       IF (CRASH.EQ.1) THEN
          NumPasses = INT (n / (2 * (NDEF-NDEF1)))
          Limit1 = NDEF1 + 1
          Limite2 = NDEF
          IF (NUMPASSES.GT.3) NumPasses = 3
       MAIS
          THRESH1 = 5
          IF ((NDEF-NDEF1.GT.THRESH) .E. (NDEF1.GT.THRESH1)) Vá para
70
          NumPasses = 1
       END IF

        NUM_SP = 0
        CAPFACTOR = 0,02

        FAZER 375 J = 1, NumPasses
         IF (CRASH.EQ.1) THEN
           CAPTHRESH = INT (CAPFACTOR MaxCap * * (J-1))
         MAIS
           CAPTHRESH = INT (CAPFACTOR * MaxCap)
         END IF
         FAZER 374 I = limit1, Limite2
           NÓ = SAVE (I)
           IF (DFCT (nó) .LT.0) THEN
             CHAMADA SHORT_PATH (nó, CAPTHRESH)
             NUM_SP = + 1 NUM_SP
           END IF
374 CONTINUAR
375 CONTINUAR

C retificar a fluxos e DEFICITS para contabilizar
C ARCS pequena capacidade IGNORADO no menor cálculos PATH

      FAZER 376 ARC = 1, NA
         IF ((RC (ARC) .LT.0) .E. (U (ARC) .GT.0)) THEN
            T = U (ARC)
            T1 = STARTN (ARC)
```

```
            T2 = ENDN (ARC)
            X (ARC) = X (ARC) + T
            U (ARC) = 0
            DFCT (T1) = DFCT (T1) + T
            DFCT (T2) = DFCT (T2) - t
          END IF
          IF ((RC (ARC) .GT.0) .E. (X (ARC) .GT.0)) THEN
            T = X (ARC)
            T1 = STARTN (ARC)
            T2 = ENDN (ARC)
            U (ARC) = U (ARC) + T
            X (ARC) = 0
            DFCT (T1) = DFCT (T1) - t
            DFCT (T2) = DFCT (T2) + T
          END IF
376 CONTINUAR

C ****************** FIM DE INICIAÇÃO **************

70 CONTINUAR

******* C inicializar o ******* árvore

      FAZER 72 I = 1, N
        TFSTOU (I) = 0
        TFSTIN (I) = 0
72 CONTINUAR

      FAZER 74 I = 1, NA
        TNXTIN (I) = - 1
        TNXTOU (I) = - 1
        IF (RC (I) .EQ.0) THEN
          TNXTOU (I) = TFSTOU (STARTN (I))
          TFSTOU (STARTN (I)) = I
          TNXTIN (I) = TFSTIN (ENDN (I))
          TFSTIN (ENDN (I)) = I
        END IF
74 CONTINUAR


C *********** Inicializar outras variáveis ***********

      FEASBL = .TRUE.
      ITER = 0
      NMULTINODE = 0
      NDFCT = N
      NUM_AUGM = 0
      NUM_ASCNT = 0
      NUMNONZERO = 0
      CHAVE = .FALSE.
      FAZER 76 I = 1, N
        MARK (I) =. FALSE.
        SCAN (I) =. FALSE.
76 CONTINUAR
      NLABEL = 0


C relaxt utiliza uma estratégia adaptativa para decidir se
C continuar o processo de digitalização depois de uma alteração de
preço.
C O TP parâmetros de limites e TS que controlam
```

```
C desta estratégia são definidos nas próximas linhas.

      TP = 10
      TS = INT (n / 15)


C Esta versão do relaxt também usa uma FILA PARA PRESERVAR A SET DE
NÓS DE C com déficit diferente de zero ou excedentes. Nós são
apanhados
C a fila para relaxamento.

C FILA DE INICIAÇÃO

      FAZER 82 NODE = 1, N-1
        NXTQUEUE (nó) = NODE + 1
82 CONTINUAR
      NXTQUEUE (N) = 1
      NÓ = N
      PrevNode = N-1
      LASTQUEUE = N
      NDFCT = N
      NUMNONZERO = O

C ************ ALGORITHM RELAXAMENTO INÍCIO **************

100 CONTINUAR

C CODE para o avanço do FILA

          PrevNode = NÓ
          NÓ = NXTQUEUE (nó)
          DEFCIT = DFCT (nó)
          IF (NODE.EQ.LASTQUEUE) THEN
            NDFCT = NUMNONZERO
            NUMNONZERO = 0
            LASTQUEUE = PrevNode
          END IF

C COCE para excluir um nó DA FILA

         IF (DEFCIT.EQ.0) THEN
            NXTNODE = NXTQUEUE (nó)
            IF (NODE.EQ.NXTNODE) THEN
              RETURN
            MAIS
              NXTQUEUE (PrevNode) = NXTNODE
              NXTQUEUE (nó) = 0
              NÓ = NXTNODE
              Ir para 100
            END IF
          MAIS
            POSIT = (DEFCIT.GT.0)
            NUMNONZERO = + 1 NUMNONZERO
          END IF

C ***** tentar uma ITERATION único nó do nó ****

      ITER = ITER + 1
      IF (POSIT) THEN

C ************ CASO DE NÓ W / ******** DÉFICIT POSITIVOS
```

```
      PChange = .FALSE.
      INDEF = DEFCIT
      DELX = 0
      NB = 0

C Verifique saída (provavelmente) arcos equilibradas do nó.

      ARC = TFSTOU (nó)
500 IF (ARC.GT.0) THEN
         IF ((RC (ARC) .EQ.0) .E. (X (ARC) .GT.0)) THEN
            DELX = DELX + X (ARC)
            NB = NB + 1
            SAVE (NB) = ARC
         ENDIF
         ARC = TNXTOU (ARC)
         GOTO 500
      END IF

C Verifique arcos de entrada.

      ARC = TFSTIN (nó)
501 IF (ARC.GT.0) THEN
         IF ((RC (ARC) .EQ.0) .E. (U (ARC) .GT.0)) THEN
            DELX = DELX + U (ARC)
            NB = NB + 1
            SAVE (NB) = -ARC
         ENDIF
         ARC = TNXTIN (ARC)
         GOTO 501
      END IF

C ***** FIM DA INICIAL ******* NODE SCAN

18 CONTINUAR

C ********* SE NO PREÇO A mudança é possível, saia **********

      IF (DELX.GT.DEFCIT) THEN
        SAIR = (DEFCIT .lt. INDEF)
        IR PARA 16
      END IF

C relaxt PESQUISADOS ao longo da direção SUBIDA PARA O
C MELHOR PREÇO POR VERIFICAR A inclinação da COST DUAL
C em pontos de paragem sucessivas.

C QUE AGORA calcular a distância para o ponto de quebra PRÓXIMO

      DELPRC = GRANDE
      ARC = FOU (nó)
502 SE (ARC .gt. 0) THEN
         RDCOST = RC (ARC)
         IF ((RDCOST.LT.0) .E. (RDCOST.GT.-DELPRC)) THEN
            DELPRC = -RDCOST
         ENDIF
         ARC = NXTOU (ARC)
         GOTO 502
      END IF
      ARC = FIN (nó)
503 SE (ARC .gt. 0) THEN
```

```
         RDCOST = RC (ARC)
         IF ((RDCOST.GT.0) .E. (RDCOST.LT.DELPRC)) THEN
            DELPRC = RDCOST
         ENDIF
         ARC = NXTIN (ARC)
         GOTO 503
      END IF

C ******* cheque se o problema é inviável *******

      IF ((DELX.LT.DEFCIT) .E. (DELPRC.EQ.LARGE)) THEN

C ***** O custo dupla pode ser diminuída sem limites *****

        IR PARA 400
      END IF

C **** SKIP FLUXO ADJUSTEMT SE NÃO HOUVER FLUXO DE MODIFICAR ***

      IF (DELX.EQ.0) Vá para 14

C Ajuste o fluxo em arcos incidente equilibrada de nó a ser
C manter frouxidão complementar, após a mudança de preço

      FAZER 13 J = 1, NB
        ARC = SAVE (J)
        IF (ARC.GT.0) THEN
          NODE2 = ENDN (ARC)
          T1 = X (ARC)
          DFCT (NODE2) = DFCT (NODE2) + T1
          IF (NXTQUEUE (NODE2) .EQ.0) THEN
            NXTQUEUE (PrevNode) = NODE2
            NXTQUEUE (NODE2) = NÓ
            PrevNode = NODE2
          END IF
          U (ARC) = U (ARC) + T1
          X (ARC) = 0
        MAIS
          NARC = -ARC
          NODE2 = STARTN (NARC)
          T1 = U (NARC)
          DFCT (NODE2) = DFCT (NODE2) + T1
          IF (NXTQUEUE (NODE2) .EQ.0) THEN
            NXTQUEUE (PrevNode) = NODE2
            NXTQUEUE (NODE2) = NÓ
            PrevNode = NODE2
          END IF
          X (NARC) = X (NARC) + T1
          U (NARC) = 0
        END IF
13 CONTINUAR
      DEFCIT = DEFCIT-DELX
14 IF (DELPRC.EQ.LARGE) THEN
        SAIR = .TRUE.
        IR PARA 19
      END IF

C nó corresponde a uma direcção de subida duplo. Diminuir
C o preço do nó DELPRC e calcular o tamanho do passo para o
C próximo ponto de interrupção no custo dupla
```

```
      NB = 0
      PChange = .TRUE.
      DP = DELPRC
      DELPRC = GRANDE
      DELX = 0
      ARC = FOU (nó)
504 IF (ARC.GT.0) THEN
        RDCOST = RC (ARC) + DP
        RC (ARC) = RDCOST
        IF (RDCOST.EQ.0) THEN
          NB = NB + 1
          SAVE (NB) = ARC
          DELX = DELX + X (ARC)
        END IF
        IF ((RDCOST.LT.0) .E. (RDCOST.GT.-DELPRC)) DELPRC = -RDCOST
        ARC = NXTOU (ARC)
        GOTO 504
      END IF
      ARC = FIN (nó)
505 IF (ARC.GT.0) THEN
        RDCOST = RC (ARC) -DP
        RC (ARC) = RDCOST
        IF (RDCOST.EQ.0) THEN
          NB = NB + 1
          SAVE (NB) = - ARC
          DELX = DELX + U (ARC)
        END IF
        IF ((RDCOST.GT.0) .E. (RDCOST.LT.DELPRC)) DELPRC = RDCOST
        ARC = NXTIN (ARC)
        GOTO 505
      END IF

C ***** retorno para verificar se uma outra alteração de preço é
possível ******

      IR PARA 18

******* C realizar aumento do fluxo no nó ****

16 DO 11 J = 1, NB
        ARC = SAVE (J)
        IF (ARC.GT.0) THEN

C *** ARC é um arco de saída do nó ********************

          NODE2 = ENDN (ARC)
          T1 = DFCT (NODE2)
          IF (T1.LT.0) THEN

C ** Diminuir o défice total, diminuindo o fluxo de ARC **

            SAIR = .TRUE.
            T2 = X (ARC)
            DX = MIN0 (DEFCIT, -T1, T2)
            DEFCIT = DEFCIT-DX
            DFCT (NODE2) = T1 + DX
          IF (NXTQUEUE (NODE2) .EQ.0) THEN
            NXTQUEUE (PrevNode) = NODE2
            NXTQUEUE (NODE2) = NÓ
            PrevNode = NODE2
          END IF
```

```
            X (ARC) = T2-DX
            U (ARC) = U (ARC) + DX
            IF (DEFCIT.EQ.0) Vá para 19
          END IF
        MAIS

C *** -ARC é um arco de entrada para o nó ***

          NARC = -ARC
          NODE2 = STARTN (NARC)
          T1 = DFCT (NODE2)
          IF (T1.LT.0) THEN

C ** Diminuir o défice total, aumentando o fluxo de -ARC **

            SAIR = .TRUE.
            T2 = U (NARC)
            DX = MIN0 (DEFCIT, -T1, T2)
            DEFCIT = DEFCIT-DX
            DFCT (NODE2) = T1 + DX
          IF (NXTQUEUE (NODE2) .EQ.0) THEN
            NXTQUEUE (PrevNode) = NODE2
            NXTQUEUE (NODE2) = NÓ
            PrevNode = NODE2
          END IF
            X (NARC) = X (NARC) + DX
            L (NARC) = T2-DX
            IF (DEFCIT.EQ.0) Vá para 19
          END IF
        END IF
11 CONTINUAR
19 DFCT (nó) = DEFCIT

C Reconstruir a lista de arcos equilibradas adjacentes a este nó.
C Para o vizinho nós, nós
C adicionar todos os arcos recém equilibrado, mas não se preocupe se
livrar
C das já anteriormente equilibradas (elas serão excluídas da próxima
vez que o
C nó adjacente é digitalizada).

       IF (pChange) THEN
          ARC = TFSTOU (nó)
          TFSTOU (nó) = 0
506 SE (ARC .gt. 0) THEN
             NXTARC = TNXTOU (ARC)
             TNXTOU (ARC) = -1
             ARC = NXTARC
             GOTO 506
          END IF
          ARC = TFSTIN (nó)
          TFSTIN (nó) = 0
507 SE (ARC .gt. 0) THEN
             NXTARC = TNXTIN (ARC)
             TNXTIN (ARC) = -1
             ARC = NXTARC
             GOTO 507
          END IF
```

```
C Agora adicione os arcos atualmente equilibradas para a lista para
este nó
C (que agora está vazio), e as adjacentes apropriadas.

          FAZER 508 J = 1, NB
             ARC = SAVE (J)
             IF (ARC.LE.0) ARC = -ARC
             IF (TNXTOU (ARC) .lt. 0) THEN
                TNXTOU (ARC) = TFSTOU (STARTN (ARC))
                TFSTOU (STARTN (ARC)) = ARC
             END IF
             IF (TNXTIN (ARC) .lt. 0) THEN
                TNXTIN (ARC) = TFSTIN (ENDN (ARC))
                TFSTIN (ENDN (ARC)) = ARC
             END IF
508 CONTINUAR

       END IF

C *** final de única iteração nó para um nó de déficit positivo ***

      MAIS

C ******* única iteração nó para um nó défice negativo ******

      PChange = .FALSE.
      DEFCIT = -DEFCIT
      INDEF = DEFCIT
      DELX = 0
      NB = 0

      ARC = TFSTIN (nó)
509 SE (ARC .gt. 0) THEN
         IF ((RC (ARC) .EQ. 0) .E. (X (ARC) .gt. 0)) THEN
            DELX = DELX + X (ARC)
            NB = NB + 1
            SAVE (NB) = ARC
         ENDIF
         ARC = TNXTIN (ARC)
         GOTO 509
      END IF
      ARC = TFSTOU (nó)
510 SE (ARC .gt. 0) THEN
         IF ((RC (ARC) .EQ. 0) .E. (U (ARC) .gt. 0)) THEN
            DELX = DELX + U (ARC)
            NB = NB + 1
            SAVE (NB) = -ARC
         ENDIF
         ARC = TNXTOU (ARC)
         GOTO 510
      END IF

28 CONTINUAR
      IF (DELX.GE.DEFCIT) THEN
        SAIR = (DEFCIT .lt. INDEF)
        IR PARA 26
      END IF

C Compute distância para o próximo ponto de interrupção.

      DELPRC = GRANDE
```

```
      ARC = FIN (nó)
511 SE (ARC .gt. 0) THEN
         RDCOST = RC (ARC)
         IF ((RDCOST .lt. 0) .E. (RDCOST.GT.-DELPRC)) THEN
            DELPRC = -RDCOST
         ENDIF
         ARC = NXTIN (ARC)
         GOTO 511
      END IF
      ARC = FOU (nó)
512 SE (ARC .gt. 0) THEN
         RDCOST = RC (ARC)
         IF ((RDCOST.GT.0) .E. (RDCOST.LT.DELPRC)) THEN
            DELPRC = RDCOST
         ENDIF
         ARC = NXTOU (ARC)
         GOTO 512
      END IF

C ******* cheque se o problema é inviável ************************

      IF ((DELX.LT.DEFCIT) .E. (DELPRC.EQ.LARGE)) THEN
        IR PARA 400
      END IF
      IF (DELX.EQ.0) Vá para 24

Aumento do fluxo C ******* é possível ************************

      FAZER 23 J = 1, NB
        ARC = SAVE (J)
        IF (ARC.GT.0) THEN
          NODE2 = STARTN (ARC)
          T1 = X (ARC)
          DFCT (NODE2) = DFCT (NODE2) -T1
          IF (NXTQUEUE (NODE2) .EQ.0) THEN
            NXTQUEUE (PrevNode) = NODE2
            NXTQUEUE (NODE2) = NÓ
            PrevNode = NODE2
          END IF
          U (ARC) = U (ARC) + T1
          X (ARC) = 0
        MAIS
          NARC = -ARC
          NODE2 = ENDN (NARC)
          T1 = U (NARC)
          DFCT (NODE2) = DFCT (NODE2) -T1
          IF (NXTQUEUE (NODE2) .EQ.0) THEN
            NXTQUEUE (PrevNode) = NODE2
            NXTQUEUE (NODE2) = NÓ
            PrevNode = NODE2
          END IF
          X (NARC) = X (NARC) + T1
          U (NARC) = 0
        END IF
23 CONTINUAR
      DEFCIT = DEFCIT-DELX
24 IF (DELPRC.EQ.LARGE) THEN
        SAIR = .TRUE.
        IR PARA 29
      END IF
```

```
C ******* aumento de preços no nó é possível ********

      NB = 0
      PChange = .TRUE.
      DP = DELPRC
      DELPRC = GRANDE
      DELX = 0
      ARC = FIN (nó)
513 IF (ARC.GT.0) THEN
        RDCOST = RC (ARC) + DP
        RC (ARC) = RDCOST
        IF (RDCOST.EQ.0) THEN
          NB = NB + 1
          SAVE (NB) = ARC
          DELX = DELX + X (ARC)
        END IF
        IF ((RDCOST.LT.0) .E. (RDCOST.GT.-DELPRC)) DELPRC = -RDCOST
        ARC = NXTIN (ARC)
        GOTO 513
      END IF
      ARC = FOU (nó)
514 IF (ARC.GT.0) THEN
        RDCOST = RC (ARC) -DP
        RC (ARC) = RDCOST
        IF (RDCOST.EQ.0) THEN
          NB = NB + 1
          SAVE (NB) = - ARC
          DELX = DELX + U (ARC)
        END IF
        IF ((RDCOST.GT.0) .E. (RDCOST.LT.DELPRC)) DELPRC = RDCOST
        ARC = NXTOU (ARC)
        GOTO 514
      END IF
      IR PARA 28

******* C realizar aumento do fluxo no nó ****

26 DO 21 J = 1, NB
        ARC = SAVE (J)
        IF (ARC.GT.0) THEN

C *** ARC é um arco de entrada para o nó ********************

          NODE2 = STARTN (ARC)
          T1 = DFCT (NODE2)
          IF (T1.GT.0) THEN
            SAIR = .TRUE.
            T2 = X (ARC)
            DX = MIN0 (DEFCIT, T1, T2)
            DEFCIT = DEFCIT-DX
            DFCT (NODE2) = T1-DX
          IF (NXTQUEUE (NODE2) .EQ.0) THEN
            NXTQUEUE (PrevNode) = NODE2
            NXTQUEUE (NODE2) = NÓ
            PrevNode = NODE2
          END IF
            X (ARC) = T2-DX
            U (ARC) = U (ARC) + DX
            IF (DEFCIT.EQ.0) Vá para 29
          END IF
        MAIS
```

```
C *** -ARC é um arco de saída do nó ******

          NARC = -ARC
          NODE2 = ENDN (NARC)
          T1 = DFCT (NODE2)
          IF (T1.GT.0) THEN
            SAIR = .TRUE.
            T2 = U (NARC)
            DX = MIN0 (DEFCIT, T1, T2)
            DEFCIT = DEFCIT-DX
            DFCT (NODE2) = T1-DX
          IF (NXTQUEUE (NODE2) .EQ.0) THEN
            NXTQUEUE (PrevNode) = NODE2
            NXTQUEUE (NODE2) = NÓ
            PrevNode = NODE2
          END IF
            X (NARC) = X (NARC) + DX
            L (NARC) = T2-DX
            IF (DEFCIT.EQ.0) Vá para 29
          END IF
        END IF
21 CONTINUAR
29 DFCT (nó) = - DEFCIT

C Reconstruir a lista de arcos equilibradas adjacentes a este nó.

       IF (pChange) THEN

          ARC = TFSTOU (nó)
          TFSTOU (nó) = 0
515 SE (ARC .gt. 0) THEN
             NXTARC = TNXTOU (ARC)
             TNXTOU (ARC) = -1
             ARC = NXTARC
             GOTO 515
          END IF
          ARC = TFSTIN (nó)
          TFSTIN (nó) = 0
516 SE (ARC .gt. 0) THEN
             NXTARC = TNXTIN (ARC)
             TNXTIN (ARC) = -1
             ARC = NXTARC
             GOTO 516
          END IF

C *** Agora adicione os arcos atualmente equilibradas para a lista
para este nó ***
C *** (que agora está vazio), e as adjacentes apropriadas. ***

          FAZER 517 J = 1, NB
             ARC = SAVE (J)
             IF (ARC.LE.0) ARC = -ARC
             IF (TNXTOU (ARC) .lt. 0) THEN
                TNXTOU (ARC) = TFSTOU (STARTN (ARC))
                TFSTOU (STARTN (ARC)) = ARC
             END IF
             IF (TNXTIN (ARC) .lt. 0) THEN
                TNXTIN (ARC) = TFSTIN (ENDN (ARC))
                TFSTIN (ENDN (ARC)) = ARC
             END IF
```

```
517 CONTINUAR

       END IF

C ***** final de única iteração nó para um nó défice negativo ***

       END IF

       IF (SAIR) ir para 100

C ******* FAZER A ITERATION Multinode do nó *****************

       NMULTINODE = + 1 NMULTINODE
       CHAVE = (NDFCT.LT.TP)

C ******* UNMARK NODES LABELED ******* EARLIER

          FAZER 90 J = 1, NLABEL
            NODE2 = LABEL (J)
            MARK (NODE2) =. FALSE.
            SCAN (NODE2) =. FALSE.
90 CONTINUAR

C ******* INITIALIZE LABELING ******

          NLABEL = 1
          LABEL (1) = NÓ
          MARK (nó) =. TRUE.
          PRDCSR (nó) = 0

C ******** SCAN COMEÇAR NODE *********

          SCAN (nó) =. TRUE.
          NSCAN = 1
          DM = DFCT (nó)
          DELX = 0
          FAZER 95 J = 1, NB
           ARC = SAVE (J)
           IF (ARC.GT.0) THEN
            IF (POSIT) THEN
              NODE2 = ENDN (ARC)
            MAIS
              NODE2 = STARTN (ARC)
            END IF
            IF (.NOT.MARK (NODE2)) THEN
               NLABEL = + 1 NLABEL
               LABEL (NLABEL) = NODE2
               PRDCSR (NODE2) = ARC
               MARK (NODE2) =. TRUE.
               DELX = DELX + X (ARC)
            END IF
           MAIS
            NARC = -ARC
            IF (POSIT) THEN
              NODE2 = STARTN (NARC)
            MAIS
              NODE2 = ENDN (NARC)
            END IF
            IF (.NOT.MARK (NODE2)) THEN
              NLABEL = + 1 NLABEL
              LABEL (NLABEL) = NODE2
```

```
              PRDCSR (NODE2) = ARC
              MARK (NODE2) =. TRUE.
              DELX = DELX + U (NARC)
            END IF
          END IF
95 CONTINUAR

C **** iniciar a digitalização NODES rotulado ****

120 NSCAN = + 1 NSCAN

C ****** verificar para ver se a chave precisa ser definida ******
C parâmetro indica que pode agora ser melhor para continuar a
digitalização
C até que todos os nodos marcados são digitalizados

         CHAVE = CHAVE .OR ((NSCAN .gt. TS) .E. (NDFCT.LT.TS)).

C varredura **** próximo nó na lista de nós rotulados ****
C *** varredura continuará até uma superestimativa do residual
Capacidade C em toda a corte correspondente ao conjunto digitalizado
de nós (chamados
C DELX) excede o valor absoluto do défice total do digitalizada
Nós C (chamado DM), ou então um caminho de aumento foi encontrado. Que
são arcos
C na árvore, mas não são equilibradas são limpas como parte da
digitalização
C processo.

      I = LABEL (NSCAN)
      SCAN (I) =. TRUE.
      IF (POSIT) THEN

C nó digitalização ******* I para o caso de déficit positivo ******

      NAUGNOD = 0
      PRVARC = 0
      ARC = TFSTOU (I)

518 IF (ARC.GT.0) THEN

C ***** ARC é um arco de saída do nó *****

        IF (RC (ARC) .EQ. 0) THEN
           IF (X (ARC) .gt. 0) THEN
              NODE2 = ENDN (ARC)
              IF (.NOT. MARK (NODE2)) THEN

C NODE2 não está no conjunto rotulado. Adicionar NODE2 ao conjunto
rotulado.

                 PRDCSR (NODE2) = ARC
                 IF (DFCT (NODE2) .LT.0) THEN
                    NAUGNOD = + 1 NAUGNOD
                    SAVE (NAUGNOD) = NODE2
                 END IF
                 NLABEL = + 1 NLABEL
                 LABEL (NLABEL) = NODE2
                 MARK (NODE2) =. TRUE.
                 DELX = DELX + X (ARC)
              END IF
```

```
            END IF
            PRVARC = ARC
            ARC = TNXTOU (ARC)
         MAIS
            TMPARC = ARC
            ARC = TNXTOU (ARC)
            TNXTOU (TMPARC) = -1
            IF (PRVARC .EQ. 0) THEN
               TFSTOU (I) = ARC
            MAIS
               TNXTOU (PRVARC) = ARC
            END IF
         END IF
         GOTO 518
      END IF

      PRVARC = 0
      ARC = TFSTIN (I)
519 IF (ARC.GT.0) THEN

C ***** ARC é um arco de entrada em NODE *****

         IF (RC (ARC) .EQ. 0) THEN
            IF (U (ARC) .gt. 0) THEN
               NODE2 = STARTN (ARC)
               IF (.NOT. MARK (NODE2)) THEN

C * NODE2 não está no conjunto rotulado. Adicionar NODE2 ao conjunto
rotulado. *

                  PRDCSR (NODE2) = - ARC
                  IF (DFCT (NODE2) .LT.0) THEN
                     NAUGNOD = + 1 NAUGNOD
                     SAVE (NAUGNOD) = NODE2
                  END IF
                  NLABEL = + 1 NLABEL
                  LABEL (NLABEL) = NODE2
                  MARK (NODE2) =. TRUE.
                  DELX = DELX + U (ARC)
               END IF
            END IF
            PRVARC = ARC
            ARC = TNXTIN (ARC)
         MAIS
            TMPARC = ARC
            ARC = TNXTIN (ARC)
            TNXTIN (TMPARC) = -1
            IF (PRVARC .EQ. 0) THEN
               TFSTIN (I) = ARC
            MAIS
               TNXTIN (PRVARC) = ARC
            END IF
         END IF
         GOTO 519
      END IF

C * corrigir a capacidade residual dos nós digitalizados cortadas *

      ARC = PRDCSR (I)
      IF (ARC.GT.0) THEN
         DELX = DELX-X (ARC)
```

```
      MAIS
         DELX = DELX-U (-ARC)
      END IF

C ********** final da digitalização do nó i para o caso de déficit
positivo ****

              MAIS

C nó digitalização ******* I para o caso de défice negativo ****

      NAUGNOD = 0

      PRVARC = 0
      ARC = TFSTIN (I)
520 IF (ARC.GT.0) THEN
        IF (RC (ARC) .EQ. 0) THEN
           IF (X (ARC) .gt. 0) THEN
              NODE2 = STARTN (ARC)
              IF (.NOT. MARK (NODE2)) THEN
                 PRDCSR (NODE2) = ARC
                 IF (DFCT (NODE2) .GT.0) THEN
                    NAUGNOD = + 1 NAUGNOD
                    SAVE (NAUGNOD) = NODE2
                 END IF
                 NLABEL = + 1 NLABEL
                 LABEL (NLABEL) = NODE2
                 MARK (NODE2) =. TRUE.
                 DELX = DELX + X (ARC)
              END IF
           END IF
           PRVARC = ARC
           ARC = TNXTIN (ARC)
        MAIS
           TMPARC = ARC
           ARC = TNXTIN (ARC)
           TNXTIN (TMPARC) = -1
           IF (PRVARC .EQ. 0) THEN
              TFSTIN (I) = ARC
           MAIS
              TNXTIN (PRVARC) = ARC
           END IF
        END IF
        GOTO 520
      END IF

      PRVARC = 0
      ARC = TFSTOU (I)
521 IF (ARC.GT.0) THEN
        IF (RC (ARC) .EQ. 0) THEN
           IF (U (ARC) .gt. 0) THEN
              NODE2 = ENDN (ARC)
              IF (.NOT. MARK (NODE2)) THEN
                 PRDCSR (NODE2) = - ARC
                 IF (DFCT (NODE2) .GT.0) THEN
                    NAUGNOD = + 1 NAUGNOD
                    SAVE (NAUGNOD) = NODE2
                 END IF
                 NLABEL = + 1 NLABEL
                 LABEL (NLABEL) = NODE2
                 MARK (NODE2) =. TRUE.
```

```
                  DELX = DELX + U (ARC)
               END IF
            END IF
            PRVARC = ARC
            ARC = TNXTOU (ARC)
         MAIS
            TMPARC = ARC
            ARC = TNXTOU (ARC)
            TNXTOU (TMPARC) = -1
            IF (PRVARC .EQ. 0) THEN
               TFSTOU (I) = ARC
            MAIS
               TNXTOU (PRVARC) = ARC
            END IF
         END IF
         GOTO 521
      END IF

      ARC = PRDCSR (I)
      IF (ARC.GT.0) THEN
         DELX = DELX-X (ARC)
      MAIS
         DELX = DELX-U (-ARC)
      END IF
      END IF

C ****** ADD DEFICIT DE NÓ digitalizada para DM ******

       DM = DM + DFCT (I)

C Verifique se o conjunto de nós digitalizados correspondem
C para uma direção subida dual; se sim, realizar uma
Preço C ajuste da etapa, caso contrário, continue rotulagem

       IF (NSCAN.LT.NLABEL) THEN
         IF (CHAVE) ir para 210
         IF ((DELX.GE.DM) .E. (DELX.GE.-DM)) Vá para 210
       END IF

C ************* tentar uma mudança de preço ***********

C Nota que, desde DELX-ABS (DM) é uma superestimativa de ascensão
inclinação, nós
C, ocasionalmente, pode tentar uma direção que não é realmente um
sentido de subida.
C Neste caso, as rotinas ANCNTx voltar com SAIR conjunto para .FALSE.
. O
Código principal C, por sua vez, em seguida, tenta marcar mais alguns
nós.

          IF (POSIT) THEN
            ASCNT1 CALL (DM, DELX, NLABEL, AUGNOD, FEASBL,
     $ SWITCH, NSCAN, NODE, PrevNode)
            NUM_ASCNT = + 1 NUM_ASCNT
          MAIS
            ASCNT2 CALL (DM, DELX, NLABEL, AUGNOD, FEASBL,
     $ SWITCH, NSCAN, NODE, PrevNode)
            NUM_ASCNT = + 1 NUM_ASCNT
          END IF
          IF (.NOT.FEASBL) ir para 400
          IF (.NOT.SWITCH) ir para 100
```

```
          IF ((switch) .E. (AUGNOD.GT.0)) THEN
            NAUGNOD = 1
            SAVE (1) = AUGNOD
          END IF

C Verifique se AUGMENTATION é possível.
C SE NÃO VOLTAR PARA VARRER outro nó.

210 CONTINUAR

          IF (NAUGNOD.EQ.0) ir para 120

C Faça o aumento.

          FAZER 96 J = 1, NAUGNOD
          NUM_AUGM = + 1 NUM_AUGM
          AUGNOD = SAVE (J)
          IF (POSIT) THEN
               CHAMADA AUGFL1 (AUGNOD, NODE, PrevNode)
          MAIS
               CHAMADA AUGFL2 (AUGNOD, NODE, PrevNode)
          END IF
96 CONTINUAR

C ** RETURN para assumir outro nó W / DEFICIT diferente de zero **

         Ir para 100

C ******* problema é encontrado para ser inviável
**********************

400 PRINT *, "o problema é encontrado para ser inviável."
      FEASBL = .FALSE.
      RETURN
      FIM




      SUBROUTINE SHORT_PATH (nó, CAPTHRESH)

C Esta rotina é um método mais curto caminho modificado para encontrar
C mais curto aumentando caminhos a partir de nó. Ele usa
essencialmente
O método de C Dijkstra com uma estrutura de dados heap binário mas
C ignora todos os arcos com capacidade menor que o parâmetro CAPTHRESH

      NONE IMPLICIT
      INTEGER STARTN, ENDN, U, X, RC, DFCT, D, PRDCSR, FOU, FIN,
NXTIN, NXTOU
      INTEGER HP, Q, N, NA, GRANDE, NHP, OPNODE, DV, K, K2, HP2, HP3,
HP1, I, DP1
      INTEGER DX, IB, ROOT, maxd, ARC, NARC, CURNODE, NOLIST, DK, DIFF
      INTEGER início, fim, LIST, CAPTHRESH, NODE
      LÓGICO * 1 MARK
      COMUNS / matrizes / STARTN / ARRAYE / ENDN / ARRAYU / U / ARRAYX
/ X / ARRAY9 / RC
     $ / Arrayb / DFCT / BLK1 / D / BLK2 / PRDCSR / BLK3 / FOU / BLK4
/ NXTOU / BLK5 / FIN
```

```
     $ / BLK6 / NXTIN / BLK9 / MARK / BLK10 / HP / BLK11 / Q / BLK12 /
LIST
      COMUM / L / N, NA, GRANDE

      DIMENSÃO STARTN (1), ENDN (1), L (1), X (1), V (1), DFCT (1)
      DIMENSÃO D (1), fou (1), NXTOU (1), MARK (1)
      DIMENSÃO FIN (1), NXTIN (1), HP (1), PRDCSR (1), Q (1), LISTA
(1)


      FAZER 10 I = 1, N
        D (I) = GRANDE
        Q (I) = 0
        MARK (I) =. FALSE.
10 CONTINUAR


      D (nó) = 0
      PRDCSR (nó) = 0
      NOLIST = 0
      NHP = 0
      CURNODE = NÓ

18 CONTINUAR

      ARC = FOU (CURNODE)
20 IF (ARC.GT.0) THEN
        IF ((RC (ARC) .GE.0) .E. (U (ARC) .GT.CAPTHRESH)) THEN
          OPNODE = ENDN (ARC)
          DV = D (CURNODE) + RC (ARC)
          IF (D (OPNODE) .GT.DV) THEN
           D (OPNODE) = DV
           PRDCSR (OPNODE) = -ARC
           IF (Q (OPNODE) .EQ.0) THEN

C INSERIR OPNODE na pilha

             NHP = NHP + 1
             Q (OPNODE) = NHP
           END IF

C atualizar o MONTÃO

           K = Q (OPNODE)
25 K2 = K / 2
           IF (K2 .gt. 0) THEN
            HP2 = HP (K2)
            IF (DV .lt. D (HP2)) THEN
              HP (K) = HP2
              Q (HP2) = K
              K = K2
              IR PARA 25
            END IF
           END IF

           HP (K) = OPNODE
           Q (OPNODE) = K
         END IF
        END IF
        ARC = NXTOU (ARC)
        IR PARA 20
```

```
      END IF


      ARC = FIN (CURNODE)
30 IF (ARC.GT.0) THEN
        IF ((RC (ARC) .LE.0) .E. (X (ARC) .GT.CAPTHRESH)) THEN
          OPNODE = STARTN (ARC)
          DV = D (CURNODE) -RC (ARC)
          IF (D (OPNODE) .GT.DV) THEN
           D (OPNODE) = DV
           PRDCSR (OPNODE) = ARC
           IF (Q (OPNODE) .EQ. 0) THEN

C INSERIR OPNODE na pilha

             NHP = NHP + 1
             Q (OPNODE) = NHP
           END IF

C atualizar o MONTÃO

           K = Q (OPNODE)
35 K2 = K / 2
           IF (K2 .gt. 0) THEN
            HP2 = HP (K2)
            IF (DV .lt. D (HP2)) THEN
              HP (K) = HP2
              Q (HP2) = K
              K = K2
              IR PARA 35
            END IF
           END IF

           HP (K) = OPNODE
           Q (OPNODE) = K
         END IF
        END IF
        ARC = NXTIN (ARC)
        IR PARA 30
      END IF


C MARK NODE PERMANENTEMENTE LABELED

      NOLIST = NOLIST + 1
      LIST (NOLIST) = CURNODE
      MARK (CURNODE) =. TRUE.


C Retire o novo nó atual da pilha
C verificar se ele tem DEFICIT POSITIVO

      IF (NHP.EQ.0) ir para 130
      CURNODE = HP (1)
      Q (CURNODE) = 0
      IF (DFCT (CURNODE) .GT.0) ir para 130
      NHP = NHP - 1

C Verifique se o MONTÃO ESTÁ VAZIO

      IF (NHP.EQ.0) Vá para 18
```

```
C atualizar o MONTÃO

80 HP1 = HP (NHP + 1)
       DP1 = D (HP1)
       K = 1
90 K2 = 2 * K
       HP2 = HP (K2)
       IF (K2-PNC) 100110120
100 HP3 = HP (K2 + 1)
       IF (D (HP2) .lt. D (HP3)) Vá para 110
       HP2 = HP3
       K2 K2 + 1 =
110 IF (DP1 .LE. D (HP2)) Vá para 120
       HP (K) = HP2
       Q (HP2) = K
       K = K2
       IR PARA 90
120 HP (K) = HP1
       Q (HP1) = K
       IR PARA 18

130 CONTINUAR

C MENOR aumentando PATH foi calculada

C atualizar a redução dos custos de todos os arcos

      Maxd = D (CURNODE)
      FAZER 140 I = 1, NOLIST
        K = LIST (I)
        DK = D (K)
        DIFF = maxd-DK
        ARC = FOU (K)
134 IF (ARC.GT.0) THEN
          END = ENDN (ARC)
          IF (.NOT.MARK (END)) THEN
            RC (ARC) = RC (ARC) --diff
          MAIS
            RC (ARC) = RC (ARC) -D (END) + DK
          END IF
          ARC = NXTOU (ARC)
          GOTO 134
        END IF
        ARC = FIN (K)
136 IF (ARC.GT.0) THEN
          START = STARTN (ARC)
          IF (.NOT.MARK (START)) THEN
            RC (ARC) = RC (ARC) + DIFF
          END IF
          ARC = NXTIN (ARC)
          GOTO 136
        END IF
140 CONTINUAR

C Execute o aumento E SAÍDA

      DX = MIN0 (DFCT (CURNODE), - DFCT (nó))
      IF (DX.GT.0) THEN
       IB = CURNODE
```

```
150 IF (PRDCSR (IB) .NE.0) THEN
          ARC = PRDCSR (IB)
          IF (ARC.GT.0) THEN
            DX = MIN0 (DX, X (ARC))
            IB = ENDN (ARC)
          MAIS
            DX = MIN0 (DX, U (-ARC))
            IB = STARTN (-ARC)
          END IF
          GOTO 150
        END IF

        ROOT = IB

C atualização ******* a déficits de fluxo e *******

          DFCT (CURNODE) = DFCT (CURNODE) -DX
          DFCT (ROOT) = DFCT (ROOT) + DX
          IB = CURNODE
160 IF (IB.NE.ROOT) THEN
           ARC = PRDCSR (IB)
           IF (ARC.GT.0) THEN
             X (ARC) = X (ARC) -DX
             U (ARC) = U (ARC) + DX
             IB = ENDN (ARC)
           MAIS
             NARC = -ARC
             X (NARC) = X (NARC) + DX
             U (NARC) = U (NARC) -DX
             IB = STARTN (NARC)
           END IF
           GOTO 160
         END IF
       END IF

       RETURN
       FIM
```

CHECKCOMPSLACK SUBROUTINE

```
C Esta sub-rotina verifica frouxidão complementar.
C É utilizado para fins de diagnóstico.

      Implícita INTEGER (AZ)

      COMUNS / matrizes / STARTN / ARRAYE / ENDN / ARRAYU / U / ARRAYX
/ X / ARRAY9 / RC
     $ / Arrayb / DFCT / BLK3 / FOU / BLK4 / NXTOU / BLK5 / FIN
     $ / BLK6 / NXTIN / L / N, NA, GRANDE

      DIMENSÃO STARTN (1), ENDN (1), L (1), X (1), V (1), DFCT (1)
      DIMENSÃO fou (1), NXTOU (1)
      DIMENSÃO FIN (1), NXTIN (1)


      FAZER 100 ARC = 1, NA
```

```
        IF ((RC (ARC) .gt. 0) .E. (X (ARC) .gt. 0)) THEN
            PRINT *, 'frouxidão COMPLEMENTAR VIOLADA ", ARC
            PRINT *, RC (ARC), X (ARC)
            PAUSA
        MAIS
          IF ((RC (ARC) .lt. 0) .E. (U (ARC) .GT.0)) THEN
            PRINT *, 'frouxidão COMPLEMENTAR VIOLADA ", ARC
            PRINT *, RC (ARC), U (ARC)
            PAUSA
          END IF
        END IF
100 CONTINUAR

      RETURN
      FIM



      PRINTFLOWS sub-rotina (nó)

C Esta sub-rotina imprime o défice e os fluxos de
C arcos incidente ao nó. É utilizado para fins de diagnóstico
C no caso de um problema inviável.
C
      Implícita INTEGER (AZ)

      COMUNS / matrizes / STARTN / ARRAYE / ENDN / ARRAYU / U / ARRAYX
/ X
     $ / Arrayb / DFCT / BLK3 / FOU / BLK4 / NXTOU / BLK5 / FIN / BLK6
/ NXTIN

      DIMENSÃO STARTN (1), ENDN (1), L (1), X (1), DFCT (1)
      DIMENSÃO fou (1), NXTOU (1)
      DIMENSÃO FIN (1), NXTIN (1)


      PRINT *, «défice (IE, rede de fluxo OUT) de um nó = ', DFCT (nó)
      PRINT *, "fluxos e capacidades dos INCIDENTE arcos de nó ', NODE
      IF (FOU (nó) .EQ.0) THEN
        PRINT *, 'Não arcos de saída'
      MAIS
        ARC = FOU (nó)
5 IF (ARC.GT.0) THEN
          PRINT *, "ARC", ARC, 'entre nós', NODE, ENDN (ARC)
          PRINT *, 'FLUXO =', X (ARC)
          PRINT *, "capacidade residual = ', U (ARC)
          ARC = NXTOU (ARC)
          IR PARA 5
        END IF
      END IF

      IF (FIN (nó) .EQ.0) THEN
        PRINT *, "não seus arcos de entrada '
      MAIS
        ARC = FIN (nó)
10 IF (ARC.GT.0) THEN
          PRINT *, "ARC", ARC, 'entre nós', STARTN (ARC), NODE
          PRINT *, 'FLUXO =', X (ARC)
          PRINT *, "capacidade residual = ', U (ARC)
          ARC = NXTIN (ARC)
          IR PARA 10
```

```
        END IF
      END IF

      RETURN
      FIM



      SUBROUTINE AUGFL1 (AUGNOD, NODE, PrevNode)

C Esta sub-rotina executa o passo de aumento de fluxo.
CA caminho de aumento de fluxo foi identificado na digitalização
Passo C e aqui o fluxo de todos os arcos positivamente (negativamente)
C orientada no caminho de aumento de fluxo é diminuída (aumentada)
C para diminuir o défice total.

      Implícita INTEGER (AZ)
      COMUNS / matrizes / STARTN / ARRAYE / ENDN / ARRAYU / U / ARRAYX
/ X
     $ / Arrayb / DFCT / BLK2 / PRDCSR
      DIMENSÃO STARTN (1), ENDN (1), L (1), X (1), DFCT (1), PRDCSR
(1)
      COMUM / BLK14 / NXTQUEUE
      DIMENSÃO NXTQUEUE (1)


CA fluir aumentando caminho terminando no AUGNOD foi encontrado.
C Primeira determinar DX, a quantidade de mudança de fluxo.

      DX = -DFCT (AUGNOD)
      IB = AUGNOD
500 IF (PRDCSR (IB) .NE.0) THEN
        ARC = PRDCSR (IB)
        IF (ARC.GT.0) THEN
          DX = MIN0 (DX, X (ARC))
          IB = STARTN (ARC)
        MAIS
          DX = MIN0 (DX, U (-ARC))
          IB = ENDN (-ARC)
        END IF
        GOTO 500
      END IF
      ROOT = IB
      DX = MIN0 (DX, DFCT (ROOT))
      IF (DX .LE. 0) RETURN

C ***** Atualizar o fluxo através da diminuição (aumento) o fluxo de
C todos os arcos positivamente (negativamente) no fluxo orientado
Caminho de aumento C. Ajuste os déficits em conformidade. *****

      DFCT (AUGNOD) = DFCT (AUGNOD) + DX
      IF (NXTQUEUE (AUGNOD) .EQ.0) THEN
            NXTQUEUE (PrevNode) = AUGNOD
            NXTQUEUE (AUGNOD) = NÓ
            PrevNode = AUGNOD
      END IF
      DFCT (ROOT) = DFCT (ROOT) -DX
      IB = AUGNOD
501 IF (IB.NE.ROOT) THEN
```

```
        ARC = PRDCSR (IB)
        IF (ARC.GT.0) THEN
          X (ARC) = X (ARC) -DX
          U (ARC) = U (ARC) + DX
          IB = STARTN (ARC)
        MAIS
          NARC = -ARC
          X (NARC) = X (NARC) + DX
          U (NARC) = U (NARC) -DX
          IB = ENDN (NARC)
        END IF
        GOTO 501
      END IF
      RETURN
      FIM



      SUBROUTINE ASCNT1 (DM, DELX, NLABEL, AUGNOD, FEASBL, SWITCH,
     $ NSCAN, StartNode, PrevNode)

C Esta sub-rotina realiza a multi-node
Preço C passo de ajuste. Ele primeiro verifica se o conjunto
C digitalizados de nodos corresponde a uma direcção de subida duplo.
C Se sim, então diminuir o preço de todos os nós digitalizados.
C Há duas possibilidades de ajuste de preços:
C Se o interruptor = .TRUE. em seguida, o conjunto de nós
digitalizadas
C corresponde a uma direcção primária de máxima
C taxa de subida, caso em que o preço de tudo digitalizado
Nós C são diminuiu até o próximo ponto de interrupção no
C custo dupla é encontrado. Neste ponto, alguns arco
C torna-se nó (s) equilibrada e mais são adicionados à
C rotulados definido.
C Se o interruptor = .FALSE. em seguida, os preços de todos os nós
digitalizados
C são diminuídos até que a taxa de subida se torna
C negativo (o que corresponde ao ajuste de preço
C passo em que tanto a linha de busca e o degenerado
C iteração subida são implementadas).

      Implícita INTEGER (AZ)

      LÓGICO * 1 SCAN, MARK, SWITCH, FEASBL
      COMUNS / matrizes / STARTN / ARRAYE / ENDN / ARRAYU / U / ARRAYX
/ X / ARRAY9 / RC
     $ / Arrayb / DFCT / BLK1 / LABEL / BLK2 / PRDCSR / BLK3 / FOU /
BLK4 /
     $ NXTOU / BLK5 / FIN / BLK6 / NXTIN / BLK7 / SAVE / BLK8 / SCAN /
BLK9 / MARK
     $ / L / N, NA, GRANDE
      COMUM / BLK10 / TFSTOU / BLK11 / TNXTOU / BLK12 / TFSTIN / BLK13
/ TNXTIN
      COMUM / ASCBLK / B
      COMUM / BLK14 / NXTQUEUE
      DIMENSÃO NXTQUEUE (1)
      DIMENSÃO TFSTOU (1), TNXTOU (1), TFSTIN (1), TNXTIN (1)
      DIMENSÃO STARTN (1), ENDN (1), L (1), X (1), V (1), DFCT (1), na
etiqueta (1)
      DIMENSÃO PRDCSR (1), fou (1), NXTOU (1), FIN (1), NXTIN (1)
```

```
      DIMENSÃO SAVE (1), SCAN (1), Mark (1)

C loja os arcos entre o conjunto de nós digitalizadas e
C seu complemento no SAVE e calcular DELPRC, o stepsize
C para o próximo ponto de interrupção no sistema duplo de custo na
direcção
C da diminuição dos preços dos nós digitalizados.

      DELPRC = GRANDE
      DLX = 0
      NSalvar = 0

C Calcule a matriz SALVAR de arcos através do corte de varredura
Nós C de uma maneira diferente, dependendo se NSCAN> N / 2 ou não.
C Isto é feito para eficiência.

      IF (NSCAN.LE.N / 2) THEN
      DO 1 I = 1, NSCAN
        NÓ = LABEL (I)
          ARC = FOU (nó)
500 IF (ARC.GT.0) THEN

C ARC é um arco apontando a partir do conjunto de nós digitalizados
para o seu complemento.

            NODE2 = ENDN (ARC)
            IF (.NOT.SCAN (NODE2)) THEN
              NSalvar = nSalvar + 1
              SAVE (nSalvar) = ARC
              RDCOST = RC (ARC)
      IF ((RDCOST.EQ.0) .E. (PRDCSR (NODE2) .NE.ARC)) DLX = DLX + X
(ARC)
            IF ((RDCOST.LT.0) .E. (RDCOST.GT.-DELPRC)) DELPRC = -
RDCOST
            END IF
            ARC = NXTOU (ARC)
            GOTO 500
          END IF
          ARC = FIN (nó)

501 IF (ARC.GT.0) THEN

C ARC é um arco apontando para o conjunto de nós digitalizados a
partir de seu complemento.

            NODE2 = STARTN (ARC)
            IF (.NOT.SCAN (NODE2)) THEN
              NSalvar = nSalvar + 1
              SAVE (nSalvar) = - ARC
              RDCOST = RC (ARC)
      IF ((RDCOST.EQ.0) .E. (PRDCSR (NODE2) .NE.-ARC)) DLX = DLX + U
(ARC)
              IF ((RDCOST.GT.0) .E. (RDCOST.LT.DELPRC)) DELPRC =
RDCOST
            END IF
            ARC = NXTIN (ARC)
            GOTO 501
          END IF
1 CONTINUAR

      MAIS
```

```
      NÓ DO 2 = 1, N
        IF (SCAN (nó)) Vá para 2
          ARC = FIN (nó)
502 IF (ARC.GT.0) THEN
            NODE2 = STARTN (ARC)
            IF (SCAN (NODE2)) THEN
              NSalvar = nSalvar + 1
              SAVE (nSalvar) = ARC
              RDCOST = RC (ARC)
      IF ((RDCOST.EQ.0) .E. (PRDCSR (nó) .NE.ARC)) DLX = DLX + X (ARC)
            IF ((RDCOST.LT.0) .E. (RDCOST.GT.-DELPRC)) DELPRC = -
RDCOST
            END IF
            ARC = NXTIN (ARC)
            GOTO 502
          END IF
          ARC = FOU (nó)
503 IF (ARC.GT.0) THEN
            NODE2 = ENDN (ARC)
            IF (SCAN (NODE2)) THEN
              NSalvar = nSalvar + 1
              SAVE (nSalvar) = - ARC
              RDCOST = RC (ARC)
      IF ((RDCOST.EQ.0) .E. (PRDCSR (nó) .NE.-ARC)) DLX = DLX + U
(ARC)
              IF ((RDCOST.GT.0) .E. (RDCOST.LT.DELPRC)) DELPRC =
RDCOST
            END IF
            ARC = NXTOU (ARC)
            GOTO 503
          END IF
2 CONTINUAR
      END IF

C Verifique se o conjunto de nós digitalizados verdadeiramente
corresponde
C a uma direcção de subida duplo. Aqui DELX + DLX é a exata
C su m do fluxo em arcos do conjunto digitalizado para o
C set unscanned mais o (capacidade - flow) em arcos de
C o conjunto unscanned ao conjunto digitalizado.

C Se este não for o caso, coloque o interruptor para .TRUE.
C e sair do sub-rotina.

      IF (DELX + DLX.GE.DM) THEN
        CHAVE = .TRUE.
        AUGNOD = 0
        DO 3 I = NSCAN + 1, NLABEL
          NÓ = LABEL (I)
          IF (DFCT (nó) .LT.0) AUGNOD = NÓ
3 CONTINUAR
        RETURN
      END IF
      DELX = DELX + DLX

C Verifique se o problema é viável

4 IF (DELPRC.EQ.LARGE) THEN

C que pode diminuir o custo dupla sem limites.
```

```
C Portanto, o problema primal é inviável.

        FEASBL = .FALSE.
        RETURN
      END IF

C diminuição dos preços dos nós digitalizados, adicione mais
Nós C do conjunto rotulado e verificar se um nó recém-marcado
C tem déficit negativo.

      IF (CHAVE) THEN
        AUGNOD = 0
        FAZER 7 I = 1, nSalvar
          ARC = SAVE (I)
          IF (ARC.GT.0) THEN
            RC (ARC) = RC (ARC) + DELPRC
            IF (RC (ARC) .EQ.0) THEN
              NODE2 = ENDN (ARC)
              IF (TNXTOU (ARC) .lt. 0) THEN
                TNXTOU (ARC) = TFSTOU (STARTN (ARC))
                TFSTOU (STARTN (ARC)) = ARC
              END IF
              IF (TNXTIN (ARC) .lt. 0) THEN
                TNXTIN (ARC) = TFSTIN (NODE2)
                TFSTIN (NODE2) = ARC
              END IF
              PRDCSR (NODE2) = ARC
              IF (DFCT (NODE2) .LT.0) THEN
                AUGNOD = NODE2
              MAIS
                IF (.NOT.MARK (NODE2)) THEN
                 MARK (NODE2) =. TRUE.
                 NLABEL = + 1 NLABEL
                 LABEL (NLABEL) = NODE2
                END IF
              END IF
            END IF
          MAIS
            ARC = -ARC
            RC (ARC) = RC (ARC) -DELPRC
            IF (RC (ARC) .EQ.0) THEN
              NODE2 = STARTN (ARC)
              IF (TNXTOU (ARC) .lt. 0) THEN
                TNXTOU (ARC) = TFSTOU (NODE2)
                TFSTOU (NODE2) = ARC
              END IF
              IF (TNXTIN (ARC) .lt. 0) THEN
                TNXTIN (ARC) = TFSTIN (ENDN (ARC))
                TFSTIN (ENDN (ARC)) = ARC
              END IF
              PRDCSR (NODE2) = - ARC
              IF (DFCT (NODE2) .LT.0) THEN
                AUGNOD = NODE2
              MAIS
               IF (.NOT.MARK (NODE2)) THEN
                MARK (NODE2) =. TRUE.
                NLABEL = + 1 NLABEL
                LABEL (NLABEL) = NODE2
               END IF
              END IF
            END IF
```

```
          END IF
7 CONTINUAR
        RETURN

      MAIS

C diminuem os preços dos nós examinadas pelo DELPRC.
C Ajuste o fluxo de arco para manter frouxidão complementar com
C os preços.

      NB = 0
      FAZER 6 I = 1, nSalvar
        ARC = SAVE (I)
        IF (ARC.GT.0) THEN
            T1 = RC (ARC)
            IF (T1.EQ.0) THEN
             T2 = X (ARC)
             T3 = STARTN (ARC)
             DFCT (T3) = DFCT (T3) -T2
             IF (NXTQUEUE (T3) .EQ.0) THEN
                NXTQUEUE (PrevNode) = T3
                NXTQUEUE (T3) = startNode
                PrevNode = T3
             END IF
             T3 = ENDN (ARC)
             DFCT (T3) = DFCT (T3) + T2
             IF (NXTQUEUE (T3) .EQ.0) THEN
                NXTQUEUE (PrevNode) = T3
                NXTQUEUE (T3) = startNode
                PrevNode = T3
             END IF

             U (ARC) = U (ARC) + T2
             X (ARC) = 0
            END IF
          RC (ARC) = T1 + DELPRC
          IF (RC (ARC) .EQ.0) THEN
             DELX = DELX + X (ARC)
             NB = NB + 1
             PRDCSR (NB) = ARC
          ENDIF
        MAIS
          ARC = -ARC
          T1 = RC (ARC)
          IF (T1.EQ.0) THEN
            T2 = U (ARC)
            T3 = STARTN (ARC)
            DFCT (T3) = DFCT (T3) + T2
            IF (NXTQUEUE (T3) .EQ.0) THEN
                NXTQUEUE (PrevNode) = T3
                NXTQUEUE (T3) = startNode
                PrevNode = T3
             END IF

            T3 = ENDN (ARC)
            DFCT (T3) = DFCT (T3) -T2
            IF (NXTQUEUE (T3) .EQ.0) THEN
                NXTQUEUE (PrevNode) = T3
                NXTQUEUE (T3) = startNode
                PrevNode = T3
             END IF
```

```
            X (ARC) = X (ARC) + T2
            U (ARC) = 0
          END IF
          RC (ARC) = T1-DELPRC
          IF (RC (ARC) .EQ.0) THEN
             DELX = DELX + U (ARC)
             NB = NB + 1
             PRDCSR (NB) = ARC
          END IF
        END IF
6 CONTINUAR
      END IF

      IF (DELX.LE.DM) THEN

C O conjunto de nós digitalizados ainda corresponde a um
C dual (possivelmente degenerado) direção de subida. Computar
C o stepsize DELPRC para o próximo ponto de interrupção no
C custo dual.

        DELPRC = GRANDE
        FAZER 10 I = 1, nSalvar
          ARC = SAVE (I)
          IF (ARC.GT.0) THEN
            RDCOST = RC (ARC)
            IF ((RDCOST.LT.0) .E. (RDCOST.GT.-DELPRC)) DELPRC = -
RDCOST
          MAIS
            ARC = -ARC
            RDCOST = RC (ARC)
            IF ((RDCOST.GT.0) .E. (RDCOST.LT.DELPRC)) DELPRC = RDCOST
          END IF
10 CONTINUAR
        IF ((DELPRC.NE.LARGE) .OR. (DELX.LT.DM)) Vá para 4
      END IF

C Adicionar novos arcos equilibradas para o super conjunto de arcos
equilibradas.

      FAZER 9 I = 1, NB
        ARC = PRDCSR (I)
          IF (TNXTIN (ARC) .EQ.-1) THEN
            J = ENDN (ARC)
            TNXTIN (ARC) = TFSTIN (J)
            TFSTIN (J) = ARC
          END IF
          IF (TNXTOU (ARC) .EQ.-1) THEN
            J = STARTN (ARC)
            TNXTOU (ARC) = TFSTOU (J)
            TFSTOU (J) = ARC
          END IF
9 CONTINUAR

      RETURN
      FIM




      SUBROUTINE AUGFL2 (AUGNOD, NODE, PrevNode)
```

```
      Implícita INTEGER (AZ)
      COMUNS / matrizes / STARTN / ARRAYE / ENDN / ARRAYU / U / ARRAYX
/ X
     $ / Arrayb / DFCT / BLK2 / PRDCSR
      COMUM / BLK14 / NXTQUEUE
      DIMENSÃO NXTQUEUE (1)
      DIMENSÃO STARTN (1), ENDN (1), L (1), X (1), DFCT (1), PRDCSR
(1)

      DX = DFCT (AUGNOD)
      IB = AUGNOD
500 IF (PRDCSR (IB) .NE.0) THEN
        ARC = PRDCSR (IB)
        IF (ARC.GT.0) THEN
          DX = MIN0 (DX, X (ARC))
          IB = ENDN (ARC)
        MAIS
          DX = MIN0 (DX, U (-ARC))
          IB = STARTN (-ARC)
        END IF
        GOTO 500
       END IF
      ROOT = IB
      DX = MIN0 (DX, -DFCT (ROOT))
      IF (DX .LE. 0) RETURN

C Atualizar a déficits de fluxo e

      DFCT (AUGNOD) = DFCT (AUGNOD) -DX
      IF (NXTQUEUE (AUGNOD) .EQ.0) THEN
            NXTQUEUE (PrevNode) = AUGNOD
            NXTQUEUE (AUGNOD) = NÓ
            PrevNode = AUGNOD
      END IF
      DFCT (ROOT) = DFCT (ROOT) + DX
      IB = AUGNOD
501 IF (IB.NE.ROOT) THEN
        ARC = PRDCSR (IB)
        IF (ARC.GT.0) THEN
          X (ARC) = X (ARC) -DX
          U (ARC) = U (ARC) + DX
          IB = ENDN (ARC)
        MAIS
          NARC = -ARC
          X (NARC) = X (NARC) + DX
          U (NARC) = U (NARC) -DX
          IB = STARTN (NARC)
        END IF
        GOTO 501
      END IF
      RETURN
      FIM




      SUBROUTINE ASCNT2 (DM, DELX, NLABEL, AUGNOD, FEASBL, SWITCH,
     $ NSCAN, StartNode, PrevNode)
      Implícita INTEGER (AZ)

      LÓGICO * 1 SCAN, MARK, SWITCH, FEASBL
```

```
      COMUNS / matrizes / STARTN / ARRAYE / ENDN / ARRAYU / U / ARRAYX
/ X / ARRAY9 / RC
     $ / Arrayb / DFCT / BLK1 / LABEL / BLK2 / PRDCSR / BLK3 / FOU /
BLK4 /
     $ NXTOU / BLK5 / FIN / BLK6 / NXTIN / BLK7 / SAVE / BLK8 / SCAN /
BLK9 / MARK
     $ / L / N, NA, GRANDE
      COMUM / BLK10 / TFSTOU / BLK11 / TNXTOU / BLK12 / TFSTIN / BLK13
/ TNXTIN
      COMUM / ASCBLK / B
      COMUM / BLK14 / NXTQUEUE
      DIMENSÃO NXTQUEUE (1)
      DIMENSÃO TFSTOU (1), TNXTOU (1), TFSTIN (1), TNXTIN (1)
      DIMENSÃO STARTN (1), ENDN (1), L (1), X (1), V (1), DFCT (1), na
etiqueta (1)
      DIMENSÃO PRDCSR (1), fou (1), NXTOU (1), FIN (1), NXTIN (1)
      DIMENSÃO SAVE (1), SCAN (1), Mark (1)

C ******* augment flui através da ascensão cut & compute preço *****

      DELPRC = GRANDE
      DLX = 0
      NSalvar = 0
      IF (NSCAN.LE.N / 2) THEN
      DO 1 I = 1, NSCAN
        NÓ = LABEL (I)
          ARC = FIN (nó)
500 IF (ARC.GT.0) THEN
            NODE2 = STARTN (ARC)
            IF (.NOT.SCAN (NODE2)) THEN
              NSalvar = nSalvar + 1
              SAVE (nSalvar) = ARC
              RDCOST = RC (ARC)
      IF ((RDCOST.EQ.0) .E. (PRDCSR (NODE2) .NE.ARC)) DLX = DLX + X
(ARC)
            IF ((RDCOST.LT.0) .E. (RDCOST.GT.-DELPRC)) DELPRC = -
RDCOST
            END IF
            ARC = NXTIN (ARC)
            GOTO 500
          END IF
          ARC = FOU (nó)
501 IF (ARC.GT.0) THEN
            NODE2 = ENDN (ARC)
            IF (.NOT.SCAN (NODE2)) THEN
              NSalvar = nSalvar + 1
              SAVE (nSalvar) = - ARC
              RDCOST = RC (ARC)
        IF ((RDCOST.EQ.0) .E. (PRDCSR (NODE2) .NE.-ARC)) DLX = DLX + U
(ARC)
              IF ((RDCOST.GT.0) .E. (RDCOST.LT.DELPRC)) DELPRC =
RDCOST
            END IF
            ARC = NXTOU (ARC)
            GOTO 501
          END IF
1 CONTINUAR
      MAIS
      NÓ DO 2 = 1, N
        IF (SCAN (nó)) Vá para 2
          ARC = FOU (nó)
```

```
502 IF (ARC.GT.0) THEN
            NODE2 = ENDN (ARC)
            IF (SCAN (NODE2)) THEN
              NSalvar = nSalvar + 1
              SAVE (nSalvar) = ARC
              RDCOST = RC (ARC)
      IF ((RDCOST.EQ.0) .E. (PRDCSR (nó) .NE.ARC)) DLX = DLX + X (ARC)
            IF ((RDCOST.LT.0) .E. (RDCOST.GT.-DELPRC)) DELPRC = -
RDCOST
            END IF
            ARC = NXTOU (ARC)
            GOTO 502
          END IF
          ARC = FIN (nó)
503 IF (ARC.GT.0) THEN
            NODE2 = STARTN (ARC)
            IF (SCAN (NODE2)) THEN
              NSalvar = nSalvar + 1
              SAVE (nSalvar) = - ARC
              RDCOST = RC (ARC)
        IF ((RDCOST.EQ.0) .E. (PRDCSR (nó) .NE.-ARC)) DLX = DLX + U
(ARC)
              IF ((RDCOST.GT.0) .E. (RDCOST.LT.DELPRC)) DELPRC =
RDCOST
            END IF
            ARC = NXTIN (ARC)
            GOTO 503
          END IF
2 CONTINUAR
      END IF
      IF (DELX + DLX.GE.-DM) THEN
        CHAVE = .TRUE.
        AUGNOD = 0
        DO 3 I = NSCAN + 1, NLABEL
          NÓ = LABEL (I)
          IF (DFCT (nó) .GT.0) AUGNOD = NÓ
3 CONTINUAR
        RETURN
      END IF
      DELX = DELX + DLX

C ******* verificação de que o problema é viável ********************

4 IF (DELPRC.EQ.LARGE) THEN
        FEASBL = .FALSE.
        RETURN
      END IF

C ***** preços aumentam *****

      IF (CHAVE) THEN
        AUGNOD = 0
        FAZER 7 I = 1, nSalvar
          ARC = SAVE (I)
          IF (ARC.GT.0) THEN
            RC (ARC) = RC (ARC) + DELPRC
            IF (RC (ARC) .EQ.0) THEN
              NODE2 = STARTN (ARC)
              IF (TNXTOU (ARC) .lt. 0) THEN
                TNXTOU (ARC) = TFSTOU (NODE2)
                TFSTOU (NODE2) = ARC
```

```
              END IF
              IF (TNXTIN (ARC) .lt. 0) THEN
                TNXTIN (ARC) = TFSTIN (ENDN (ARC))
                TFSTIN (ENDN (ARC)) = ARC
              END IF
              PRDCSR (NODE2) = ARC
              IF (DFCT (NODE2) .GT.0) THEN
                AUGNOD = NODE2
              MAIS
               IF (.NOT.MARK (NODE2)) THEN
                 MARK (NODE2) =. TRUE.
                NLABEL = + 1 NLABEL
                LABEL (NLABEL) = NODE2
               END IF
              END IF
            END IF
          MAIS
            ARC = -ARC
            RC (ARC) = RC (ARC) -DELPRC
            IF (RC (ARC) .EQ.0) THEN
              NODE2 = ENDN (ARC)
              IF (TNXTOU (ARC) .lt. 0) THEN
                TNXTOU (ARC) = TFSTOU (STARTN (ARC))
                TFSTOU (STARTN (ARC)) = ARC
              END IF
              IF (TNXTIN (ARC) .lt. 0) THEN
                TNXTIN (ARC) = TFSTIN (NODE2)
                TFSTIN (NODE2) = ARC
              END IF
              PRDCSR (NODE2) = - ARC
              IF (DFCT (NODE2) .GT.0) THEN
                AUGNOD = NODE2
              MAIS
               IF (.NOT.MARK (NODE2)) THEN
                MARK (NODE2) =. TRUE.
                NLABEL = + 1 NLABEL
                LABEL (NLABEL) = NODE2
               END IF
              END IF
            END IF
          END IF
7 CONTINUAR
        RETURN

      MAIS

      NB = 0
      FAZER 6 I = 1, nSalvar
        ARC = SAVE (I)
        IF (ARC.GT.0) THEN
            T1 = RC (ARC)
            IF (T1.EQ.0) THEN
             T2 = X (ARC)
             T3 = STARTN (ARC)
             DFCT (T3) = DFCT (T3) -T2
             IF (NXTQUEUE (T3) .EQ.0) THEN
                NXTQUEUE (PrevNode) = T3
                NXTQUEUE (T3) = startNode
                PrevNode = T3
             END IF
```

```
            T3 = ENDN (ARC)
            DFCT (T3) = DFCT (T3) + T2
            IF (NXTQUEUE (T3) .EQ.0) THEN
               NXTQUEUE (PrevNode) = T3
               NXTQUEUE (T3) = startNode
               PrevNode = T3
            END IF

            U (ARC) = U (ARC) + T2
            X (ARC) = 0
          END IF
        RC (ARC) = T1 + DELPRC
        IF (RC (ARC) .EQ.0) THEN
           DELX = DELX + X (ARC)
           NB = NB + 1
           PRDCSR (NB) = ARC
        END IF
      MAIS
        ARC = -ARC
        T1 = RC (ARC)
          IF (T1.EQ.0) THEN
            T2 = U (ARC)
            T3 = STARTN (ARC)
            DFCT (T3) = DFCT (T3) + T2
            IF (NXTQUEUE (T3) .EQ.0) THEN
              NXTQUEUE (PrevNode) = T3
              NXTQUEUE (T3) = startNode
              PrevNode = T3
           END IF

            T3 = ENDN (ARC)
            DFCT (T3) = DFCT (T3) -T2
            IF (NXTQUEUE (T3) .EQ.0) THEN
              NXTQUEUE (PrevNode) = T3
              NXTQUEUE (T3) = startNode
              PrevNode = T3
           END IF

            X (ARC) = X (ARC) + T2
            U (ARC) = 0
          END IF
        RC (ARC) = T1-DELPRC
        IF (RC (ARC) .EQ.0) THEN
           DELX = DELX + U (ARC)
           NB = NB + 1
           PRDCSR (NB) = ARC
      END IF
    END IF
6 CONTINUAR

    END IF
    IF (DELX.LE.-DM) THEN
      DELPRC = GRANDE
      FAZER 10 I = 1, nSalvar
        ARC = SAVE (I)
        IF (ARC.GT.0) THEN
          RDCOST = RC (ARC)
          IF ((RDCOST.LT.0) .E. (RDCOST.GT.-DELPRC)) DELPRC = -
RDCOST
        MAIS
          ARC = -ARC
```

```
            RDCOST = RC (ARC)
            IF ((RDCOST.GT.0) .E. (RDCOST.LT.DELPRC)) DELPRC = RDCOST
          END IF
10 CONTINUAR
        IF ((DELPRC.NE.LARGE) .OR. (DELX.LT.-DM)) Vá para 4
      END IF

C Adicionar novos arcos de equilíbrio para o super conjunto de arcos
equilibradas.

      FAZER 9 I = 1, NB
        ARC = PRDCSR (I)
          IF (TNXTIN (ARC) .EQ.-1) THEN
            J = ENDN (ARC)
            TNXTIN (ARC) = TFSTIN (J)
            TFSTIN (J) = ARC
          END IF
          IF (TNXTOU (ARC) .EQ.-1) THEN
            J = STARTN (ARC)
            TNXTOU (ARC) = TFSTOU (J)
            TFSTOU (J) = ARC
          END IF
9 CONTINUAR

      RETURN
      FIM
```

```
************************************************
************************
```

Leilão Códigos para problemas de atribuição

```
************************************************
************************
```

Desde atribuição
problemas não pode ser gerada pelo programa GRIDGEN, os códigos de atribuição
incluem um gerador problema aleatório embutido. Este gerador pode ser substituído por
outro código que lê um problema de atribuição de um arquivo.

LEILÃO

Este código implementa o leilão para a frente
algoritmo com $ \ e $ scaling para o problema de atribuição simétrica; cf. \
Seção 4.1.

```
C ********************************************** ****************
C
C programa de exemplo de chamada para o ALGORITHM LEILÃO
C
C ESTE MOTORISTA cria um problema SIMÉTRICA CESSÃO
C com igual número de linhas e colunas,
C e chama o SUBROUTINE LEILÃO para encontrar um
C atribuição do valor máximo.
C
```

```
C ********************************************** *****************

        PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

        NONE IMPLICIT
        INTEGER N, NA, A, ISMALL, BEGEPS, ENDEPS, CICLOS
        INTEGER NUMPHASES, STARTINCR, TEST, o lucro
        INTEGER I, J, IA, ARC, NOASS, ICOST, ABSCOST, CURARC
        INTEGER CURCOL, FSTARC, LSTARC, MINCOST, MAXCOST, MCOST
        INTEGER FOUT (maxNodes), COLASS (maxNodes)
        INTEGER ASSIGN (maxNodes), PCOL (maxNodes)
        INTEGER COST (MAXARCS), END (MAXARCS)
        Real * 8 FACTOR, TT1, TT2, TCOST, AVERAGE
        COMUM / ARRAYC / CUSTO / matrizes / END / ARRAYF / FOUT / BK1
/ N, A, ISMALL
     $ / BK2 / ciclos, Médio, NUMPHASES / ARRAYA / ASSIGN / PCOLA /
PCOL

C ********************************************** ***************
C
C problema de geração de código começa AQUI
C o usuário pode substituir este código com um código que lê
C HIS / seu problema de um arquivo
C
C ********************************************** ***************

C Este código inclui UM UNIFORME Random Number Generator
C que retorna um valor na (0,1)

C INITIALIZE Gerador Aleatório

      INTEGER MULT, MODUL, I15, I16, JRAN, ISEED
      RAN REAIS
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN

      ISEED = 13502460
      CHAME SETRAN (ISEED)

      PRINT * ', gerando problema de atribuição de SIMÉTRICA "
      PRINT *, "****************************************** '

C **** LER O número de linhas N & o número de arcos A ****

      PRINT *, 'Digite o número de linhas (e colunas)'
      LEIA *, N
5 PRINT *, 'Digite o número de arcos por ROW (1>) "
      LEIA *, NA
      IF (NA.LT.2) GOTO 5
      PRINT *, 'Digite o mínimo eo custo máximo'
      LEIA *, MINCOST, MAXCOST

C o número de arcos é n * NA

      A = N * NA

C Os arcos incidente a linha i SÃO FOUT (I) PARA FOUT (I + 1) -1
C Além disso, para VIABILIDADE EACH ROW está diretamente ligado
C com a coluna correspondente

      FAZER 20 I = 1, N
        FOUT (I) = 1 + (i-1) * NA
```

```
20 CONTINUAR
      FOUT (N + 1) = A + 1

C gerar a END (ARC) eo custo (ARC), que são os COLUNA
C eo coeficiente custo associado a ARC

      FAZER 25 ARC = 1, A
        END (ARC) = 1 + RAN () * N
        IF ((END (ARC) .GT.N) .OR. (END (ARC) .LT.1)) THEN
          PRINT *, 'Erro no problema de geração de'
          PAUSA
          PARE
        END IF
        COST (ARC) = MINCOST + RAN () * (MAXCOST-MINCOST)
25 CONTINUAR

C MODIFICAR O final da última ARC OUT de cada linha de viabilidade
C e defina seu CUSTO PARA -MAXCOST

      FAZER 30 I = 1, N
        END (FOUT (I + 1 -1)) = I
        CUSTO (FOUT (I + 1) -1) = - MAXCOST
30 CONTINUAR


C
C ********************************************** ***************
C
C PROBLEMA geração de código TERMINA AQUI
C
C ********************************************** ***************

C ESCALA O CUSTO PARA TRABALHAR COM INTEIRO EPSILON

        MAXCOST = 0
        FAZER 35 IA = 1, A
           ABSCOST = IABS (COST (IA))
           IF (ABSCOST.GT.MAXCOST) MAXCOST = ABSCOST
           COST (IA) = CUSTO (IA) * (N + 1)
35 CONTINUAR

C *** ISMALL é um inteiro MUITO PEQUENO PARA O SEU APARELHO ***

        ISMALL = -2000000000

        IF (MAXCOST.GT.INT (ABS (ISMALL) / (N + 1))) THEN
          PRINT *, 'A faixa de custo é muito grande para INTEGER
ARITHMETIC'
          PAUSA
          PARE
        END IF

        MAXCOST MAXCOST = * (N + 1)


C os seguintes parâmetros BEGEPS, FACTOR, ENDEPS E STARTINCR
C são passados para o LEILÃO algoritmo. Valores entre
C (A) MAXCOST / 5 E MAXCOST / 2 para BEGEPS
C (B) 4 e 6 para FACTOR
C (C) N / 10 e 1 para ENDEPS
C (D) 1 AND BEGEPS PARA STARTINCR
```

```
C têm funcionado bem para PROBLEMAS esparsos de grande porte.
C PARA PROBLEMAS DENSOS RECOMENDA-SE QUE
C BEGEPS ser ajustado para um valor menor,
ENDEPS C ser definido como 1,
C STARTINCR ser definido como 1.

        PRINT *, "****************************************** '
        PRINT *, "custo máximo é ', MAXCOST
        PRINT *, 'Digite o EPSILON PARTIDA'
        LER *, BEGEPS
        IF (BEGEPS.LT.1) BEGEPS = 1
        PRINT *, 'Insira o fator EPSILON REDUÇÃO'
        LER *, FACTOR
        ENDEPS = N / 10
        IF (ENDEPS.LT.1) ENDEPS = 1
        IF (ENDEPS.GT.BEGEPS) ENDEPS = BEGEPS
        STARTINCR = BEGEPS / 10
        IF (STARTINCR.LT.1) STARTINCR = 1

        PRINT *, "****************************************** '
        PRINT *, 'começar EPSILON =', BEGEPS
        PRINT *, 'Epsilon redução de fatores =', FACTOR
        PRINT *, «limiar EPSILON ANTES ela é definida como 1 = ',
ENDEPS
        PRINT *, "COMEÇAR MIN DE LICITAÇÃO INCREMENTO = ', STARTINCR
        PRINT *, "CHAMANDO LEILÃO para resolver o problema"
        PRINT *, "****************************************** '

C GET hora de início para o Mac II

      TT1 = longa (362) /60.0

      CHAMADA Leilão (BEGEPS, fator, ENDEPS, STARTINCR)

C GET TÉRMINO TEMPO PARA O MAC II

      TT2 = longa (362) /60.0 - TT1
      PRINT *, "acabado --- tempo de CPU TOTAL ', TT2,' s '
      PRINT *, "****************************************** '

C *** Mostrar resultados ***

X Write (9,2010) CICLOS
X2010 FORMATO ('NO LEILÃO DE CICLOS', I7)
X WRITE (9,2020) MÉDIA
X2020 FORMATO ("número médio de propostas por ciclo", F9.3)
       Write (9,2030) NUMPHASES
2030 FORMATO ('NO DE EPSILON subproblemas RESOLVIDO =', I7)

C VERIFICAÇÃO otimalidade da solução de e calcular o custo

        DO 40 I = 1, N
          COLASS (I) = 0
40 CONTINUAR
        NOASS = 0
        FAZER 50 J = 1, N
           IF (ASSIGN (J) .GT.0) THEN
             NOASS = + 1 NOASS
             COLASS (ASSIGN (J)) = J
           END IF
50 CONTINUAR
```

```
        IF (NOASS.NE.N) THEN
          PRINT *, '# de linhas atribuído não IGUAL # de linhas'
        END IF
        TCOST = 0
        FAZER 60 I = 1, N
          J = COLASS (I)
          IF (J.EQ.0) THEN
            PRINT *, 'ROW', I, 'não é atribuído "
          END IF
          FSTARC = FOUT (I)
          LSTARC FOUT = (I + 1) -1
          MCOST = ISMALL
          FAZER 70 CURARC = FSTARC, LSTARC
            CURCOL = END (CURARC)
            IF (CURCOL.EQ.J) THEN
              IF (MCOST.LT.COST (CURARC)) THEN
                 MCOST = COST (CURARC)
              END IF
            END IF
70 CONTINUAR
          LUCRO = MCOST-PCOL (J)
          FAZER 72 CURARC = FSTARC, LSTARC
             J = END (CURARC)
             TEST = COST (CURARC) -PCOL (J) lucrativos
             IF (TEST.GT.1) THEN
                PRINT *, '1-CS VIOLADOS AT ARC', CURARC
             END IF
72 CONTINUAR
          TCOST = TCOST MCOST + / (N + 1)
60 CONTINUAR
        ESCREVA (9,2100) TCOST
2100 FORMATO ('atribuição de custo =', F14.2)
        PRINT *, 'programa terminou; <CR> PARA SAIR '
        PAUSA
        FIM



C ********************************************** ****************
C
C CODE LEILÃO PARA N por N problemas de atribuição
C
C ESCRITO POR DIMITRI P. Bertsekas
C (MODIFICAÇÕES por Paul TSENG)
C
C versão 1.1, setembro 1990
C
C ESTE CÓDIGO implementa o algoritmo LEILÃO COM E-escala.
C Ele resolve uma seqüência de subproblemas e diminui
C EPSILON por um fator constante entre subproblemas.
C ESTE versão corresponda a um MODO Gauss-Seidel
C e resolve EPSILON subproblemas inexata.
C
C THE CODE é uma versão melhorada de um antigo (SEPT. 1985)
C CODE LEILÃO COM ESCALA E-ESCRITO POR DIMITRI P. Bertsekas
C
C DO CÓDIGO trata o problema como um problema de maximização.
C para resolver um problema de minimização, inverta o sinal da
C ARC CUSTOS antes de chamar LEILÃO, E RECUO DE NOVO
C O SINAL DO custo ótimo na volta do LEILÃO.
C Este código permite ARCS múltipla entre uma linha e uma coluna.
```

```
C
C Esta versão do ALGORITHM LEILÃO é adaptativa. AQUI NO MÍNIMO
INCREMENTO DE LICITAÇÃO C (armazenado no INCR variável) pode ser menor
que
C EPSILON. PARA CADA subproblema, INCR COMEÇA no valor do parâmetro
C STARTINCR (que é passado para a rotina LEILÃO), estando aumentada
C por um fator de 2 no final de cada ciclo, até um valor máximo de
C EPSILON. ESTA FUNÇÃO adaptativo é particularmente eficaz para
C PROBLEMAS densa. TI pode ser derrotado por SELEÇÃO STARTINCR =
BEGEPS
C
C ************************************************
******************
C
C o usuário deve fornecer os seguintes dados problema no FRENTE DA
ESTRELA FORMATO
C (ou seja, todos os arcos da linha SAME são numerados
consecutivamente):
CN = número de linhas (igual número de colunas)
CA = número de arcos
C FOUT (ROW) = Primeiro ARC QUE SAI DE ROW
C COST (ARC) = CUSTO DE ARC
C END (ARC) = COLUNA correspondente ao arco
C
C E os seguintes parâmetros para o algoritmo LEILÃO:
C BEGEPS = Valor inicial EPSILON (não deve ser menor que 1)
C ENDEPS = valor final da EPSILON ANTES ela é definida como 1
C FACTOR = fator pelo qual EPSILON é diminuída ENTRE subproblemas
C STARTINCR = O valor inicial do incremento LICITAÇÃO
ENDEPS C não devem exceder BEGEPS.
C fator deve ser maior do que 1.
C
C FOUT (.) É uma matriz de comprimento N.
C COST (.), END (.) São matrizes de COMPRIMENTO A.
C
C, a solução está contida no ASSIGN Array (.) ONDE
C ASSIGN (COL) Dá o ROW ATRIBUÍDO COL.
C
C Este algoritmo não verifica a inviabilidade do problema.
C TO garantir que o problema é viável o usuário pode ADD
Adicional C ARCS custo muito pequeno.
C
C Este código pode falhar devido a integer overflow IF o número de nós
C ou o intervalo COST (ou ambos) são grandes. Para corrigir esta
situação,
C OS PREÇOS E outras variáveis relacionadas (Max1, max2, TMAX etc.)
devem
C DECLARAR AS REAIS precisão dupla.
C ************************************************ ****************
C ALL problema de dados são integer
C ************************************************ ****************


        SUBROUTINE Leilão (BEGEPS, fator, ENDEPS, STARTINCR)

        PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

        NONE IMPLICIT
        INTEGER NONEWLIST, THRESH, INCR, STARTINCR, INCRFACTOR
        INTEIRO A, K, N, I, J, CURARC, CURCOL
```

```
        ROW INTEIRO, FSTARC, FSTCOL, SNDARC, SNDCOL, TMAX, TMIN,
BSTCOL
        INTEGER Max1, MAX2, TRDARC, EPSILON, BEGEPS, ENDEPS
        INTEGER M, ISMALL, ILARGE, LARGEINCR, CICLOS
        INTEGER NUMPHASES, LSTARC, NOLIST, OLDROW
        INTEGER FOUT (maxNodes), PCOL (maxNodes), LIST (maxNodes)
               INTEGER ASSIGN (maxNodes)
        INTEGER COST (MAXARCS), END (MAXARCS)
        Real * 8 AVERAGE, FACTOR
               COMUM / ARRAYC / CUSTO / matrizes / END / ARRAYF / FOUT
/ BK1 / N, A, ISMALL
     $ / BK2 / ciclos, Médio, NUMPHASES / ARRAYA / ASSIGN / PCOLA /
PCOL

C ********************************************** **************

Check C ******* VALIDADE DOS PARÂMETROS ******* PASSOU

        IF (BEGEPS.LT.1) THEN
          PRINT *, 'Valor inicial EPSILON é inferior a 1'
          PRINT *, 'EXECUÇÃO ABORTADO'
          PARE
        END IF
        IF (ENDEPS.GT.BEGEPS) THEN
          PRINT *, 'ENDEPS parâmetro é maior que o parâmetro BEGEPS'
          PRINT *, 'ENDEPS é definido no valor padrão de 1'
          ENDEPS = 1
        END IF
        IF ((FACTOR.LE.1) .E. (BEGEPS.GT.1)) THEN
          PRINT *, 'Epsilon redução de fatores não é maior que 1'
          PRINT *, 'EXECUÇÃO ABORTADO'
          PARE
        END IF
        IF (STARTINCR.LT.1) THEN
          PRINT *, 'MIN DE LICITAÇÃO incremento é menor que 1'
          PRINT *, 'STARTINCR é definido no valor padrão de 1'
          STARTINCR = 1
        END IF

C ******* INITIALIZATION *******

        EPSILON = BEGEPS
        ILARGE = -ISMALL
        LARGEINCR = INT (ILARGE / 10)
        THRESH = INT (0,2 * N)
        INCRFACTOR = 2
        IF (THRESH.GT.100) THRESH = 100
CICLOS x = 1
X MÉDIA = N
         NUMPHASES = 1
         FAZER 10 J = 1, N
           ASSIGN (J) = 0
           PCOL (J) = ISMALL
10 CONTINUAR

        FOUT (N + 1) = A + 1
        NOLIST = N
        FAZER 20 I = 1, N
            LISTA (I) = I
20 CONTINUAR
```

```
C ********************************************** ***************
C
C Esta implementação do leilão ALGORITHM opera em ciclos.
C Cada ciclo consiste em um lance em cada uma das linhas QUE SEJAM
C não atribuído NO INÍCIO DO CICLO (essas linhas são armazenadas em
C lista de matriz (.)). AS que o ciclo progride NOVO
LINHAS C ficarem não atribuídas; Estes são armazenados em Lista (.)
C e apresentará um lance no próximo ciclo.
C NOTA que apenas uma linha apresenta uma proposta de cada vez; Isto é
conhecido como
C o método de Gauss-Seidel versão do algoritmo. A VERSÃO onde todos
C ROWS não atribuído apresentar uma proposta AO MESMO TEMPO, é
conhecido como de Jacobi
C versão do algoritmo, e não foi implementada. Geralmente
C tende a correr um pouco mais lento do que a versão de Gauss-Seidel,
MAS
C admite um grau mais elevado DE paralelização. A VERSÃO JACOBI
C pode ser preferível ON Uma máquina paralela, mas geralmente é
INFERIOR
C PARA A VERSÃO Gauss-Seidel EM UMA MÁQUINA DE SÉRIE.
C
C ********************************************** ***************
C
C INÍCIO subproblema (FASE DE ESCALA) W / NEW EPSILON
C
C ********************************************** ***************

12 CONTINUAR

        IF (EPSILON.EQ.1) debulhar = 0
        INCR = STARTINCR
        IF (INCR.GT.EPSILON) INCR = EPSILON

C ********************************************** ***************
C
C início do ciclo LEILÃO COM LISTA NOVO
C
C ********************************************** ***************

15 CONTINUAR

C REINICIAR CONTAGEM DO PRÓXIMO lista de linhas não atribuídos

        NONEWLIST = 0

C percorrer a lista atual de linhas não atribuídos

        FAZER 100 I = 1, NOLIST
        ROW = LIST (I)
        FSTARC = FOUT (ROW)
        LSTARC = FOUT (ROW + 1) -1
        FSTCOL = END (FSTARC)

C PRIMEIRA CUIDAR DA caso excepcional ROW tem apenas um ARC

        IF (FSTARC.EQ.LSTARC) THEN
          PCOL (FSTCOL) = PCOL (FSTCOL) + LARGEINCR
            OLDROW = ASSIGN (FSTCOL)
            ASSIGN (FSTCOL) = ROW
            IF (OLDROW.GT.0) THEN
              NONEWLIST = + 1 NONEWLIST
```

```
              LIST (NONEWLIST) = OLDROW
            END IF
          Ir para 100
        END IF

C tomar em seguida CUIDADOS DO CASO regular, onde linha tem ARCS
MÚLTIPLAS

        SNDARC = + 1 FSTARC
        SNDCOL = END (SNDARC)
        Max1 = COST (FSTARC) -PCOL (FSTCOL)
        MAX2 = COST (SNDARC) -PCOL (SNDCOL)
        IF (MAX1.GE.MAX2) THEN
           BSTCOL = FSTCOL
        MAIS
           TMAX = Max1
           Max1 = MAX2
           MAX2 = TMAX
           BSTCOL = SNDCOL
        END IF
        IF (SNDARC.LT.LSTARC) THEN
        TRDARC = + 1 SNDARC
        DO 40 CURARC = TRDARC, LSTARC
           CURCOL = END (CURARC)
           TMAX = COST (CURARC) -PCOL (CURCOL)
           IF (TMAX.GT.MAX2) THEN
             IF (TMAX.GT.MAX1) THEN
                  MAX2 = Max1
                  Max1 = TMAX
                  BSTCOL = CURCOL
             MAIS
                  MAX2 = TMAX
             END IF
           END IF
40 CONTINUAR
        END IF


APOSTAS C linha para BSTCOL AUMENTAR SEU PREÇO, E é atribuído
C TO BSTCOL, enquanto qualquer ROW ATRIBUÍDO BSTCOL TORNA não
atribuídos

          PCOL (BSTCOL) = PCOL (BSTCOL) + Max1-MAX2 + INCR
          OLDROW = ASSIGN (BSTCOL)
          ASSIGN (BSTCOL) = ROW
          IF (OLDROW.GT.0) THEN
            NONEWLIST = + 1 NONEWLIST
            LIST (NONEWLIST) = OLDROW
          END IF

100 CONTINUAR

C ************* FIM DE UM CICLO DE LEILÃO ***************

C OPTIONALLY coletar estatísticas

X Médio = (ciclos * MÉDIA + NOLIST) / (CICLOS + 1)
CICLOS X = CICLOS + 1

C Verifique se ainda existem ROWS não atribuído 'muitos', isto é, se o
C número de linhas não atribuído É MAIOR DO QUE
```

```
C DO THRESH PARAMETER. NÃO SE, REPLACE lista atual com a lista NEW,
C e ir para outro ciclo. Caso contrário, se EPSILON> 1, REDUZIR
EPSILON,
C Redefinir a atribuição ao vazio e REINICIAR LEILÃO;
C IF EPSILON = 1 finalizar.
C também aumentam a LICITAÇÃO MINIMAL INCREMENTO até um máximo
C VALOR DO EPSILON (esta é a característica adaptativa)

        INCR = INCR * INCRFACTOR
        IF (INCR.GT.EPSILON) INCR = EPSILON
        IF (NONEWLIST.GT.THRESH) THEN
            NOLIST = NONEWLIST
            IR PARA 15
        END IF

C ********************************************** **************
C
C FIM DE subproblema (escalonamento FASE)
C
C ********************************************** **************

C ****** SE EPSILON É uma RESCINDIR ******

       IF (EPSILON.EQ.1) THEN
          RETURN
       MAIS

C MAIS REDUZIR EPSILON e redefinir o CESSÃO ESVAZIAR

          NUMPHASES + 1 = NUMPHASES
          EPSILON = INT (EPSILON / FACTOR)
          IF (EPSILON.GT.INCR) EPSILON = INT (EPSILON / FACTOR)
          IF ((EPSILON.LT.1) .OR. (EPSILON.LT.ENDEPS)) EPSILON = 1
          THRESH = INT (THRESH / FACTOR)

X PRINT *, '*** FIM DE UMA FASE DE ESCALA; NEW EPSILON = ', EPSILON

          TMIN = ILARGE
          FAZER 200 J = 1, N
           IF (PCOL (J) .LT.TMIN) TMIN = PCOL (J)
           IF (ASSIGN (J) .GT.0) THEN
            NONEWLIST = + 1 NONEWLIST
            LIST (NONEWLIST) = ASSIGN (J)
            ASSIGN (J) = 0
           END IF
200 CONTINUAR

C Redefinir preço mínimo a ISMALL

          INCR = TMIN-ISMALL
          DO 210 J = 1, N
            PCOL (J) = PCOL (J) -INCR
210 CONTINUAR


C FINAIS PARÂMETROS atualizações antes de voltar para outra ESCALA
FASE

          NOLIST = NONEWLIST
          IF (STARTINCR.LT.EPSILON) STARTINCR = FACTOR * STARTINCR
          IR PARA 12
```

```
        END IF

        FIM


      SUBROUTINE SETRAN (ISEED)
      IMPLICIT real * 8 (AH, OZ), Integer * 4 (IN)
C ***********************************************
**********************
C PORTABLE congruential (uniforme) RANDOM gerador de números:
NEXT_VALUE C = [(7 ** 5) * PREVIOUS_VALUE] MODULO [(2 ** 31) -1]
C
C Este gerador é composto por duas ROTINAS:
C (1) SETRAN - inicializa constantes e SEED
C (2) RRAN - gera um número aleatório REAIS
C
C O GERADOR DE RODA uma máquina com pelo menos 32 bits de precisão.
C DA SEED (ISEED) deve estar no intervalo (1, (2 ** 31 -1)).
C ***********************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      IF (ISEED.LT.1) PARADA 77
      MULT = 16807
      MODUL = 2147483647
      I15 = 2 ** 15
      I16 = 2 ** 16
      JRAN = ISEED
      RETURN
      FIM

      RAN função real ()
      IMPLICIT real * 4 (AH, OZ), Integer * 4 (IN)
C ***********************************************
**********************
C RAN gera um número aleatório real entre 0 e 1
C ***********************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      Ixhi = JRAN / I16
      IXLO = JRAN-Ixhi * I16
      IXALO = IXLO * MULT
      LEFTLO = IXALO / I16
      IXAHI = Ixhi * MULT
      IFULHI = IXAHI + LEFTLO
      IRTLO = IXALO-LEFTLO * I16
      IOVER = IFULHI / I15
      IRTHI = IFULHI-IOVER * I15
      JRAN = ((IRTLO-MODUL) + IRTHI * I16) + IOVER
      IF (JRAN.LT.0) JRAN = JRAN + MODUL
      RAN = FLOAT (JRAN) / FLOAT (MODUL)
      RETURN
      FIM
```

AUCTION_FLP

Este código é o mesmo que o anterior excepto que

```
ele usa a aritmética de ponto flutuante ao atualizar preços. Isso é
útil quando
a gama de custo e a dimensão do problema é grande, em cujo caso
os preços pode transbordar o intervalo inteiro no decurso do
algoritmo.
No entanto, a aritmética de ponto flutuante retarda o código um pouco.

C ********************************************** ****************
C
C programa de exemplo de chamada para o ALGORITHM LEILÃO
C
C ESTE MOTORISTA cria um problema SIMÉTRICA CESSÃO
C com igual número de linhas e colunas,
C e chama o SUBROUTINE LEILÃO para encontrar um
C atribuição do valor máximo.
C
C MODIFICADO POR DAVID Castanon (OCT. 1991) PARA TRABALHAR COM
arbitrariamente
C GAMA grande custo (até ao intervalo inteiro da máquina);
C Para conseguir isso, os preços de objeto e EPSILON SÃO
VARIÁVEIS C precisão dupla, para que os preços NÃO PODE inundarão o
INTEIRO DO GAMA C MACHINE.
C
C ********************************************** ****************

        PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

        NONE IMPLICIT
        INTEGER N, NA, A, ISMALL, CICLOS
        INTEGER NUMPHASES
        INTEGER I, J, IA, ARC, NOASS, ICOST, ABSCOST, CURARC
        INTEGER CURCOL, FSTARC, LSTARC, MINCOST, MAXCOST, MCOST
        INTEGER FOUT (maxNodes), COLASS (maxNodes)
        INTEGER ASSIGN (maxNodes)
        INTEGER COST (MAXARCS), END (MAXARCS)
        Real * 8 TT1, TT2, TCOST
        Real * 8 PCOL (maxNodes)
        Real * 8 MÉDIA
        Real * 8 BEGEPS, ENDEPS, FACTOR, STARTINCR, TEST, o lucro
        COMUM / ARRAYC / CUSTO / matrizes / END / ARRAYF / FOUT / BK1
/ N, A, ISMALL
     $ / BK2 / ciclos, Médio, NUMPHASES / ARRAYA / ASSIGN / PCOLA /
PCOL

C ********************************************** ***************
C
C problema de geração de código começa AQUI
C o usuário pode substituir este código com um código que lê
C HIS / seu problema de um arquivo
C
C ********************************************** ***************

C Este código inclui UM UNIFORME Random Number Generator
C que retorna um valor na (0,1)

C INITIALIZE Gerador Aleatório

      INTEGER MULT, MODUL, I15, I16, JRAN, ISEED
      RAN REAIS
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
```

```
      ISEED = 13502460
      CHAME SETRAN (ISEED)

      PRINT * ', gerando problema de atribuição de SIMÉTRICA "
      PRINT *, "****************************************** '

C **** LER O número de linhas N & o número de arcos A ****

      PRINT *, 'Digite o número de linhas (e colunas)'
      LEIA *, N
5 PRINT *, 'Digite o número de arcos por ROW (1>) "
      LEIA *, NA
      IF (NA.LT.2) GOTO 5
      PRINT *, 'Digite o mínimo eo custo máximo'
      LEIA *, MINCOST, MAXCOST

C o número de arcos é n * NA

      A = N * NA

C Os arcos incidente a linha i SÃO FOUT (I) PARA FOUT (I + 1) -1
C Além disso, para VIABILIDADE EACH ROW está diretamente ligado
C com a coluna correspondente

      FAZER 20 I = 1, N
        FOUT (I) = 1 + (i-1) * NA
20 CONTINUAR
      FOUT (N + 1) = A + 1

C gerar a END (ARC) eo custo (ARC), que são os COLUNA
C eo coeficiente custo associado a ARC

      FAZER 25 ARC = 1, A
        END (ARC) = 1 + RAN () * N
        IF ((END (ARC) .GT.N) .OR. (END (ARC) .LT.1)) THEN
          PRINT *, 'Erro no problema de geração de'
          PAUSA
          PARE
        END IF
        COST (ARC) = MINCOST + RAN () * (MAXCOST-MINCOST)
25 CONTINUAR

C MODIFICAR O final da última ARC OUT de cada linha de viabilidade
C e defina seu CUSTO PARA -MAXCOST

      FAZER 30 I = 1, N
        END (FOUT (I + 1 -1)) = I
        CUSTO (FOUT (I + 1) -1) = - MAXCOST
30 CONTINUAR


C
C ********************************************** **************
C
C PROBLEMA geração de código TERMINA AQUI
C
C ********************************************** **************

C ESCALA O CUSTO PARA TRABALHAR COM INTEIRO EPSILON

        MAXCOST = 0
```

```
        FAZER 35 IA = 1, A
           ABSCOST = IABS (COST (IA))
           IF (ABSCOST.GT.MAXCOST) MAXCOST = ABSCOST
35 CONTINUAR

C *** ISMALL é um inteiro MUITO PEQUENO PARA O SEU APARELHO ***

        ISMALL = -2000000000

        IF (MAXCOST.GT.ABS (ISMALL)) THEN
          PRINT *, 'A faixa de custo é muito grande para INTEGER
ARITHMETIC'
          PAUSA
          PARE
        END IF


C os seguintes parâmetros BEGEPS, FACTOR, ENDEPS E STARTINCR
C são passados para o LEILÃO algoritmo. Valores entre
C (A) MAXCOST / 5 E MAXCOST / 2 para BEGEPS
C (B) 4 e 6 para FACTOR
C (C) 1/10 e 1 / (N + 1) PARA ENDEPS
C (D) 1 AND BEGEPS PARA STARTINCR
C têm funcionado bem para PROBLEMAS esparsos de grande porte.
C PARA PROBLEMAS DENSOS RECOMENDA-SE QUE
C BEGEPS ser ajustado para um valor menor,
ENDEPS C ser definido como 1 / (N + 1),
C STARTINCR ser definido como 1 / (N + 1).

        PRINT *, "****************************************** '
        PRINT *, "custo máximo é ', MAXCOST
        PRINT *, 'Digite o EPSILON PARTIDA'
        LER *, BEGEPS
        IF (BEGEPS.LT.1.0 / (N + 1)) BEGEPS = 1,0 / (N + 1)
        PRINT *, 'Insira o fator EPSILON REDUÇÃO'
        LER *, FACTOR
        ENDEPS = 1,0 / 10,0
        IF (ENDEPS.LT.1.0 / (N + 1)) ENDEPS = 1,0 / (N + 1)
        IF (ENDEPS.GT.BEGEPS) ENDEPS = BEGEPS
        STARTINCR = BEGEPS / 10.0
        IF (STARTINCR.LT.1.0 / (N + 1)) STARTINCR = 1,0 / (N + 1)

        PRINT *, "****************************************** '
        PRINT *, 'começar EPSILON =', BEGEPS
        PRINT *, 'Epsilon redução de fatores =', FACTOR
        PRINT *, 'Epsilon' Limite 'antes ela é definida como 1 / (N +
1) =', ENDEPS
        PRINT *, "COMEÇAR MIN DE LICITAÇÃO INCREMENTO = ', STARTINCR
        PRINT *, "CHAMANDO LEILÃO para resolver o problema"
        PRINT *, "****************************************** '

C GET hora de início para o Mac II

      TT1 = longa (362) /60.0

      CHAMADA Leilão (BEGEPS, fator, ENDEPS, STARTINCR)

C GET TÉRMINO TEMPO PARA O MAC II

      TT2 = longa (362) /60.0 - TT1
      PRINT *, "acabado --- tempo de CPU TOTAL ', TT2,' s '
```

```
      PRINT *, "****************************************** '

C *** Mostrar resultados ***

X Write (9,2010) CICLOS
X2010 FORMATO ('NO LEILÃO DE CICLOS', I7)
X WRITE (9,2020) MÉDIA
X2020 FORMATO ("número médio de propostas por ciclo", F9.3)
       Write (9,2030) NUMPHASES
2030 FORMATO ('NO DE EPSILON subproblemas RESOLVIDO =', I7)

C verificação de viabilidade de uma solução e calcular o custo

        DO 40 I = 1, N
          COLASS (I) = 0
40 CONTINUAR
        NOASS = 0
        FAZER 50 J = 1, N
           IF (ASSIGN (J) .GT.0) THEN
             NOASS = + 1 NOASS
             COLASS (ASSIGN (J)) = J
           END IF
50 CONTINUAR
        IF (NOASS.NE.N) THEN
          PRINT *, '# de linhas atribuído não IGUAL # de linhas'
        END IF
        TCOST = 0
        FAZER 60 I = 1, N
          J = COLASS (I)
          IF (J.EQ.0) THEN
            PRINT *, 'ROW', I, 'não é atribuído "
          END IF
          FSTARC = FOUT (I)
          LSTARC FOUT = (I + 1) -1
          MCOST = ISMALL
          FAZER 70 CURARC = FSTARC, LSTARC
            CURCOL = END (CURARC)
            IF (CURCOL.EQ.J) THEN
              IF (MCOST.LT.COST (CURARC)) THEN
                 MCOST = COST (CURARC)
              END IF
            END IF
70 CONTINUAR
          LUCRO = MCOST-PCOL (J)
          FAZER 72 CURARC = FSTARC, LSTARC
             J = END (CURARC)
             TEST = COST (CURARC) -PCOL (J) lucrativos
             IF (TEST.GT.1.0 / N) THEN
                PRINT *, '1 / (N + 1) -cs VIOLADOS AT ARC', CURARC
             END IF
72 CONTINUAR
          TCOST = TCOST + MCOST
60 CONTINUAR
        ESCREVA (9,2100) TCOST
2100 FORMATO ('atribuição de custo =', F22.2)
        PRINT *, 'programa terminou; <CR> PARA SAIR '
        PAUSA
        FIM
```

```
C ********************************************** ****************
C
C ponto flutuante LEILÃO CÓDIGO DE N por N problemas de atribuição
C
C ESCRITO POR DIMITRI P. Bertsekas
C (MODIFICAÇÕES por David Castanon E PAULO TSENG)
C
C versão 1.2, outubro 1991
C
C
C MODIFICADO POR DAVID Castanon (OCT. 1991) PARA TRABALHAR COM
arbitrariamente
C GAMA grande custo (até ao intervalo inteiro da máquina);
C Para conseguir isso, os preços de objeto e EPSILON SÃO
VARIÁVEIS C precisão dupla, para que os preços NÃO PODE inundarão o
INTEIRO DO GAMA C MACHINE.
C
C ESTE CÓDIGO implementa o algoritmo LEILÃO COM E-escala.
C Ele resolve uma seqüência de subproblemas e diminui
C EPSILON por um fator constante entre subproblemas.
C ESTE versão corresponda a um MODO Gauss-Seidel
C e resolve EPSILON subproblemas inexata.
C
C THE CODE é uma versão melhorada de um antigo (SEPT. 1985)
C CODE LEILÃO COM ESCALA E-ESCRITO POR DIMITRI P. Bertsekas
C
C DO CÓDIGO trata o problema como um problema de maximização.
C para resolver um problema de minimização, inverta o sinal da
C ARC CUSTOS antes de chamar LEILÃO, E RECUO DE NOVO
C O SINAL DO custo ótimo na volta do LEILÃO.
C Este código permite ARCS múltipla entre uma linha e uma coluna.
C
C Esta versão do ALGORITHM LEILÃO é adaptativa. AQUI NO MÍNIMO
INCREMENTO DE LICITAÇÃO C (armazenado no INCR variável) pode ser menor
que
C EPSILON. PARA CADA subproblema, INCR COMEÇA no valor do parâmetro
C STARTINCR (que é passado para a rotina LEILÃO), estando aumentada
C por um fator de 2 no final de cada ciclo, até um valor máximo de
C EPSILON. ESTA FUNÇÃO adaptativo é particularmente eficaz para
C PROBLEMAS densa. TI pode ser derrotado por SELEÇÃO STARTINCR =
BEGEPS
C
C **********************************************
******************
C
C o usuário deve fornecer os seguintes dados problema no FRENTE DA
ESTRELA FORMATO
C (ou seja, todos os arcos da linha SAME são numerados
consecutivamente):
CN = número de linhas (igual número de colunas)
CA = número de arcos
C FOUT (ROW) = Primeiro ARC QUE SAI DE ROW
C COST (ARC) = CUSTO DE ARC
C END (ARC) = COLUNA correspondente ao arco
C
C E os seguintes parâmetros para o algoritmo LEILÃO:
C BEGEPS = Valor inicial EPSILON (não deve ser menor do que 1 / (N +
1))
C ENDEPS = valor final da EPSILON ANTES ela é definida como 1 / (N +
1)
C FACTOR = fator pelo qual EPSILON é diminuída ENTRE subproblemas
```

```
C STARTINCR = O valor inicial do incremento LICITAÇÃO
ENDEPS C não devem exceder BEGEPS.
C fator deve ser maior do que 1.
C
C FOUT (.) É uma matriz de comprimento N.
C COST (.), END (.) São matrizes de COMPRIMENTO A.
C
C, a solução está contida no ASSIGN Array (.) ONDE
C ASSIGN (COL) Dá o ROW ATRIBUÍDO COL.
C
C Este algoritmo não verifica a inviabilidade do problema.
C TO garantir que o problema é viável o usuário pode ADD
Adicional C ARCS custo muito pequeno.
C
C ********************************************** *****************
C ALL problema de dados são integer
C ********************************************** *****************


        SUBROUTINE Leilão (BEGEPS, fator, ENDEPS, STARTINCR)

        PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

        NONE IMPLICIT
        INTEGER NONEWLIST, THRESH
        INTEIRO A, K, N, I, J, CURARC, CURCOL
        ROW INTEIRO, FSTARC, FSTCOL, SNDARC, SNDCOL, BSTCOL
        INTEGER TRDARC
        INTEGER M, ISMALL, ILARGE, LARGEINCR, CICLOS
        INTEGER NUMPHASES, LSTARC, NOLIST, OLDROW
        INTEGER FOUT (maxNodes), LIST (maxNodes)
        INTEGER ASSIGN (maxNodes)
        INTEGER COST (MAXARCS), END (MAXARCS)
        Real * 8 BEGEPS, ENDEPS, FACTOR, STARTINCR, INCRFACTOR
        Real * 8 PCOL (maxNodes)
        Real * 8 Max1, MAX2, TMAX, EPSILON, INCR
        Real * 8 MÉDIA
        COMUM / ARRAYC / CUSTO / matrizes / END / ARRAYF / FOUT / BK1
/ N, A, ISMALL
     $ / BK2 / ciclos, Médio, NUMPHASES / ARRAYA / ASSIGN / PCOLA /
PCOL

C ********************************************** **************

Check C ******* VALIDADE DOS PARÂMETROS ******* PASSOU

        IF (BEGEPS.LE.1.0 / (N + 2)) THEN
          PRINT *, 'Valor inicial EPSILON é inferior a 1 / (N + 1)'
          PRINT *, 'EXECUÇÃO ABORTADO'
          PARE
        END IF
        IF (ENDEPS.GT.BEGEPS) THEN
          PRINT *, 'ENDEPS parâmetro é maior que o parâmetro BEGEPS'
          PRINT *, 'ENDEPS é definido no valor padrão de 1 / (N + 1)'
          ENDEPS = 1,0 / (N + 1)
        END IF
        IF ((FACTOR.LE.1) .E. (BEGEPS.GE.1.0 / N)) THEN
          PRINT *, 'Epsilon redução de fatores não é maior que 1'
          PRINT *, 'EXECUÇÃO ABORTADO'
          PARE
        END IF
```

```
        IF (STARTINCR.LE.1.0 / (N + 2)) THEN
          PRINT *, 'MIN DE LICITAÇÃO incremento é menor que 1 / (N +
1)'
          PRINT *, 'STARTINCR é definido no valor padrão de 1 / (N +
1)'
          STARTINCR = 1,0 / (N + 1)
        END IF

C ******* INITIALIZATION *******

        EPSILON = BEGEPS
        ILARGE = -ISMALL
        LARGEINCR = INT (ILARGE / 10)
        THRESH = INT (0,2 * N)
        INCRFACTOR = 2,0
        IF (THRESH.GT.100) THRESH = 100
CICLOS x = 1
X MÉDIA = N
         NUMPHASES = 1
         FAZER 10 J = 1, N
           ASSIGN (J) = 0
           PCOL (J) = ISMALL
10 CONTINUAR

        FOUT (N + 1) = A + 1
        NOLIST = N
        FAZER 20 I = 1, N
            LISTA (I) = I
20 CONTINUAR

C ********************************************** ***************
C
C Esta implementação do leilão ALGORITHM opera em ciclos.
C Cada ciclo consiste em um lance em cada uma das linhas QUE SEJAM
C não atribuído NO INÍCIO DO CICLO (essas linhas são armazenadas em
C lista de matriz (.)). AS que o ciclo progride NOVO
LINHAS C ficarem não atribuídas; Estes são armazenados em Lista (.)
C e apresentará um lance no próximo ciclo.
C NOTA que apenas uma linha apresenta uma proposta de cada vez; Isto é
conhecido como
C o método de Gauss-Seidel versão do algoritmo. A VERSÃO onde todos
C ROWS não atribuído apresentar uma proposta AO MESMO TEMPO, é
conhecido como de Jacobi
C versão do algoritmo, e não foi implementada. Geralmente
C tende a correr um pouco mais lento do que a versão de Gauss-Seidel,
MAS
C admite um grau mais elevado DE paralelização. A VERSÃO JACOBI
C pode ser preferível ON Uma máquina paralela, mas geralmente é
INFERIOR
C PARA A VERSÃO Gauss-Seidel EM UMA MÁQUINA DE SÉRIE.
C
C ********************************************** ***************
C
C INÍCIO subproblema (FASE DE ESCALA) W / NEW EPSILON
C
C ********************************************** ***************

12 CONTINUAR

        IF (EPSILON.LE.1.0 / (N + 1)) debulhar = 0
        INCR = STARTINCR
```

```
        IF (INCR.GT.EPSILON) INCR = EPSILON

C ********************************************** **************
C
C início do ciclo LEILÃO COM LISTA NOVO
C
C ********************************************** **************

15 CONTINUAR

C REINICIAR CONTAGEM DO PRÓXIMO lista de linhas não atribuídos

        NONEWLIST = 0

C percorrer a lista atual de linhas não atribuídos

        FAZER 100 I = 1, NOLIST
        ROW = LIST (I)
        FSTARC = FOUT (ROW)
        LSTARC = FOUT (ROW + 1) -1
        FSTCOL = END (FSTARC)

C PRIMEIRA CUIDAR DA caso excepcional ROW tem apenas um ARC

        IF (FSTARC.EQ.LSTARC) THEN
          PCOL (FSTCOL) = PCOL (FSTCOL) + LARGEINCR
          OLDROW = ASSIGN (FSTCOL)
          ASSIGN (FSTCOL) = ROW
          IF (OLDROW.GT.0) THEN
            NONEWLIST = + 1 NONEWLIST
            LIST (NONEWLIST) = OLDROW
          END IF
          Ir para 100
        END IF

C tomar em seguida CUIDADOS DO CASO regular, onde linha tem ARCS
MÚLTIPLAS

        SNDARC = + 1 FSTARC
        SNDCOL = END (SNDARC)
        Max1 = COST (FSTARC) -PCOL (FSTCOL)
        MAX2 = COST (SNDARC) -PCOL (SNDCOL)
        IF (MAX1.GE.MAX2) THEN
           BSTCOL = FSTCOL
        MAIS
           TMAX = Max1
           Max1 = MAX2
           MAX2 = TMAX
           BSTCOL = SNDCOL
        END IF
        IF (SNDARC.LT.LSTARC) THEN
        TRDARC = + 1 SNDARC
        DO 40 CURARC = TRDARC, LSTARC
           CURCOL = END (CURARC)
           TMAX = COST (CURARC) -PCOL (CURCOL)
           IF (TMAX.GT.MAX2) THEN
             IF (TMAX.GT.MAX1) THEN
                  MAX2 = Max1
                  Max1 = TMAX
                  BSTCOL = CURCOL
             MAIS
```

```
                MAX2 = TMAX
              END IF
            END IF
40 CONTINUAR
        END IF


APOSTAS C linha para BSTCOL AUMENTAR SEU PREÇO, E é atribuído
C TO BSTCOL, enquanto qualquer ROW ATRIBUÍDO BSTCOL TORNA não
atribuídos

          PCOL (BSTCOL) = PCOL (BSTCOL) + Max1-MAX2 + INCR
          OLDROW = ASSIGN (BSTCOL)
          ASSIGN (BSTCOL) = ROW
          IF (OLDROW.GT.0) THEN
            NONEWLIST = + 1 NONEWLIST
            LIST (NONEWLIST) = OLDROW
          END IF

100 CONTINUAR

C ************* FIM DE UM CICLO DE LEILÃO ***************

C OPTIONALLY coletar estatísticas

X Médio = (ciclos * MÉDIA + NOLIST) / (CICLOS + 1)
CICLOS X = CICLOS + 1

C Verifique se ainda existem ROWS não atribuído 'muitos', isto é, se o
C número de linhas não atribuído É MAIOR DO QUE
C DO THRESH PARAMETER. NÃO SE, REPLACE lista atual com a lista NEW,
C e ir para outro ciclo. Caso contrário, se EPSILON> 1, REDUZIR
EPSILON,
C Redefinir a atribuição ao vazio e REINICIAR LEILÃO;
C IF EPSILON = 1 finalizar.
C também aumentam a LICITAÇÃO MINIMAL INCREMENTO até um máximo
C VALOR DO EPSILON (esta é a característica adaptativa)

         INCR = INCR * INCRFACTOR
         IF (INCR.GT.EPSILON) INCR = EPSILON
         IF (NONEWLIST.GT.THRESH) THEN
             NOLIST = NONEWLIST
             IR PARA 15
         END IF

C ********************************************** **************
C
C FIM DE subproblema (escalonamento FASE)
C
C ********************************************** **************

C ****** SE EPSILON É uma RESCINDIR ******

        IF (EPSILON.LE.1.0 / N) THEN
           RETURN
        MAIS

C MAIS REDUZIR EPSILON e redefinir o CESSÃO ESVAZIAR

           NUMPHASES + 1 = NUMPHASES
           EPSILON = EPSILON / FACTOR
```

```
          IF (EPSILON.GT.INCR) EPSILON = EPSILON / FACTOR
          IF ((EPSILON.LT.1.0 / N) .OR. (EPSILON.LT.ENDEPS)) THEN
             EPSILON = 1,0 / (N + 1)
          END IF
          THRESH = INT (THRESH / FACTOR)

X PRINT *, '*** FIM DE UMA FASE DE ESCALA; NEW EPSILON = ', EPSILON

          FAZER 200 J = 1, N
           IF (ASSIGN (J) .GT.0) THEN
            NONEWLIST = + 1 NONEWLIST
            LIST (NONEWLIST) = ASSIGN (J)
            ASSIGN (J) = 0
           END IF
200 CONTINUAR



C FINAIS PARÂMETROS atualizações antes de voltar para outra ESCALA
FASE

          NOLIST = NONEWLIST
          IF (STARTINCR.LT.EPSILON) STARTINCR = FACTOR * STARTINCR
          IR PARA 12
        END IF


        FIM



      SUBROUTINE SETRAN (ISEED)
      IMPLICIT real * 8 (AH, OZ), Integer * 4 (IN)
C ***********************************************
**********************
C PORTABLE congruential (uniforme) RANDOM gerador de números:
NEXT_VALUE C = [(7 ** 5) * PREVIOUS_VALUE] MODULO [(2 ** 31) -1]
C
C Este gerador é composto por duas ROTINAS:
C (1) SETRAN - inicializa constantes e SEED
C (2) RRAN - gera um número aleatório REAIS
C
C O GERADOR DE RODA uma máquina com pelo menos 32 bits de precisão.
C DA SEED (ISEED) deve estar no intervalo (1, (2 ** 31 -1)).
C ***********************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      IF (ISEED.LT.1) PARADA 77
      MULT = 16807
      MODUL = 2147483647
      I15 = 2 ** 15
      I16 = 2 ** 16
      JRAN = ISEED
      RETURN
      FIM

      RAN função real ()
      IMPLICIT real * 4 (AH, OZ), Integer * 4 (IN)
C ***********************************************
**********************
C RAN gera um número aleatório real entre 0 e 1
```

```
C ***********************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      Ixhi = JRAN / I16
      IXLO = JRAN-Ixhi * I16
      IXALO = IXLO * MULT
      LEFTLO = IXALO / I16
      IXAHI = Ixhi * MULT
      IFULHI = IXAHI + LEFTLO
      IRTLO = IXALO-LEFTLO * I16
      IOVER = IFULHI / I15
      IRTHI = IFULHI-IOVER * I15
      JRAN = ((IRTLO-MODUL) + IRTHI * I16) + IOVER
      IF (JRAN.LT.0) JRAN = JRAN + MODUL
      RAN = FLOAT (JRAN) / FLOAT (MODUL)
      RETURN
      FIM
```

LEILÃO-AS

Este código implementa o leilão para a frente / verso
algoritmo com $ \ e $ scaling para o problema de atribuição
assimétrica; cf. \
Seção 4.2.1. Ele também pode resolver como um caso especial da
atribuição simétrica
problema. Neste caso, bem como no caso em que $ \ $ e scaling é
ignorada
($ \ E = 1 $ ao longo do algoritmo), apenas a parte da frente do
algoritmo é
utilizado. Note também que este código usa o `` terceiro melhor ''
implementação de objeto
descrito no Exercício 1.7 da Seção 4.1.

```
C ********************************************** *****************
C
C programa de amostra CHAMADA PARA LEILÃO ALGORITHM
C para o problema de atribuição ASYMMETRIC
C
C ESTE MOTORISTA cria um problema de atribuição ASYMMETRIC
C COM menor ou igual número de linhas que as colunas,
C e chama o SUBROUTINE AUCTION_AS para encontrar um
C atribuição do valor máximo.
C
C esta versão usa um terceiro melhor VALOR
C
C ********************************************** *****************

        PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

        NONE IMPLICIT
        INTEGER M, N, NA, A, IA, K, ILARGE, BEGEPS, ENDEPS, CICLOS
        INTEGER NUMPHASES, STARTINCR, LAMBDA
        INTEGER I, J, ARC, NOASS, ICOST, ABSCOST, CURARC
        INTEGER CURCOL, FSTARC, LSTARC, MINCOST, MAXCOST, MCOST
        INTEGER EXTRA, o restante, INDEX, COUNT
        INTEGER COL_ASSIGNED_TO (maxNodes), ROW_ASSIGNED_TO (maxNodes)
        INTEGER FOUT (maxNodes), FIN (maxNodes), NXTIN (MAXARCS)
```

```
        CUSTO INTEIRO (MAXARCS), START (MAXARCS), END (MAXARCS)
        INTEGER PCOL (maxNodes), PROW (maxNodes)
        INTEGER PRDARC (maxNodes)
        Real * 8 FACTOR, TT1, TT2, TCOST, AVERAGE
        COMUM / ARRAYC / CUSTO / matrizes / START / ARRAYE / END
        COMUM / ARRAYFO / FOUT / ARRAYFI / FIN / ARRAYNI / NXTIN
        COMUM / ARRAYRA / ROW_ASSIGNED_TO / ARRAYCA / COL_ASSIGNED_TO
        COMUM / ARRAYPC / PCOL / ARRAYPR / PROW
        COMUM / BK1 / M, N, A, ILARGE
        Common / BK2 / ciclos, Médio, NUMPHASES, LAMBDA

C ********************************************** **************
C
C problema de geração de código começa AQUI
C o usuário pode substituir este código com um código que lê
C HIS / seu problema de um arquivo
C
C ********************************************** **************

C Este código inclui UM UNIFORME Random Number Generator
C que retorna um valor na (0,1)

C INITIALIZE Gerador Aleatório

      INTEGER MULT, MODUL, I15, I16, JRAN, ISEED
      RAN REAIS
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN

      ISEED = 13502460
      CHAME SETRAN (ISEED)

      PRINT *, "gerando um problema de atribuição ASYMMETRIC '
      PRINT *, "****************************************** '

C **** LER O número de linhas M, N colunas e arcos A ****

      PRINT *, 'Digite o número de linhas'
      LEIA *, M

2 PRINT *, 'Digite o número de colunas (> = # de linhas)'
      LEIA *, N
      IF (N.LT.M) ir para 2

5 PRINT *, 'Digite o número de arcos por ROW (1>) "
      LEIA *, NA
      IF (NA.LT.2) GOTO 5

      PRINT *, 'Digite o mínimo eo custo máximo'
      LEIA *, MINCOST, MAXCOST

C o número de arcos é M * NA + NM

      A = H * + NA NM
      PRINT *, "o número de arcos IS ', A

C Os arcos incidente a linha i SÃO FOUT (I) PARA FOUT (I + 1) -1
C Além disso, para VIABILIDADE EACH ROW está diretamente ligado
C com a coluna correspondente;
C TAMBÉM cada coluna tem pelo menos um ARC

      EXTRA = INT ((NM) / M)
```

```
      FAZER 20 I = 1, M
        FOUT (I) = 1 + (i-1) * (EXTRA NA +)
20 CONTINUAR
      FOUT (M + 1) = A + 1

      FAZER 22 I = 1, M
        FAZER 23 ARC = FOUT (I), FOUT (I + 1) -1
          START (ARC) = I
23 CONTINUAR
22 CONTINUAR

C gerar a END (ARC) eo custo (ARC), que são os COLUNA
C eo coeficiente custo associado a ARC

      FAZER 25 ARC = 1, A
        END (ARC) = 1 + RAN () * N
        IF ((END (ARC) .GT.N) .OR. (END (ARC) .LT.1)) THEN
          PRINT *, 'Erro no problema de geração de'
          PAUSA
          PARE
        END IF
        COST (ARC) = MINCOST + RAN () * (MAXCOST-MINCOST)
25 CONTINUAR

C MODIFICAR O final da última ARC OUT de cada linha de viabilidade
C e defina seu CUSTO PARA -MAXCOST

      FAZER 30 I = 1, M
        END (FOUT (I + 1 -1)) = I
        CUSTO (FOUT (I + 1) -1) = - MAXCOST
30 CONTINUAR

C MODIFICAR final de algumas ARCS modo que cada coluna tem pelo menos
um ARC

      IF (EXTRA.GE.1) THEN
        FAZER 32 I = 1, M
          FAZER 33 K = 1, EXTRA
            END (FOUT (I) + K-1) = K * M + I
33 CONTINUAR
32 CONTINUAR
       END IF

       REMAINDER = NM * (1 + EXTRA)
       IF (REMAINDER.GE.1) THEN
         INDEX = FOUT (M) + EXTRA-1
         FAZER 35 J = 1, REMAINDER
           END (INDEX + J) = (1 + EXTRA) * M + J
35 CONTINUAR
       END IF

C construir a FIN () e NXTIN () ARRAYS

      DO 42 J = 1, N
        FIN (J) = 0
        PRDARC (J) = 0
42 CONTINUAR

      FAZER 43 ARC = 1, A
        NXTIN (ARC) = 0
```

```
        J = END (ARC)
        IF (FIN (J) .NE.0) THEN
          NXTIN (PRDARC (J)) = ARC
        MAIS
          FIN (J) = ARC
        END IF
        PRDARC (J) = ARC
43 CONTINUAR


C ********************************************** **************
C
C PROBLEMA geração de código TERMINA AQUI
C
C ********************************************** **************



C ESCALA O CUSTO PARA TRABALHAR COM INTEIRO EPSILON

        MAXCOST = 0
        FAZER 45 IA = 1, A
           ABSCOST = IABS (COST (IA))
           IF (ABSCOST.GT.MAXCOST) MAXCOST = ABSCOST
           COST (IA) = CUSTO (IA) * (M + 1)
45 CONTINUAR

C *** ILARGE é um inteiro MUITO GRANDE PARA SUA MÁQUINA ***

        ILARGE = 2000000000

        IF (MAXCOST.GT.INT (ILARGE / (M + 1))) THEN
          PRINT *, 'A faixa de custo é muito grande para INTEGER
ARITHMETIC'
          PAUSA
          PARE
        END IF

        MAXCOST MAXCOST * = (M + 1)

        LAMBDA = -ILARGE

C os seguintes parâmetros BEGEPS, FACTOR, ENDEPS E STARTINCR
C são passados para o LEILÃO algoritmo. Valores entre
C (A) MAXCOST / 5 E MAXCOST / 2 para BEGEPS
C (B) 4 e 6 para FACTOR
C (C) M / 10 e 1 para ENDEPS
C (D) 1 AND BEGEPS PARA STARTINCR
C têm funcionado bem para PROBLEMAS esparsos de grande porte.
C PARA PROBLEMAS PARA densa e problemas muito ASSIMÉTRICAS
C RECOMENDA-SE QUE
C BEGEPS ser ajustado para um valor menor (possivelmente 1),
ENDEPS C ser definido como 1,
C STARTINCR ser definido como 1.

C PARA problemas muito assimétrico, BEGEPS = 1, correspondendo a
C NO E-dimensionamento, pode funcionar melhor e deve ser pelo menos
tentei.

        PRINT *, "custo máximo é ', MAXCOST
        PRINT *, 'Digite o EPSILON PARTIDA'
```

```
        LER *, BEGEPS
        IF (BEGEPS.LT.1) BEGEPS = 1
        PRINT *, 'Insira o fator EPSILON REDUÇÃO'
        LER *, FACTOR
        ENDEPS = H / 10
        IF (ENDEPS.LT.1) ENDEPS = 1
        IF (ENDEPS.GT.BEGEPS) ENDEPS = BEGEPS
        STARTINCR = BEGEPS / 10
        IF (STARTINCR.LT.1) STARTINCR = 1

        PRINT *, "****************************************** '
        PRINT *, 'começar EPSILON =', BEGEPS
        PRINT *, 'Epsilon redução de fatores =', FACTOR
        PRINT *, «limiar EPSILON ANTES ela é definida como 1 = ',
ENDEPS
        PRINT *, "COMEÇAR MIN DE LICITAÇÃO INCREMENTO = ', STARTINCR
        PRINT *, "CHAMANDO ASYMMETRIC LEILÃO CESSÃO '
        PRINT *, "****************************************** '

C GET hora de início para o Mac II

      TT1 = longa (362) /60.0

      CHAMADA AUCTION_AS (BEGEPS, fator, ENDEPS, STARTINCR)

C GET TÉRMINO TEMPO PARA O MAC II

      TT2 = longa (362) /60.0 - TT1
      PRINT *, "acabado --- tempo de CPU TOTAL ', TT2,' s '
      PRINT *, "****************************************** '

C *** Mostrar resultados ***

X Write (9,2010) CICLOS
X2010 FORMATO ('NO LEILÃO DE CICLOS', I7)
X WRITE (9,2020) MÉDIA
X2020 FORMATO ("número médio de propostas por ciclo", F9.3)
       Write (9,2030) NUMPHASES
2030 FORMATO ('NO DE EPSILON subproblemas RESOLVIDO =', I7)

C VERIFICAÇÃO Optimality & calcular o custo

        FAZER 48 J = 1, N
          I = ROW_ASSIGNED_TO (J)
          IF (N.GT.M) THEN
            ROW_ASSIGNED_TO (J) = 0
          MAIS
            IF (I.GT.0) THEN
              COL_ASSIGNED_TO (I) = J
            END IF
          END IF
48 CONTINUAR


        TCOST = 0
        FAZER 50 I = 1, M
          J = COL_ASSIGNED_TO (I)
          ROW_ASSIGNED_TO (J) = I
          IF (J.EQ.0) THEN
             PRINT *, 'ROW', I, 'não é atribuído "
          END IF
```

```
          IF (PCOL (J) .LT.LAMBDA) THEN
             PRINT *, 'CS VIOLAÇÃO EM COLUNA afectadas », J
          END IF
          FSTARC = FOUT (I)
          LSTARC FOUT = (I + 1) -1
          MCOST = -ILARGE
          FAZER 55 ARC = FSTARC, LSTARC
            CURCOL = END (ARC)
            IF (PROW (I) + PCOL (CURCOL) .LT.COST (ARC -1)) THEN
              PRINT *, '1-CS VIOLADOS AT ARC ", ARC
            END IF
            IF (CURCOL.EQ.J) THEN
              IF (MCOST.LT.COST (ARC)) THEN
                 MCOST = CUSTO (ARC)
              END IF
            END IF
55 CONTINUAR
          TCOST = TCOST MCOST + / (M + 1)
          IF (PROW (I) + PCOL (J) .NE.MCOST) THEN
            PRINT *, '1-CS VIOLADA na linha', I
          END IF
50 CONTINUAR

        COUNT = 0
        DO 60 J = 1, N
           IF (ROW_ASSIGNED_TO (J) .EQ.0) THEN
             IF (PCOL (J) .GT.LAMBDA) THEN
               PRINT *, 'CS VIOLAÇÃO EM COLUNA não atribuído', J
             END IF
           MAIS
             COUNT = COUNT + 1
           END IF
60 CONTINUAR

        IF (COUNT.LT.M) THEN
          PRINT *, "o número de colunas atribuído é errado"
        END IF


        ESCREVA (9,2100) TCOST
2100 FORMATO ('atribuição de custo =', F18.2)
        PRINT *, 'programa terminou; <CR> PARA SAIR '
        PAUSA
        FIM




C ********************************************** *****************
C
C CODE LEILÃO PARA ASYMMETRIC M por N problemas de atribuição
C
C ESCRITO POR DIMITRI P. Bertsekas
C
C dezembro 1990
C
C ESTE CÓDIGO implementa um avanço / retrocesso ALGORITHM LEILÃO
C COM E-escala para o problema de atribuição assimétrica.
C Ele resolve uma seqüência de subproblemas e diminui
C EPSILON por um fator constante entre subproblemas.
C ESTE versão corresponda a um MODO Gauss-Seidel.
```

```
C
C DO CÓDIGO trata o problema como um problema de maximização.
C para resolver um problema de minimização, inverta o sinal da
C ARC CUSTOS antes de chamar LEILÃO, E RECUO DE NOVO
C O SINAL DO custo ótimo na volta do LEILÃO.
C Este código permite ARCS múltipla entre uma linha e uma coluna.
C
C Esta versão do ALGORITHM LEILÃO é adaptativa. AQUI NO MÍNIMO
INCREMENTO DE LICITAÇÃO C (armazenado no INCR variável) pode ser menor
que
C EPSILON. PARA CADA subproblema, INCR COMEÇA no valor do parâmetro
C STARTINCR (que é passado para a rotina LEILÃO), estando aumentada
C por um fator de 2 no final de cada ciclo, até um valor máximo de
C EPSILON. ESTA FUNÇÃO adaptativo é particularmente eficaz para
C PROBLEMAS densa. TI pode ser derrotado por SELEÇÃO STARTINCR =
BEGEPS
C
C ************************************************
******************
C
C o usuário deve fornecer os seguintes dados problema no FRENTE DA
ESTRELA FORMATO
C (ou seja, todos os arcos da linha SAME são numerados
consecutivamente):
CM = número de linhas
CN = número de colunas
CA = número de arcos
C FOUT (ROW) = Primeiro ARC QUE SAI DE ROW
C FIN (COL) = PRIMEIRO ARC VINDO EM COL
C NXTIN (ARC) = Próxima ARC incidente à COLUNA MESMO AS ARC
C COST (ARC) = CUSTO DE ARC
C START (ARC) = ROW correspondente ao arco
C END (ARC) = COLUNA correspondente ao arco
C
C E os seguintes parâmetros para o algoritmo LEILÃO:
C BEGEPS = Valor inicial EPSILON (não deve ser menor que 1)
C ENDEPS = valor final da EPSILON ANTES ela é definida como 1
C FACTOR = fator pelo qual EPSILON é diminuída ENTRE subproblemas
C STARTINCR = O valor inicial do incremento LICITAÇÃO
ENDEPS C não devem exceder BEGEPS.
C fator deve ser maior do que 1.
C
C FOUT (.) É uma matriz de comprimento N.
C COST (.), END (.) São matrizes de COMPRIMENTO A.
C
C a solução é contidos na matriz ROW_ASSIGNED_TO (.) ONDE
C ROW_ASSIGNED_TO (COL) Dá o ROW ATRIBUÍDO COL.
C TAMBÉM COL_ASSIGNED_TO (ROW) DÁ A COLUNA ATRIBUÍDO linha.
C
C Este algoritmo não verifica a inviabilidade do problema.
C TO garantir que o problema é viável o usuário pode ADD
Adicional C ARCS custo muito pequeno.
C
C Este código pode falhar devido a integer overflow IF o número de nós
C ou o intervalo COST (ou ambos) são grandes. Para corrigir esta
situação,
C OS PREÇOS E outras variáveis relacionadas (Max1, max2, TMAX etc.)
devem
C DECLARAR AS REAIS precisão dupla.
C ********************************************** **********************
C ALL problema de dados são integer
```

```
C ********************************************** *****************


        SUBROUTINE AUCTION_AS (BEGEPS, fator, ENDEPS, STARTINCR)

        PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

        NONE IMPLICIT
        INTEIRO A, K, N, I, J, M, CURARC, CURCOL, Currow, LAMBDA,
DELTA
        INTEGER THRESH, INCR, STARTINCR, INCRFACTOR, EPS_THRESH
        INTEGER NOLIST, RNOLIST, NONEWLIST, RNONEWLIST
        INTEGER linha, coluna BSTROW, BSTCOL
        INTEGER FSTARC, FSTCOL, NXTARC, NXTROW, LSTARC, SNDARC, SNDCOL
        INTEGER Max1, MAX2, TMAX, TMIN, TRDARC, EPSILON, BEGEPS,
ENDEPS
        INTEGER ISMALL, ILARGE, LARGEINCR, CICLOS
        INTEGER NUMPHASES, OLDROW, OLDCOL
        INTEGER MAX3, BSTARC, SBSTARC
        INTEGER CUR_BARC, TRDVAL (maxNodes)
        LÓGICO INIT
        INTEGER BEST_ARC (maxNodes), SECD_ARC (maxNodes)
        INTEGER COL_ASSIGNED_TO (maxNodes), ROW_ASSIGNED_TO (maxNodes)
        INTEGER FOUT (maxNodes), FIN (maxNodes), NXTIN (MAXARCS)
        CUSTO INTEIRO (MAXARCS), START (MAXARCS), END (MAXARCS)
        LISTA INTEIRO (maxNodes), REV_LIST (maxNodes)
        INTEGER PCOL (maxNodes), PROW (maxNodes)
        Real * 8 AVERAGE, FACTOR
        COMUM / ARRAYC / CUSTO / matrizes / START / ARRAYE / END
        COMUM / ARRAYFO / FOUT / ARRAYFI / FIN / ARRAYNI / NXTIN
        COMUM / ARRAYRA / ROW_ASSIGNED_TO / ARRAYCA / COL_ASSIGNED_TO
        COMUM / ARRAYPC / PCOL / ARRAYPR / PROW
        COMUM / BK1 / M, N, A, ILARGE
        Common / BK2 / ciclos, Médio, NUMPHASES, LAMBDA

C ********************************************** **************

Check C ******* VALIDADE DOS PARÂMETROS ******* PASSOU

        IF (BEGEPS.LT.1) THEN
          PRINT *, 'Valor inicial EPSILON é inferior a 1'
          PRINT *, 'EXECUÇÃO ABORTADO'
          PARE
        END IF
        IF (ENDEPS.GT.BEGEPS) THEN
          PRINT *, 'ENDEPS parâmetro é maior que o parâmetro BEGEPS'
          PRINT *, 'ENDEPS é definido no valor padrão de 1'
          ENDEPS = 1
        END IF
        IF ((FACTOR.LE.1) .E. (BEGEPS.GT.1)) THEN
          PRINT *, 'Epsilon redução de fatores não é maior que 1'
          PRINT *, 'EXECUÇÃO ABORTADO'
          PARE
        END IF
        IF (STARTINCR.LT.1) THEN
          PRINT *, 'MIN DE LICITAÇÃO incremento é menor que 1'
          PRINT *, 'STARTINCR é definido no valor padrão de 1'
          STARTINCR = 1
        END IF
```

```
C ******* INITIALIZATION *******

          EPSILON = BEGEPS
          ISMALL = -ILARGE
          LARGEINCR = INT (ILARGE / 10)
          THRESH = INT (0,2 * M)
          INCRFACTOR = 2
          EPS_THRESH = BEGEPS / (FACTOR ** 3)
          IF (EPS_THRESH.LT.2) EPS_THRESH = 2
          INIT = .FALSE.
          IF (THRESH.GT.100) THRESH = 100
CICLOS x = 1
X MÉDIA = N
          NUMPHASES = 1

          FAZER 10 I = 1, N
            PCOL (I) = ISMALL
10 CONTINUAR

          FOUT (M + 1) = A + 1

C ********************************************** ***************
C
C Esta implementação do leilão ALGORITHM opera em ciclos.
C Cada ciclo consiste em um lance em cada uma das linhas QUE SEJAM
C não atribuído NO INÍCIO DO CICLO (essas linhas são armazenadas em
C lista de matriz (.)). AS que o ciclo progride NOVO
LINHAS C ficarem não atribuídas; Estes são armazenados em Lista (.)
C e apresentará um lance no próximo ciclo.
C
C O primeiro algoritmo encontra uma atribuição possível, onde todos
C forem atribuídos, USANDO UM COMBINADO COM EXPECTATIVA / leilão
reverso.
C ENTÃO O algoritmo usa um leilão reverso modificado para PARA
SATISFAZER
C DO RESTANTE CONDIÇÕES E-CS.
C
C ********************************************** ***************
C
C INÍCIO subproblema (FASE DE ESCALA) W / NEW EPSILON
C
C ********************************************** ***************

12 CONTINUAR

C INITIALIZE linha lista ATRIBUIÇÃO

         NOLIST = M
         FAZER 20 I = 1, M
            LISTA (I) = I
20 CONTINUAR

         DO 22 J = 1, N
            ROW_ASSIGNED_TO (J) = 0
22 CONTINUAR

         INCR = STARTINCR
         IF (INCR.GT.EPSILON) INCR = EPSILON
         IF (EPSILON.EQ.1) debulhar = 0
```

```
C ********************************************** ***************
C
C partida de avanço LEILÃO CICLO
C
C ********************************************** ***************

      IF (EPSILON.LT.EPS_THRESH) THEN

17 CONTINUAR

C REINICIAR CONTAGEM DO PRÓXIMO lista de linhas não atribuídos

        NONEWLIST = 0

C percorrer a lista atual de linhas não atribuídos

        FAZER 103 I = 1, NOLIST
          ROW = LIST (I)
          FSTARC = FOUT (ROW)
          LSTARC = FOUT (ROW + 1) -1
          FSTCOL = END (FSTARC)

C PRIMEIRA CUIDAR DA caso excepcional ROW tem apenas um ARC

          IF (FSTARC.EQ.LSTARC) THEN
            PCOL (FSTCOL) = PCOL (FSTCOL) + LARGEINCR
            PROW (ROW) = COST (FSTARC) -PCOL (FSTCOL)
            OLDROW = ROW_ASSIGNED_TO (FSTCOL)
            ROW_ASSIGNED_TO (FSTCOL) = ROW
            IF (OLDROW.GT.0) THEN
              NONEWLIST = + 1 NONEWLIST
              LIST (NONEWLIST) = OLDROW
            END IF
            IR PARA 103
          END IF

C tomar em seguida CUIDADOS DO CASO regular, onde linha tem ARCS
MÚLTIPLAS

          IF (((FSTARC + 2) .LE.LSTARC) .E. (INIT)) THEN

            BSTARC = BEST_ARC (ROW)
            SBSTARC = SECD_ARC (ROW)
            BSTCOL = END (BSTARC)
            SNDCOL = END (SECD_ARC (ROW))

            Max1 = COST (BSTARC) -PCOL (BSTCOL)
            MAX2 = COST (SBSTARC) -PCOL (SNDCOL)
            IF ((MAX1.GE.TRDVAL (ROW)). E. (MAX2.GE.TRDVAL (ROW)))
THEN

                IF (MAX1.LT.MAX2) THEN
                  TMAX = Max1
                  Max1 = MAX2
                  MAX2 = TMAX
                  BSTCOL = SNDCOL
                END IF
                IR PARA 70

             END IF
          END IF
```

```
          SNDARC = + 1 FSTARC
          SNDCOL = END (SNDARC)
          Max1 = COST (FSTARC) -PCOL (FSTCOL)
          MAX2 = COST (SNDARC) -PCOL (SNDCOL)
          IF (MAX1.GE.MAX2) THEN
              BSTARC = FSTARC
              SBSTARC = SNDARC
          MAIS
             TMAX = Max1
             Max1 = MAX2
             MAX2 = TMAX
             BSTARC = SNDARC
             SBSTARC = FSTARC
          END IF
          IF (SNDARC.LT.LSTARC) THEN
            TRDARC = + 1 SNDARC
            MAX3 = ISMALL
            FAZER 43 CURARC = TRDARC, LSTARC
               CURCOL = END (CURARC)
               TMAX = COST (CURARC) -PCOL (CURCOL)
               IF (TMAX.GT.MAX2) THEN
                IF (TMAX.GT.MAX1) THEN
                      SBSTARC = BSTARC
                      BSTARC = CURARC
                      MAX3 = MAX2
                      MAX2 = Max1
                      Max1 = TMAX
                 MAIS
                      SBSTARC = CURARC
                      MAX3 = MAX2
                      MAX2 = TMAX
                END IF
               MAIS
                 IF (MAX3.LT.TMAX) MAX3 = TMAX
               END IF
43 CONTINUAR
          END IF

        BEST_ARC (ROW) = BSTARC
        BSTCOL = END (BSTARC)
        SECD_ARC (ROW) = SBSTARC
        TRDVAL (ROW) = MAX3

70 CONTINUAR


APOSTAS C linha para BSTCOL AUMENTAR SEU PREÇO, E é atribuído
C TO BSTCOL, enquanto qualquer ROW ATRIBUÍDO BSTCOL TORNA não
atribuídos

          PCOL (BSTCOL) = PCOL (BSTCOL) + Max1-MAX2 + INCR
          PROW (ROW) = MAX2-INCR
          OLDROW = ROW_ASSIGNED_TO (BSTCOL)
          ROW_ASSIGNED_TO (BSTCOL) = ROW
          IF (OLDROW.GT.0) THEN
            NONEWLIST = + 1 NONEWLIST
            LIST (NONEWLIST) = OLDROW
          END IF

103 CONTINUAR
```

```
C ************* fim de um ciclo LEILÃO PARA A FRENTE ***************

C OPTIONALLY coletar estatísticas

X Médio = (ciclos * MÉDIA + NOLIST) / (CICLOS + 1)
CICLOS X = CICLOS + 1

C verificar se a mudança para MODIFICADO leilão reverso
C DEVE SER FEITO (IF NONEWLIST igual a zero).
C também aumentam a LICITAÇÃO MINIMAL INCREMENTO até um máximo
C VALOR DO EPSILON (esta é a característica adaptativa)

          INCR = INCR * INCRFACTOR
          IF (INCR.GT.EPSILON) INCR = EPSILON
          IF (NONEWLIST.GT.THRESH) THEN
              NOLIST = NONEWLIST
              INIT = .TRUE.
              IR PARA 17
          END IF

        MAIS

C ********************************************** ***************
C
C partida de avanço LEILÃO CICLO
C
C ********************************************** ***************

15 CONTINUAR

C REINICIAR CONTAGEM DO PRÓXIMO lista de linhas não atribuídos

         NONEWLIST = 0

C percorrer a lista atual de linhas não atribuídos

         FAZER 100 I = 1, NOLIST
           ROW = LIST (I)
           FSTARC = FOUT (ROW)
           LSTARC = FOUT (ROW + 1) -1
           FSTCOL = END (FSTARC)

C PRIMEIRA CUIDAR DA caso excepcional ROW tem apenas um ARC

           IF (FSTARC.EQ.LSTARC) THEN
             PCOL (FSTCOL) = PCOL (FSTCOL) + LARGEINCR
             PROW (ROW) = COST (FSTARC) -PCOL (FSTCOL)
             OLDROW = ROW_ASSIGNED_TO (FSTCOL)
             ROW_ASSIGNED_TO (FSTCOL) = ROW
             IF (OLDROW.GT.0) THEN
               NONEWLIST = + 1 NONEWLIST
               LIST (NONEWLIST) = OLDROW
             END IF
             Ir para 100
           END IF

C tomar em seguida CUIDADOS DO CASO regular, onde linha tem ARCS
MÚLTIPLAS

           SNDARC = + 1 FSTARC
```

```
          SNDCOL = END (SNDARC)
          Max1 = COST (FSTARC) -PCOL (FSTCOL)
          MAX2 = COST (SNDARC) -PCOL (SNDCOL)
          IF (MAX1.GE.MAX2) THEN
             BSTCOL = FSTCOL
          MAIS
             TMAX = Max1
             Max1 = MAX2
             MAX2 = TMAX
             BSTCOL = SNDCOL
          END IF
          IF (SNDARC.LT.LSTARC) THEN
            TRDARC = + 1 SNDARC
            DO 40 CURARC = TRDARC, LSTARC
               CURCOL = END (CURARC)
               TMAX = COST (CURARC) -PCOL (CURCOL)
               IF (TMAX.GT.MAX2) THEN
                 IF (TMAX.GT.MAX1) THEN
                      MAX2 = Max1
                      Max1 = TMAX
                      BSTCOL = CURCOL
                 MAIS
                      MAX2 = TMAX
                 END IF
               END IF
40 CONTINUAR
          END IF


APOSTAS C linha para BSTCOL AUMENTAR SEU PREÇO, E é atribuído
C TO BSTCOL, enquanto qualquer ROW ATRIBUÍDO BSTCOL TORNA não
atribuídos

          PCOL (BSTCOL) = PCOL (BSTCOL) + Max1-MAX2 + INCR
          PROW (ROW) = MAX2-INCR
          OLDROW = ROW_ASSIGNED_TO (BSTCOL)
          ROW_ASSIGNED_TO (BSTCOL) = ROW
          IF (OLDROW.GT.0) THEN
            NONEWLIST = + 1 NONEWLIST
            LIST (NONEWLIST) = OLDROW
          END IF

100 CONTINUAR

C ************* fim de um ciclo LEILÃO PARA A FRENTE ***************

C OPTIONALLY coletar estatísticas

X Médio = (ciclos * MÉDIA + NOLIST) / (CICLOS + 1)
CICLOS X = CICLOS + 1

C verificar se a mudança para MODIFICADO leilão reverso
C DEVE SER FEITO (IF NONEWLIST HÁ MAIS do que o limite).
C também aumentam a LICITAÇÃO MINIMAL INCREMENTO até um máximo
C VALOR DO EPSILON (esta é a característica adaptativa)

         INCR = INCR * INCRFACTOR
         IF (INCR.GT.EPSILON) INCR = EPSILON
         IF (NONEWLIST.GT.THRESH) THEN
             NOLIST = NONEWLIST
             IR PARA 15
```

```
         END IF

      END IF

C ********************************************** **************
C
C INÍCIO DE leilão reverso MODIFICADO
C
C ********************************************** **************

       IF (EPSILON.GT.1) GOTO 400
       IF (N.EQ.M) GOTO 400

       INCR = EPSILON


C COMPUTE LAMBDA que é o mínimo coluna de preços sobre todos
C COLUNAS atribuído, e compilar a lista de colunas não atribuídos

       LAMBDA = ILARGE
       RNOLIST = 0
       FAZER 310 J = 1, N
         ROW = ROW_ASSIGNED_TO (J)
         IF (ROW.GT.0) THEN
           COL_ASSIGNED_TO (ROW) = J
           IF (LAMBDA.GT.PCOL (J)) LAMBDA = PCOL (J)
         MAIS
           RNOLIST = + 1 RNOLIST
           REV_LIST (RNOLIST) = J
         END IF
310 CONTINUAR


       IF (RNOLIST.NE.NM) THEN
          PRINT *, 'número de linhas não definidos é errado "
          PAUSA
          PARE
       END IF


C início de um novo ciclo reverso MODIFICADO LEILÃO


315 CONTINUAR

C REINICIAR CONTAGEM DO PRÓXIMO lista de colunas não atribuídos

        RNONEWLIST = 0

C percorrer a lista atual de COLUNAS não atribuídos

        FAZER 340 J = 1, RNOLIST
          COLUNA = REV_LIST (J)
          IF (PCOL (coluna) .LE.LAMBDA) ir para 340
          CURARC = FIN (coluna)
          NXTARC = NXTIN (CURARC)
          Currow = START (CURARC)

C PRIMEIRA CUIDAR DA caso excepcional ROW tem apenas um ARC

          IF (NXTARC.EQ.0) THEN
```

```
            Max1 = COST (CURARC) -PROW (Currow)
            IF (LAMBDA.GE.MAX1-INCR) THEN
               PCOL (coluna) = LAMBDA
               IR PARA 340
            END IF
            PCOL (coluna) = LAMBDA
            PROW (Currow) = COST (CURARC) -LAMBDA
            OLDCOL = COL_ASSIGNED_TO (Currow)
            COL_ASSIGNED_TO (Currow) = COLUNA
            IF (PCOL (OLDCOL) .GT.LAMBDA) THEN
              RNONEWLIST = + 1 RNONEWLIST
              REV_LIST (RNONEWLIST) = OLDCOL
            END IF
            IR PARA 340
          END IF
```

C tomar em seguida CUIDADOS DO CASO regular, onde linha tem ARCS MÚLTIPLAS

```
          NXTROW = START (NXTARC)
          Max1 = COST (CURARC) -PROW (Currow)
          MAX2 = COST (NXTARC) -PROW (NXTROW)
          IF (MAX1.GE.MAX2) THEN
             BSTROW = Currow
          MAIS
             TMAX = Max1
             Max1 = MAX2
             MAX2 = TMAX
             BSTROW = NXTROW
          END IF

          CURARC = NXTIN (NXTARC)

330 IF (CURARC.GT.0) THEN
            Currow = START (CURARC)
            TMAX = COST (CURARC) -PROW (Currow)
            IF (TMAX.GT.MAX2) THEN
              IF (TMAX.GT.MAX1) THEN
                   MAX2 = Max1
                   Max1 = TMAX
                   BSTROW = Currow
              MAIS
                   MAX2 = TMAX
              END IF
            END IF
            CURARC = NXTIN (CURARC)
            IR PARA 330
          END IF
```

APOSTAS C COLUNA BSTROW AUMENTAR SEU PREÇO, E é atribuído C TO BSTROW, enquanto qualquer COLUNA ATRIBUÍDO BSTROW TORNA não atribuídos

```
          IF (LAMBDA.GE.MAX1-INCR) THEN
             PCOL (coluna) = LAMBDA
             IR PARA 340
          END IF
          DELTA = Max1-MAX2 + INCR
          IF (DELTA.GT.MAX1-LAMBDA) DELTA = Max1-LAMBDA
```

```
          PROW (BSTROW) = PROW (BSTROW) + DELTA
          PCOL (coluna) = Max1-DELTA
          OLDCOL = COL_ASSIGNED_TO (BSTROW)
          COL_ASSIGNED_TO (BSTROW) = COLUNA
          IF (PCOL (OLDCOL) .GT.LAMBDA) THEN
            RNONEWLIST = + 1 RNONEWLIST
            REV_LIST (RNONEWLIST) = OLDCOL
          END IF

340 CONTINUAR

C ************* fim de um ciclo de leilão reverso MODIFICADO
***************

C OPTIONALLY coletar estatísticas

X Médio = (ciclos * MÉDIA + RNOLIST) / (CICLOS + 1)
CICLOS X = CICLOS + 1

C Verifique se ainda existem COLUNAS ELEGÍVEIS não atribuído 'muitos',
isto é,
C Se o número de colunas não atribuído ELEGÍVEIS É MAIOR DO QUE
C DO THRESH PARAMETER. NÃO SE, REPLACE lista atual com a lista NEW,
C e ir para outro ciclo. Caso contrário, se EPSILON> 1, REDUZIR
EPSILON,
C Redefinir a atribuição ao vazio e REINICIAR LEILÃO;
C IF EPSILON = 1 finalizar.


         IF (RNONEWLIST.GT.THRESH) THEN
           RNOLIST = RNONEWLIST
           IR PARA 315
         END IF


C ********************************************** **************
C
C FIM DE subproblema (escalonamento FASE)
C
C ********************************************** **************

400 CONTINUAR

C ****** SE EPSILON É uma RESCINDIR ******

       IF (EPSILON.EQ.1) THEN
          RETURN
       MAIS

C MAIS REDUZIR EPSILON e atualizar os parâmetros para a FASE DE ESCALA
PRÓXIMO

          NUMPHASES + 1 = NUMPHASES
          EPSILON = INT (EPSILON / FACTOR)
          IF (EPSILON.GT.INCR) EPSILON = INT (EPSILON / FACTOR)
          IF ((EPSILON.LT.1) .OR. (EPSILON.LT.ENDEPS)) EPSILON = 1
          IF (STARTINCR.LT.EPSILON) STARTINCR = FACTOR * STARTINCR
          THRESH = INT (THRESH / FACTOR)


X PRINT *, '*** FIM DE UMA FASE DE ESCALA; NEW EPSILON = ', EPSILON
```

```
          IR PARA 12
        END IF


        FIM


      SUBROUTINE SETRAN (ISEED)
      IMPLICIT real * 8 (AH, OZ), Integer * 4 (IN)
C ***********************************************
**********************
C PORTABLE congruential (uniforme) RANDOM gerador de números:
NEXT_VALUE C = [(7 ** 5) * PREVIOUS_VALUE] MODULO [(2 ** 31) -1]
C
C Este gerador é composto por duas ROTINAS:
C (1) SETRAN - inicializa constantes e SEED
C (2) RRAN - gera um número aleatório REAIS
C
C O GERADOR DE RODA uma máquina com pelo menos 32 bits de precisão.
C DA SEED (ISEED) deve estar no intervalo (1, (2 ** 31 -1)).
C ***********************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      IF (ISEED.LT.1) PARADA 77
      MULT = 16807
      MODUL = 2147483647
      I15 = 2 ** 15
      I16 = 2 ** 16
      JRAN = ISEED
      RETURN
      FIM
C
      RAN função real ()
      IMPLICIT real * 4 (AH, OZ), Integer * 4 (IN)
C ***********************************************
**********************
C RAN gera um número aleatório real entre 0 e 1
C ***********************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      Ixhi = JRAN / I16
      IXLO = JRAN-Ixhi * I16
      IXALO = IXLO * MULT
      LEFTLO = IXALO / I16
      IXAHI = Ixhi * MULT
      IFULHI = IXAHI + LEFTLO
      IRTLO = IXALO-LEFTLO * I16
      IOVER = IFULHI / I15
      IRTHI = IFULHI-IOVER * I15
      JRAN = ((IRTLO-MODUL) + IRTHI * I16) + IOVER
      IF (JRAN.LT.0) JRAN = JRAN + MODUL
      RAN = FLOAT (JRAN) / FLOAT (MODUL)
      RETURN
      FIM
```

```
Este código implementa o leilão reverso / combinado para a frente
algoritmo com $ \ e $ scaling para o problema de atribuição simétrica;
cf. \ Seção
4.2. Para um bom desempenho, ele freqüentemente não é imprescindível a
utilização de $ \ e $ scaling em
este código. Portanto, é recomendável que o código ser julgado sem
$ \ E $ scaling, definindo a partida $ \ e $ 1.

C ********************************************** ****************
C
C programa de amostra CHAMADA PARA COMBINADO frente / para trás
C ALGORITHM LEILÃO
C
C ESTE MOTORISTA cria um problema SIMÉTRICA CESSÃO
C com igual número de linhas e colunas,
C e chama o SUBROUTINE AUCTION_FR para encontrar um
C atribuição do valor máximo.
C
C ********************************************** ****************


        PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

        NONE IMPLICIT
        INTEGER N, NA, A, IA, ILARGE, BEGEPS, ENDEPS, CICLOS
        INTEGER NUMPHASES, STARTINCR
        INTEGER I, J, ARC, NOASS, ICOST, ABSCOST, CURARC
        INTEGER CURCOL, FSTARC, LSTARC, MINCOST, MAXCOST, MCOST
        INTEGER COL_ASSIGNED_TO (maxNodes), ROW_ASSIGNED_TO (maxNodes)
        INTEGER FOUT (maxNodes), FIN (maxNodes), NXTIN (MAXARCS)
        CUSTO INTEIRO (MAXARCS), START (MAXARCS), END (MAXARCS)
        INTEGER PCOL (maxNodes), PROW (maxNodes)
        INTEGER PRDARC (maxNodes)
        Real * 8 FACTOR, TT1, TT2, TCOST, AVERAGE
        COMUM / ARRAYC / CUSTO / matrizes / START / ARRAYE / END
        COMUM / ARRAYFO / FOUT / ARRAYFI / FIN / ARRAYNI / NXTIN
        COMUM / ARRAYRA / ROW_ASSIGNED_TO / ARRAYCA / COL_ASSIGNED_TO
        COMUM / ARRAYPC / PCOL / ARRAYPR / PROW
        Common / BK1 / N, A, ILARGE / BK2 / ciclos, Médio, NUMPHASES

C ********************************************** **************
C
C problema de geração de código começa AQUI
C o usuário pode substituir este código com um código que lê
C HIS / seu problema de um arquivo
C
C ********************************************** **************

C Este código inclui UM UNIFORME Random Number Generator
C que retorna um valor na (0,1)

C INITIALIZE Gerador Aleatório

      INTEGER MULT, MODUL, I15, I16, JRAN, ISEED
      RAN REAIS
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN

      ISEED = 13502460
      CHAME SETRAN (ISEED)

      PRINT * ', gerando problema de atribuição de SIMÉTRICA "
```

```
      PRINT *, "****************************************** '

C **** LER O número de linhas N & o número de arcos A ****

      PRINT *, 'Digite o número de linhas (e colunas)'
      LEIA *, N
5 PRINT *, 'Digite o número de arcos por ROW (1>) "
      LEIA *, NA
      IF (NA.LT.2) GOTO 5
      PRINT *, 'Digite o mínimo eo custo máximo'
      LEIA *, MINCOST, MAXCOST

C o número de arcos é n * NA

      A = N * NA

C Os arcos incidente a linha i SÃO FOUT (I) PARA FOUT (I + 1) -1
C Além disso, para VIABILIDADE EACH ROW está diretamente ligado
C com a coluna correspondente

      FAZER 20 I = 1, N
        FOUT (I) = 1 + (i-1) * NA
20 CONTINUAR
      FOUT (N + 1) = A + 1

      FAZER 22 I = 1, N
        FAZER 23 ARC = FOUT (I), FOUT (I + 1) -1
          START (ARC) = I
23 CONTINUAR
22 CONTINUAR

C gerar a END (ARC) eo custo (ARC), que são os COLUNA
C eo coeficiente custo associado a ARC

      FAZER 25 ARC = 1, A
        END (ARC) = 1 + RAN () * N
        IF ((END (ARC) .GT.N) .OR. (END (ARC) .LT.1)) THEN
          PRINT *, 'Erro no problema de geração de'
          PAUSA
          PARE
        END IF
        COST (ARC) = MINCOST + RAN () * (MAXCOST-MINCOST)
25 CONTINUAR

C MODIFICAR O final da última ARC OUT de cada linha de viabilidade
C e defina seu CUSTO PARA -MAXCOST

      FAZER 30 I = 1, N
        END (FOUT (I + 1 -1)) = I
        CUSTO (FOUT (I + 1) -1) = - MAXCOST
30 CONTINUAR

C construir a FIN () e NXTIN () ARRAYS

      DO 32 J = 1, N
        FIN (J) = 0
        PRDARC (J) = 0
32 CONTINUAR

      FAZER 33 ARC = 1, A
        NXTIN (ARC) = 0
```

```
        J = END (ARC)
        IF (FIN (J) .NE.0) THEN
          NXTIN (PRDARC (J)) = ARC
        MAIS
          FIN (J) = ARC
        END IF
        PRDARC (J) = ARC
33 CONTINUAR


C ********************************************** **************
C
C PROBLEMA geração de código TERMINA AQUI
C
C ********************************************** **************


C ESCALA O CUSTO PARA TRABALHAR COM INTEIRO EPSILON

        MAXCOST = 0
        FAZER 35 IA = 1, A
           ABSCOST = IABS (COST (IA))
           IF (ABSCOST.GT.MAXCOST) MAXCOST = ABSCOST
           COST (IA) = CUSTO (IA) * (N + 1)
35 CONTINUAR

C *** ILARGE é um inteiro MUITO GRANDE PARA SUA MÁQUINA ***
C OS CUSTOS escalado deve ser significativamente menor do
C ILARGE E significativamente maior do que -ILARGE


        ILARGE = 2000000000


        IF (MAXCOST.GT.INT (ILARGE / (N + 1))) THEN
          PRINT *, 'A faixa de custo é muito grande para INTEGER
ARITHMETIC'
          PAUSA
          PARE
        END IF

        MAXCOST MAXCOST = * (N + 1)


C os seguintes parâmetros BEGEPS, FACTOR, ENDEPS E STARTINCR
C são passados para o LEILÃO algoritmo. Valores entre
C (A) MAXCOST / 2 e 1 para BEGEPS
C (B) 4 e 6 para FACTOR
C (C) N e 1 para ENDEPS
C (D) 1 AND BEGEPS PARA STARTINCR
C têm funcionado bem para PROBLEMAS esparsos de grande porte.
C PARA PROBLEMAS DENSOS RECOMENDA-SE QUE
C BEGEPS ser ajustado para um valor menor,
ENDEPS C ser definido como 1,
C STARTINCR ser definido como 1.

C Geralmente, o atacante COMBINADO / REVERSE ALGORITHM
C funciona melhor com o menor valor de BEGEPS DO QUE A
C algoritmo Forward.
C Vale a pena tentar BEGEPS = 1, CASO EM QUE HÁ
C só UMA ESCALA DE FASE (ou seja, nenhum E-escala que é utilizada).
```

```
        PRINT *, "****************************************** '
        PRINT *, "custo máximo é ', MAXCOST
        PRINT *, 'Digite o EPSILON PARTIDA'
        LER *, BEGEPS
        IF (BEGEPS.LT.1) BEGEPS = 1
        PRINT *, 'Insira o fator EPSILON REDUÇÃO'
        LER *, FACTOR
        ENDEPS = N / 10
        IF (ENDEPS.LT.1) ENDEPS = 1
        IF (ENDEPS.GT.BEGEPS) ENDEPS = BEGEPS
        STARTINCR = 1
        IF (STARTINCR.LT.1) STARTINCR = 1

        PRINT *, "****************************************** '
        PRINT *, 'começar EPSILON =', BEGEPS
        PRINT *, 'Epsilon redução de fatores =', FACTOR
        PRINT *, «limiar EPSILON ANTES ela é definida como 1 = ',
ENDEPS
        PRINT *, "COMEÇAR MIN DE LICITAÇÃO INCREMENTO = ', STARTINCR
        PRINT *, "CHAMANDO A FRENTE / leilão reverso '
        PRINT *, "****************************************** '

C GET hora de início para o Mac II

      TT1 = longa (362) /60.0

      CHAMADA AUCTION_FR (BEGEPS, fator, ENDEPS, STARTINCR)

C GET TÉRMINO TEMPO PARA O MAC II

      TT2 = longa (362) /60.0 - TT1
      PRINT *, "acabado --- tempo de CPU TOTAL ', TT2,' s '
      PRINT *, "****************************************** '

C *** Mostrar resultados ***

X Write (9,2010) CICLOS
X2010 FORMATO ('NO LEILÃO DE CICLOS', I7)
X WRITE (9,2020) MÉDIA
X2020 FORMATO ("número médio de propostas por ciclo", F9.3)
       Write (9,2030) NUMPHASES
2030 FORMATO ('NO DE EPSILON subproblemas RESOLVIDO =', I7)

C VERIFICAÇÃO Optimality & calcular o custo

        TCOST = 0
        DO 40 I = 1, N
          J = COL_ASSIGNED_TO (I)
          IF (J.EQ.0) THEN
              PRINT *, 'ROW', I, 'não é atribuído "
          END IF
          IF (ROW_ASSIGNED_TO (J) .NE.I) THEN
              PRINT *, 'Atribuição confusão: ROW', I, 'COLUMN', J
          END IF
          FSTARC = FOUT (I)
          LSTARC FOUT = (I + 1) -1
          MCOST = -ILARGE
          FAZER 45 ARC = FSTARC, LSTARC
            CURCOL = END (ARC)
            IF (PROW (I) + PCOL (CURCOL) .LT.COST (ARC -1)) THEN
```

```
              PRINT *, '1-CS VIOLADOS AT ARC ", ARC
            END IF
            IF (CURCOL.EQ.J) THEN
              IF (MCOST.LT.COST (ARC)) THEN
                 MCOST = CUSTO (ARC)
              END IF
            END IF
45 CONTINUAR
          TCOST = TCOST MCOST + / (N + 1)
          IF (PROW (I) + PCOL (J) .NE.MCOST) THEN
            PRINT *, '1-CS VIOLADA na linha', I
          END IF
40 CONTINUAR

        FAZER 50 J = 1, N
           IF (ROW_ASSIGNED_TO (J) .EQ.0) THEN
              PRINT *, 'COLUMN', J, "não é atribuído"
           END IF
50 CONTINUAR

        ESCREVA (9,2100) TCOST
2100 FORMATO ('atribuição de custo =', F18.2)
        PRINT *, 'programa terminou; <CR> PARA SAIR '
        PAUSA
        FIM



C ********************************************** ****************
C
C frente / para trás CODE LEILÃO PARA N por N problemas de atribuição
C
C ESCRITO POR DIMITRI P. Bertsekas
C
C dezembro 1990
C
C ESTE CÓDIGO implementa a frente / para trás ALGORITHM LEILÃO
C COM E-escala para SIMÉTRICA N por N problemas de atribuição.
C Ele resolve uma seqüência de subproblemas e diminui
C EPSILON por um fator constante entre subproblemas.
C ESTE versão corresponda a um MODO Gauss-Seidel
C e resolve EPSILON subproblemas inexata.
C
C THE CODE é uma versão melhorada de um antigo (SEPT. 1985)
C CODE LEILÃO COM ESCALA E-ESCRITO POR DIMITRI P. Bertsekas
C
C DO CÓDIGO trata o problema como um problema de maximização.
C para resolver um problema de minimização, inverta o sinal da
C ARC CUSTOS antes de chamar LEILÃO, E RECUO DE NOVO
C O SINAL DO custo ótimo na volta do LEILÃO.
C Este código permite ARCS múltipla entre uma linha e uma coluna.
C
C Esta versão do ALGORITHM LEILÃO é adaptativa. AQUI NO MÍNIMO
INCREMENTO DE LICITAÇÃO C (armazenado no INCR variável) pode ser menor
que
C EPSILON. PARA CADA subproblema, INCR COMEÇA no valor do parâmetro
C STARTINCR (que é passado para a rotina LEILÃO), estando aumentada
C por um fator de 2 no final de cada ciclo, até um valor máximo de
C EPSILON. ESTA FUNÇÃO adaptativo é particularmente eficaz para
```

```
C PROBLEMAS densa. TI pode ser derrotado por SELEÇÃO STARTINCR =
BEGEPS
C
C Este código pode falhar devido a integer overflow IF o número de nós
C ou o intervalo COST (ou ambos) são grandes. Para corrigir esta
situação,
C OS PREÇOS E outras variáveis relacionadas (Max1, max2, TMAX etc.)
devem
C DECLARAR AS REAIS precisão dupla.
C ************************************************
******************
C
C o usuário deve fornecer os seguintes dados problema no FRENTE DA
ESTRELA FORMATO
C (ou seja, todos os arcos da linha SAME são numerados
consecutivamente):
CN = número de linhas (igual número de colunas)
CA = número de arcos
C FOUT (ROW) = Primeiro ARC QUE SAI DE ROW
C FIN (COL) = PRIMEIRO ARC VINDO EM COL
C NXTIN (ARC) = Próxima ARC incidente à COLUNA MESMO AS ARC
C COST (ARC) = CUSTO DE ARC
C START (ARC) = ROW correspondente ao arco
C END (ARC) = COLUNA correspondente ao arco
C
C E os seguintes parâmetros para o algoritmo LEILÃO:
C BEGEPS = Valor inicial EPSILON (não deve ser menor que 1)
C ENDEPS = valor final da EPSILON ANTES ela é definida como 1
C FACTOR = fator pelo qual EPSILON é diminuída ENTRE subproblemas
C STARTINCR = O valor inicial do incremento LICITAÇÃO
ENDEPS C não devem exceder BEGEPS.
C fator deve ser superior a 1 (a menos que BEGEPS = 1).
C
C FOUT (.) É uma matriz de comprimento N.
C COST (.), END (.) São matrizes de COMPRIMENTO A.
C
C a solução é contidos na matriz ROW_ASSIGNED_TO (.) ONDE
C ROW_ASSIGNED_TO (COL) Dá o ROW ATRIBUÍDO COL.
C TAMBÉM COL_ASSIGNED_TO (ROW) DÁ A COLUNA ATRIBUÍDO linha.
C
C Este algoritmo não verifica a inviabilidade do problema.
C TO garantir que o problema é viável o usuário pode ADD
Adicional C ARCS custo muito pequeno.
C
C ********************************************** **********************
C ALL problema de dados são integer
C ********************************************** **********************


        SUBROUTINE AUCTION_FR (BEGEPS, fator, ENDEPS, STARTINCR)

        PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

        NONE IMPLICIT
        INTEIRO A, K, N, I, J, M, CURARC, CURCOL, Currow
        INTEGER THRESH, INCR, STARTINCR, INCRFACTOR
        INTEGER NOLIST, RNOLIST, NONEWLIST, RNONEWLIST
        INTEGER linha, coluna BSTROW, BSTCOL
        INTEGER FSTARC, FSTCOL, NXTARC, NXTROW, LSTARC, SNDARC, SNDCOL
        INTEGER Max1, MAX2, TMAX, TMIN, TRDARC, EPSILON, BEGEPS,
ENDEPS
```

```
        INTEGER ISMALL, ILARGE, LARGEINCR, CICLOS
        INTEGER NUMPHASES, OLDROW, OLDCOL
        INTEGER COL_ASSIGNED_TO (maxNodes), ROW_ASSIGNED_TO (maxNodes)
        INTEGER FOUT (maxNodes), FIN (maxNodes), NXTIN (MAXARCS)
        CUSTO INTEIRO (MAXARCS), START (MAXARCS), END (MAXARCS)
        LISTA INTEIRO (maxNodes), REV_LIST (maxNodes)
        INTEGER PCOL (maxNodes), PROW (maxNodes)
        Real * 8 AVERAGE, FACTOR
        READY_SWITCH LÓGICO, Switch
        COMUM / ARRAYC / CUSTO / matrizes / START / ARRAYE / END
        COMUM / ARRAYFO / FOUT / ARRAYFI / FIN / ARRAYNI / NXTIN
        COMUM / ARRAYRA / ROW_ASSIGNED_TO / ARRAYCA / COL_ASSIGNED_TO
        COMUM / ARRAYPC / PCOL / ARRAYPR / PROW
        Common / BK1 / N, A, ILARGE / BK2 / ciclos, Médio, NUMPHASES

C ********************************************** **************

Check C ******* VALIDADE DOS PARÂMETROS ******* PASSOU

        IF (BEGEPS.LT.1) THEN
          PRINT *, 'Valor inicial EPSILON é inferior a 1'
          PRINT *, 'EXECUÇÃO ABORTADO'
          PARE
        END IF
        IF (ENDEPS.GT.BEGEPS) THEN
          PRINT *, 'ENDEPS parâmetro é maior que o parâmetro BEGEPS'
          PRINT *, 'ENDEPS é definido no valor padrão de 1'
          ENDEPS = 1
        END IF
        IF ((FACTOR.LE.1) .E. (BEGEPS.GT.1)) THEN
          PRINT *, 'Epsilon redução de fatores não é maior que 1'
          PRINT *, 'EXECUÇÃO ABORTADO'
          PARE
        END IF
        IF (STARTINCR.LT.1) THEN
          PRINT *, 'MIN DE LICITAÇÃO incremento é menor que 1'
          PRINT *, 'STARTINCR é definido no valor padrão de 1'
          STARTINCR = 1
        END IF

C ******* INITIALIZATION *******

        EPSILON = BEGEPS
        ISMALL = -ILARGE
        LARGEINCR = INT (ILARGE / 10)
        THRESH = INT (0,2 * N)
        INCRFACTOR = 2
        IF (THRESH.GT.100) THRESH = 100
CICLOS x = 1
X MÉDIA = N
         NUMPHASES = 1

C INITIALIZE FORWARD / PARÂMETROS botão de reversão

         READY_SWITCH = .FALSE.
         CHAVE = .TRUE.


         FAZER 10 J = 1, N
           PCOL (J) = 0
10 CONTINUAR
```

```
          FOUT (N + 1) = A + 1

C ********************************************** **************
C
C Esta implementação do leilão ALGORITHM opera em ciclos.
C cada ciclo LEILÃO PARA A FRENTE consiste em um lance em cada uma das
linhas
C não atribuídos NO INÍCIO DO CICLO (essas linhas SÃO
C armazenado na lista de matriz (.)). AS que o ciclo progride NOVO
LINHAS C ficarem não atribuídas; Estes são armazenados em Lista (.)
C e apresentará um lance no próximo ciclo.
C leilão reverso ciclos são estruturado de forma similar.
C
C ********************************************** **************
C
C INÍCIO subproblema (FASE DE ESCALA) W / NEW EPSILON
C
C ********************************************** **************
C
12 CONTINUAR

C REINICIAR linha e coluna lista ATRIBUIÇÃO

         NOLIST = N
         FAZER 20 I = 1, N
            COL_ASSIGNED_TO (I) = 0
            LISTA (I) = I
20 CONTINUAR

         RNOLIST = N
         DO 22 J = 1, N
            ROW_ASSIGNED_TO (J) = 0
            REV_LIST (J) = J
22 CONTINUAR

         INCR = STARTINCR
         IF (INCR.GT.EPSILON) INCR = EPSILON
         IF (EPSILON.EQ.1) debulhar = 0

C ********************************************** **************
C
C partida de avanço LEILÃO CICLO
C
C ********************************************** **************

15 CONTINUAR

C REINICIAR CONTAGEM DO PRÓXIMO lista de linhas não atribuídos

         NONEWLIST = 0

C percorrer a lista atual de linhas não atribuídos

         FAZER 100 I = 1, NOLIST
           ROW = LIST (I)
           IF (COL_ASSIGNED_TO (ROW) .GT.0) ir para 100
           FSTARC = FOUT (ROW)
           LSTARC = FOUT (ROW + 1) -1
           FSTCOL = END (FSTARC)
```

```
C PRIMEIRA CUIDAR DA caso excepcional ROW tem apenas um ARC

          IF (FSTARC.EQ.LSTARC) THEN
            PCOL (FSTCOL) = PCOL (FSTCOL) + LARGEINCR
            PROW (ROW) = COST (FSTARC) -PCOL (FSTCOL)
            OLDROW = ROW_ASSIGNED_TO (FSTCOL)
            ROW_ASSIGNED_TO (FSTCOL) = ROW
            COL_ASSIGNED_TO (ROW) = FSTCOL
            IF (OLDROW.GT.0) THEN
              COL_ASSIGNED_TO (OLDROW) = 0
              NONEWLIST = + 1 NONEWLIST
              LIST (NONEWLIST) = OLDROW
            END IF
            Ir para 100
          END IF

C tomar em seguida CUIDADOS DO CASO regular, onde linha tem ARCS
MÚLTIPLAS

          SNDARC = + 1 FSTARC
          SNDCOL = END (SNDARC)
          Max1 = COST (FSTARC) -PCOL (FSTCOL)
          MAX2 = COST (SNDARC) -PCOL (SNDCOL)
          IF (MAX1.GE.MAX2) THEN
             BSTCOL = FSTCOL
          MAIS
             TMAX = Max1
             Max1 = MAX2
             MAX2 = TMAX
             BSTCOL = SNDCOL
          END IF
          IF (SNDARC.LT.LSTARC) THEN
            TRDARC = + 1 SNDARC
            DO 40 CURARC = TRDARC, LSTARC
               CURCOL = END (CURARC)
               TMAX = COST (CURARC) -PCOL (CURCOL)
               IF (TMAX.GT.MAX2) THEN
                 IF (TMAX.GT.MAX1) THEN
                      MAX2 = Max1
                      Max1 = TMAX
                      BSTCOL = CURCOL
                 MAIS
                      MAX2 = TMAX
                 END IF
               END IF
40 CONTINUAR
          END IF


APOSTAS C linha para BSTCOL AUMENTAR SEU PREÇO, E é atribuído
C TO BSTCOL, enquanto qualquer ROW ATRIBUÍDO BSTCOL TORNA não
atribuídos

          PCOL (BSTCOL) = PCOL (BSTCOL) + Max1-MAX2 + INCR
          PROW (ROW) = MAX2-INCR
          COL_ASSIGNED_TO (ROW) = BSTCOL
          OLDROW = ROW_ASSIGNED_TO (BSTCOL)
          ROW_ASSIGNED_TO (BSTCOL) = ROW
          IF (OLDROW.GT.0) THEN
            COL_ASSIGNED_TO (OLDROW) = 0
            NONEWLIST = + 1 NONEWLIST
```

```
            LIST (NONEWLIST) = OLDROW
          END IF

100 CONTINUAR

C ************* fim de um ciclo LEILÃO PARA A FRENTE ***************

C OPTIONALLY coletar estatísticas

X Médio = (ciclos * MÉDIA + NOLIST) / (CICLOS + 1)
CICLOS X = CICLOS + 1

C Verifique se ainda existem ROWS não atribuído 'muitos', isto é, se o
C número de linhas não atribuído É MAIOR DO QUE
C DO THRESH PARAMETER. NÃO SE, REPLACE lista atual com a lista NEW,
C e ir para outro ciclo. Caso contrário, se EPSILON> 1, REDUZIR
EPSILON,
C Redefinir a atribuição ao vazio e REINICIAR LEILÃO;
C IF EPSILON = 1 finalizar.
C também aumentam a LICITAÇÃO MINIMAL INCREMENTO até um máximo
C VALOR DO EPSILON (esta é a característica adaptativa)


          INCR = INCR * INCRFACTOR
          IF (INCR.GT.EPSILON) INCR = EPSILON
          IF (NONEWLIST.GT.THRESH) THEN
              IF (CHAVE) THEN
                NOLIST = NONEWLIST
                IR PARA 115
              END IF
              IF (NONEWLIST.LT.NOLIST) READY_SWITCH = .TRUE.
              IF ((NONEWLIST.EQ.NOLIST) .E. (READY_SWITCH)) THEN
                READY_SWITCH = .FALSE.
                IR PARA 115
              MAIS
                NOLIST = NONEWLIST
                IR PARA 15
              END IF
          MAIS
              IR PARA 300
          END IF


C ********************************************** ***************
C
C Início de ciclo reverso LEILÃO
C
C ********************************************** ***************

115 CONTINUAR

C REINICIAR CONTAGEM DO PRÓXIMO lista de colunas não atribuídos

         RNONEWLIST = 0

C percorrer a lista atual de COLUNAS não atribuídos

         FAZER 200 J = 1, RNOLIST
           COLUNA = REV_LIST (J)
           IF (ROW_ASSIGNED_TO (coluna) .GT.0) ir para 200
           CURARC = FIN (coluna)
```

```
          NXTARC = NXTIN (CURARC)
          Currow = START (CURARC)

C PRIMEIRA CUIDAR DA caso excepcional ROW tem apenas um ARC

          IF (NXTARC.EQ.0) THEN
            PROW (Currow) = PROW (Currow) + LARGEINCR
            PCOL (coluna) = COST (CURARC) -PROW (Currow)
            OLDCOL = COL_ASSIGNED_TO (Currow)
            COL_ASSIGNED_TO (Currow) = COLUNA
            ROW_ASSIGNED_TO (coluna) = Currow
            IF (OLDCOL.GT.0) THEN
              ROW_ASSIGNED_TO (OLDCOL) = 0
              RNONEWLIST = + 1 RNONEWLIST
              REV_LIST (RNONEWLIST) = OLDCOL
            END IF
            IR PARA 200
          END IF

C tomar em seguida CUIDADOS DO CASO regular, onde linha tem ARCS
MÚLTIPLAS

          NXTROW = START (NXTARC)
          Max1 = COST (CURARC) -PROW (Currow)
          MAX2 = COST (NXTARC) -PROW (NXTROW)
          IF (MAX1.GE.MAX2) THEN
             BSTROW = Currow
          MAIS
             TMAX = Max1
             Max1 = MAX2
             MAX2 = TMAX
             BSTROW = NXTROW
          END IF

          CURARC = NXTIN (NXTARC)

130 IF (CURARC.GT.0) THEN
            Currow = START (CURARC)
            TMAX = COST (CURARC) -PROW (Currow)
            IF (TMAX.GT.MAX2) THEN
              IF (TMAX.GT.MAX1) THEN
                   MAX2 = Max1
                   Max1 = TMAX
                   BSTROW = Currow
              MAIS
                   MAX2 = TMAX
              END IF
            END IF
            CURARC = NXTIN (CURARC)
            IR PARA 130
          END IF


APOSTAS C COLUNA BSTROW AUMENTAR SEU PREÇO, E é atribuído
C TO BSTROW, enquanto qualquer COLUNA ATRIBUÍDO BSTROW TORNA não
atribuídos


          PROW (BSTROW) = PROW (BSTROW) + Max1-MAX2 + INCR
          PCOL (coluna) = MAX2-INCR
          ROW_ASSIGNED_TO (coluna) = BSTROW
```

```
          OLDCOL = COL_ASSIGNED_TO (BSTROW)
          COL_ASSIGNED_TO (BSTROW) = COLUNA
          IF (OLDCOL.GT.0) THEN
            ROW_ASSIGNED_TO (OLDCOL) = 0
            RNONEWLIST = + 1 RNONEWLIST
            REV_LIST (RNONEWLIST) = OLDCOL
          END IF

200 CONTINUAR

C ************* fim de um ciclo leilão reverso ***************

C OPTIONALLY coletar estatísticas

X Médio = (ciclos * MÉDIA + RNOLIST) / (CICLOS + 1)
CICLOS X = CICLOS + 1

C Verifique se ainda existem COLUNAS não atribuído 'muitos', isto é,
se o
C Número de colunas não atribuído É MAIOR DO QUE
C DO THRESH PARAMETER. NÃO SE, REPLACE lista atual com a lista NEW,
C e ir para outro ciclo. Caso contrário, se EPSILON> 1, REDUZIR
EPSILON,
C Redefinir a atribuição ao vazio e REINICIAR LEILÃO;
C IF EPSILON = 1 finalizar.
C também aumentam a LICITAÇÃO MINIMAL INCREMENTO até um máximo
C VALOR DO EPSILON (esta é a característica adaptativa)


        INCR = INCR * INCRFACTOR
        IF (INCR.GT.EPSILON) INCR = EPSILON
        IF (RNONEWLIST.GT.THRESH) THEN
             IF (CHAVE) THEN
               CHAVE = .FALSE.
               RNOLIST = RNONEWLIST
               IR PARA 15
             END IF
             IF (RNONEWLIST.LT.RNOLIST) READY_SWITCH = .TRUE.
             IF ((RNONEWLIST.EQ.RNOLIST) .E. (READY_SWITCH)) THEN
               READY_SWITCH = .FALSE.
               IR PARA 15
             MAIS
               RNOLIST = RNONEWLIST
               IR PARA 115
             END IF
        MAIS
            IR PARA 300
        END IF


C ********************************************** **************
C
C FIM DE subproblema (escalonamento FASE)
C
C ********************************************** **************


300 CONTINUAR

C ****** SE EPSILON É uma RESCINDIR ******
```

```
        IF (EPSILON.EQ.1) THEN
           RETURN
        MAIS

C MAIS REDUZIR EPSILON e atualizar os parâmetros para a FASE DE ESCALA
PRÓXIMO

           NUMPHASES + 1 = NUMPHASES
           EPSILON = INT (EPSILON / FACTOR)
           IF (EPSILON.GT.INCR) EPSILON = INT (EPSILON / FACTOR)
           IF ((EPSILON.LT.1) .OR. (EPSILON.LT.ENDEPS)) EPSILON = 1
           IF (STARTINCR.LT.EPSILON) STARTINCR = FACTOR * STARTINCR
           THRESH = INT (THRESH / FACTOR)

X PRINT *, '*** FIM DE UMA FASE DE ESCALA; NEW EPSILON = ', EPSILON

           IR PARA 12
         END IF

         FIM



      SUBROUTINE SETRAN (ISEED)
      IMPLICIT real * 8 (AH, OZ), Integer * 4 (IN)
C ************************************************
**********************
C PORTABLE congruential (uniforme) RANDOM gerador de números:
NEXT_VALUE C = [(7 ** 5) * PREVIOUS_VALUE] MODULO [(2 ** 31) -1]
C
C Este gerador é composto por duas ROTINAS:
C (1) SETRAN - inicializa constantes e SEED
C (2) RRAN - gera um número aleatório REAIS
C
C O GERADOR DE RODA uma máquina com pelo menos 32 bits de precisão.
C DA SEED (ISEED) deve estar no intervalo (1, (2 ** 31 -1)).
C ************************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      IF (ISEED.LT.1) PARADA 77
      MULT = 16807
      MODUL = 2147483647
      I15 = 2 ** 15
      I16 = 2 ** 16
      JRAN = ISEED
      RETURN
      FIM
C
      RAN função real ()
      IMPLICIT real * 4 (AH, OZ), Integer * 4 (IN)
C ************************************************
**********************
C RAN gera um número aleatório real entre 0 e 1
C ************************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      Ixhi = JRAN / I16
      IXLO = JRAN-Ixhi * I16
      IXALO = IXLO * MULT
      LEFTLO = IXALO / I16
```

```
      IXAHI = Ixhi * MULT
      IFULHI = IXAHI + LEFTLO
      IRTLO = IXALO-LEFTLO * I16
      IOVER = IFULHI / I15
      IRTHI = IFULHI-IOVER * I15
      JRAN = ((IRTLO-MODUL) + IRTHI * I16) + IOVER
      IF (JRAN.LT.0) JRAN = JRAN + MODUL
      RAN = FLOAT (JRAN) / FLOAT (MODUL)
      RETURN
      FIM



**************************************************
************************

NAUCTION_SP: Combinado
Leilão Naive e Sequential Shortest Path Código

**************************************************
************************

Este código implementa o
método caminho mais curto seqüencial para o problema de atribuição,
precedido por um
extensiva a inicialização usando o algoritmo leilão ingênuo (cf. \
Seção
1.2.4). O código é muito semelhante em estrutura e desempenho para uma
código do autor [Ber81] e ao código de [JoV86] e [JoV87].
Estes códigos também combinou uma inicialização leilão ingênuo com o
método caminho mais curto seqüencial.

C **************************************************
******************
C
C COMBINADO LEILÃO NAIVE e seqüencial MENOR método do caminho
C PARA SIMÉTRICA nXn problemas de atribuição
C ENCONTRA uma alocação ótima
C
C ESCRITO POR DIMITRI P. Bertsekas
C
C **************************************************
******************
C
C usuário deve fornecer os seguintes dados PROBLEMA
CN = número de linhas e número de colunas
CA = número de arcos
C FOUT (ROW) = 1° ARC QUE SAI DE ROW
C COST (ARC) = CUSTO DE ARC
C END (ARC) = COLUNA correspondente ao arco
C
(.) C FOUT É UM N - comprimento da matriz
C COST (.), END (.), NXTEND (.) São matrizes de comprimento A
C DO IDEAL CESSÃO está contido no ASSIGN Array (.) ONDE
C ASSIGN (COL) é a linha ATRIBUÍDO de Col
C Este algoritmo não verifica inviabilidade DO PROBLEMA
C TO garantir que o problema é viável o usuário pode ADD
C ARCS ADICIONAL custo muito elevado
C
C Este código permite ARCS múltipla entre uma linha e uma coluna.
```

```
C
C ********************************************** ****************
C ALL problema de dados são integer
C ********************************************** ****************
C
        Implícita INTEGER (AZ)
        INTEGER AUCTNUM, A, STARC, ENDARC, Currow, ARC, COUNT
        INTEGER HCOUNT, ROW, FSTARC, FSTCOL, SNDARC, SNDCOL, TMAX,
BSTCOL
        INTEGER TMARG, TA, ROWPR, OLMARG, PRICE, TPRICE, CURCOL,
MANEQUIM
        INTEGER FOUT (6000), PCOL (6000), PROW (6000), MARG (6000)
        INTEGER ASSIGN (6000), COLLAB (6000), ROWLAB (6000)
        LISTA inteiro (6000), SCAN (6000)
        CUSTO INTEIRO (70000), END (70000)
        MAXSET LÓGICO
        THRESH REAL, RAND, TT, temporizador, TCOST

C
C ********************************************** ***************
C
C problema de geração de código começa AQUI
C o usuário pode substituir este código com um código que lê
C HIS / seu problema de um arquivo
C
C ********************************************** ***************
C
C gerador aleatório STUFF
C
      INTEGER MULT, MODUL, I15, I16, JRAN, ISEED
      RAN REAIS
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
C UNIFORME gerador de números aleatórios que retorna um em valor (0,1)
C
C INITIALIZE Gerador Aleatório
C
      ISEED = 13502460
      CHAME SETRAN (ISEED)

      PRINT * ', gerando problema de atribuição de SIMÉTRICA "
      PRINT *, "****************************************** '

C **** LER O número de linhas N & o número de arcos A ****

      PRINT *, 'Digite o número de linhas (e colunas)'
      LEIA *, N
5 PRINT *, 'Digite o número de arcos por ROW (1>) "
      LEIA *, NA
      IF (NA.LT.2) GOTO 5
      PRINT *, 'Digite o mínimo eo custo máximo'
      LEIA *, MINCOST, MAXCOST

C o número de arcos é n * NA

      A = N * NA

C Os arcos incidente a linha i SÃO FOUT (I) PARA FOUT (I + 1) -1
C PARA VIABILIDADE EACH ROW está diretamente ligado
C com a coluna correspondente

      FAZER 20 I = 1, N
```

```
        FOUT (I) = 1 + (i-1) * NA
20 CONTINUAR

C **** gerar a END (IA) eo custo (IA), que são os COLUNA
C eo coeficiente custo associado a ARC IA ****

      FAZER 25 IA = 1, A
        END (IA) = 1 + RAN () * N
        IF ((END (IA) .GT.N) .OR. (END (IA) .LT.1)) THEN
          PRINT *, 'Erro no problema de geração de'
          PAUSA
          PARE
        END IF
        COST (IA) = MINCOST + RAN () * (MAXCOST-MINCOST)
25 CONTINUAR

C MODIFICAR O final da última ARC OUT de cada linha de viabilidade
C e defina seu CUSTO PARA -MAXCOST

      FAZER 30 I = 1, N
        END (FOUT (I + 1 -1)) = I
        CUSTO (FOUT (I + 1) -1) = - MAXCOST
30 CONTINUAR

C
C ********************************************** ***************
C
C PROBLEMA geração de código TERMINA AQUI
C
C ********************************************** ***************
C

C algoritmo principal COMEÇA AQUI

C
C ALGORITMO INTERNAMENTE trata o problema como um
C problema de minimização.
C Para resolver um problema de atribuição de maximização, SET
C o seguinte parâmetro MAXSET para true, defina-o mais para FALSE

        MAXSET = .TRUE.

        IF (MAXSET) THEN
          FAZER 35 IA = 1, A
            COST (IA) = - COST (IA)
35 CONTINUAR
        END IF


        PRINT *, 'NO de linhas (e colunas):', N
        PRINT *, 'NO DE ARCOS:', A
        PRINT *, "COMBINADO LEILÃO ingênuo e SEQ. SH. PATH MÉTODO '
        PRINT *, '********************************************* '
        TIMER = Long (362)

        ISIMPL = 0
        IPRC = 0
        ISMALL = -20000000
        ILARGE = -ISMALL
        COUNT = 0
        HCOUNT = 0
```

```
        FOUT (N + 1) = A + 1

C O SEGUINTE INITIALIZATION ATÉ AO INÍCIO DO
C CICLOS LEILÃO NAIVE foi sugerido por JONKER E Volgenant

C REINICIAR PREÇOS COLUNA

        DO 105 J = 1, N
           PCOL (J) = ILARGE
105 CONTINUAR

C encontrar o melhor linha para cada coluna

          FAZER 120 I = 1, N
            PROW (I) = 0
            ROWLAB (I) = 0
            FSTARC = FOUT (I)
            LSTARC FOUT = (I + 1) -1
            FAZER 110 ARC = FSTARC, LSTARC
              J = END (ARC)
              IF (COST (ARC) .LT.PCOL (J)) THEN
                PCOL (J) = CUSTO (ARC)
                ASSIGN (J) = I
              END IF
110 CONTINUAR
120 CONTINUAR

LINHAS C CESSÃO CONSTRUCT linha e DEASSIGN MULTIPLY ATRIBUIÇÃO

        FAZER 130 J = 1, N
           J0 = N-J + 1
           I = ASSIGN (J0)
           IF (ROWLAB (I) .NE.0) THEN
             ROWLAB (I) = - ABS (ROWLAB (I))
             ASSIGN (J0) = 0
           MAIS
             ROWLAB (I) = J0
           END IF
130 CONTINUAR

C CONSTRUIR lista candidata E PREÇOS coluna correta


        NEWNOL = 0
        FAZER 150 I = 1, N
          CURCOL = ROWLAB (I)
          IF (CURCOL.EQ.0) THEN
            NEWNOL = + 1 NEWNOL
            LISTA (NEWNOL) = I
            GOTO 150
          END IF
          IF (CURCOL.LT.0) THEN
            ROWLAB (I) = - CURCOL
          MAIS
            MAX2 = ISMALL
            FSTARC = FOUT (I)
            LSTARC FOUT = (I + 1) -1
            FAZER 140 ARC = FSTARC, LSTARC
              J = END (ARC)
              IF (J.NE.CURCOL) THEN
                IF (PCOL (J) -Custo (ARC) .GT.MAX2) THEN
```

```
                  MAX2 = PCOL (J) -Custo (ARC)
               END IF
             END IF
140 CONTINUAR
            PROW (I) = MAX2
            PCOL (CURCOL) = PCOL (CURCOL) + MAX2
          END IF
150 CONTINUAR


        IF (NEWNOL.EQ.0) IR PARA 2000

C ********************************************** ***********

C END da inicialização; FAÇA UM NÚMERO DE CICLOS LEILÃO
C que é definido abaixo na AUCTNUM informação com base no
C sparsity DO PROBLEMA

        IF (N * N.LT.10 * A) = 2 AUCTNUM
        IF ((N * N.GE.10 * A) .E. (N * N.LT.25 * A)) AUCTNUM = 3
        IF ((N * N.GE.25 * A) .E. (N * N.LT.50 * A)) AUCTNUM = 4
        IF ((N * N.GE.50 * A) .E. (N * N.LT.100 * A)) AUCTNUM = 5
        IF (N * N.GE.100 * A) AUCTNUM = 6

C para executar um PURE SEQÜENCIAL Shortest Path método set
C = 0 AUCTNUM

        IF (AUCTNUM.EQ.0) GOTO 1000

C INÍCIO DE UM NOVO CICLO

80 NOLIST = NEWNOL
        I = 1
               NEWNOL = 0

C retirar um fila para a iteração

100 ROW = LIST (I)
        I = I + 1

        FSTARC = FOUT (ROW)
        LSTARC = FOUT (ROW + 1) -1
        BSTCOL = END (FSTARC)
        IF (FSTARC.EQ.LSTARC) THEN
           OLDROW = ASSIGN (BSTCOL)
           ASSIGN (BSTCOL) = ROW
           ROWLAB (ROW) = BSTCOL
           PROW (ROW) = ISMALL
           PCOL (BSTCOL) = ISMALL + COST (FSTARC)
           IF (OLDROW.GT.0) THEN
             I = I-1
             LISTA (I) = OLDROW
             ROWLAB (OLDROW) = 0
           END IF
           ISIMPL = + 1 ISIMPL
           Ir para 100
        END IF
        SNDARC = + 1 FSTARC
        SNDCOL = END (SNDARC)
        Max1 = PCOL (BSTCOL) -Custo (FSTARC)
        MAX2 = PCOL (SNDCOL) -Custo (SNDARC)
```

```
          IF (MAX1.LT.MAX2) THEN
             TMAX = Max1
             Max1 = MAX2
             MAX2 = TMAX
             FSTCOL = BSTCOL
             BSTCOL = SNDCOL
             SNDCOL = FSTCOL
          END IF
          IF (SNDARC.LT.LSTARC) THEN
           TRDARC = + 1 SNDARC
           FAZER 108 ARC = TRDARC, LSTARC
             CURCOL = END (ARC)
             TMAX = PCOL (CURCOL) -Custo (ARC)
             IF (TMAX.GT.MAX2) THEN
               IF (TMAX.GT.MAX1) THEN
                  MAX2 = Max1
                  Max1 = TMAX
                  SNDCOL = BSTCOL
                  BSTCOL = CURCOL
               MAIS
                  MAX2 = TMAX
                  SNDCOL = CURCOL
               END IF
             END IF
  108 CONTINUAR
          END IF
          PROW (ROW) = MAX2
          OLDROW = ASSIGN (BSTCOL)
          INCR = Max1-MAX2
          IF (INCR.GT.0) THEN
            PCOL (BSTCOL) = PCOL (BSTCOL) -INCR
            ASSIGN (BSTCOL) = ROW
            ROWLAB (ROW) = BSTCOL
            IF (OLDROW.GT.0) THEN
               I = I-1
               LISTA (I) = OLDROW
               ROWLAB (OLDROW) = 0
               ISIMPL = + 1 ISIMPL
               GOTO 100
            END IF
          MAIS
            IF (OLDROW.GT.0) THEN
              BSTCOL = SNDCOL
              OLDROW = ASSIGN (BSTCOL)
            END IF
            IF (OLDROW.EQ.0) THEN
              ASSIGN (BSTCOL) = ROW
              ROWLAB (ROW) = BSTCOL
            MAIS
              NEWNOL = + 1 NEWNOL
              LIST (NEWNOL) = ROW
            END IF
          END IF
          ISIMPL = + 1 ISIMPL
          IF (I.LE.NOLIST) GOTO 100

  C fim de um ciclo LEILÃO NAIVE

          COUNT = COUNT + 1

          IF (NEWNOL.EQ.0) THEN
```

```
          NASSIH = NEWNOL
          GOTO 2000
        END IF
        IF (COUNT.LT.AUCTNUM) GOTO 80



C ********************************************** ***************
C
C END ******* DO LEILÃO NAIVE PARTE *******
C
C
C ******* INÍCIO DE SEQ. SH. PATH MÉTODO *******
C
C ********************************************** ***************



1000 NASSIH = NEWNOL
X = 0 IHPRC
1020 FAZER 180 I = 1, NEWNOL
           SCAN (I) = LIST (I)
180 CONTINUAR
        DO 190 J = 1, N
           MARG (J) = 1
190 CONTINUAR
        NOSCAN = NEWNOL
        IL1 = 1
        IL2 = NOSCAN
1030 DO 1060 I = IL1, IL2
           Currow = SCAN (I)
           ROWPR = PROW (Currow)
           FSTARC = FOUT (Currow)
           LSTARC = FOUT (Currow + 1) -1
           FAZER 1050 ARC = FSTARC, LSTARC
                CURCOL = END (ARC)
                OLMARG = MARG (CURCOL)
                IF (OLMARG.EQ.0) IR PARA 1050
                TMARG = PCOL (CURCOL) -ROWPR-COST (ARC)
           IF (TMARG.EQ.0) THEN
             IF (ASSIGN (CURCOL) .EQ.0) THEN
                IR PARA 1500

C Faça AUGMENTATION

             MAIS
                MARG (CURCOL) = 0
                NOSCAN = NOSCAN + 1
                SCAN (NOSCAN) = ASSIGN (CURCOL)
                COLLAB (CURCOL) = Currow
             END IF
           MAIS
             IF ((OLMARG.GT.0) .OR. (TMARG.GT.OLMARG)) THEN
               MARG (CURCOL) = TMARG
               COLLAB (CURCOL) = Currow
             END IF
           END IF
1050 CONTINUAR
1060 CONTINUAR
```

```
C atual de digitalização completa das fases; Cheque de mais linhas
para verificar

        IF (IL2.LT.NOSCAN) THEN
           IL1 IL2 + 1 =
           IL2 = NOSCAN
           IR PARA 1030
        END IF


C Preço Variação


        IPRC = IPRC + 1
X = IHPRC IHPRC + 1
        INCR = ISMALL
        FAZER 200 J = 1, N
           TMARG = MARG (J)
           IF ((TMARG.LT.0) .E. (TMARG.GT.INCR)) INCR = TMARG
200 CONTINUAR
        FAZER 210 I = 1, NOSCAN
           PROW (SCAN (I)) = PROW (SCAN (I)) + INCR
210 CONTINUAR
        DO 220 J = 1, N
           IF (MARG (J) .EQ.0) PCOL (J) = PCOL (J) + INCR
220 CONTINUAR
        DO 1400 J = 1, N
           IF (MARG (J) .GE.0) IR PARA 1400
            MARG (J) = MARG (J) -INCR
              IF (MARG (J) .EQ.0) THEN
                IF (ASSIGN (J) .EQ.0) THEN
                  CURCOL = J
                  Currow = COLLAB (CURCOL)

C Faça AUGMENTATION

                  IR PARA 1500
                MAIS
                  NOSCAN = NOSCAN + 1
                  SCAN (NOSCAN) = ASSIGN (J)
                END IF
            END IF
1400 CONTINUAR
        IL1 IL2 + 1 =
        IL2 = NOSCAN
        IR PARA 1030

C PREÇO END CHANGE


C AUGMENTATION COMEÇA AQUI

1500 IF (ROWLAB (Currow) .EQ.0) THEN
          ASSIGN (CURCOL) = Currow
          ROWLAB (Currow) = CURCOL
          IR PARA 1700
        END IF
        TA = ROWLAB (Currow)
        ASSIGN (CURCOL) = Currow
        ROWLAB (Currow) = CURCOL
        CURCOL = TA
```

```
        Currow = COLLAB (CURCOL)
        IR PARA 1500
1700 IF (NEWNOL.EQ.1) IR PARA 2000

        FAZER 240 I = 1, NEWNOL-1
           IF (LIST (I) .EQ.CURROW) THEN
             FAZER 250 K = I + 1, NEWNOL
                LIST (KI) = LIST (K)
250 CONTINUAR
             DO 260 K = 1, I-1
                LIST (NEWNOL-I + K) = SCAN (K)
260 CONTINUAR
             NEWNOL = NEWNOL-1
             IR PARA 1020
           END IF

240 CONTINUAR

        NEWNOL = NEWNOL-1
        IR PARA 1020

C AUGMENTATION END


C ********************************************** ******
C
C rotina de saída. Cheques para Optimality DA SOLUÇÃO
C calcula o custo ótimo, e compila SOLUÇÃO
C ESTATÍSTICAS. O utilizador pode substituir este pelo código que
C ESCREVE AT o dispositivo apropriado a solução ideal
C CONTIDAS NO ASSIGN Array (.).
C
C ********************************************** ******

2000 CONTINUAR

        TT = (LONG (362) - TIMER) / 60

        PRINT *, 'TIME TOTAL =', TT, 's.'
        PRINT *, 'ITERATIONS LEILÃO NO DE ingênuo:', ISIMPL
        PRINT *, 'NO DE SH. ITERATIONS caminho: ', NASSIH
X PRINT *, 'NO das variações de preços:', IHPRC

C verificação de viabilidade de uma solução e calcular o custo

        FAZER 265 I = 1, N
           ROWLAB (I) = 0
265 CONTINUAR

        NOASS = 0
        FAZER 280 J = 1, N
           IF (ASSIGN (J) .GT.0) THEN
             ROWLAB (ASSIGN (J)) = J
             NOASS = + 1 NOASS
           END IF
280 CONTINUAR

        IF (NOASS.LT.N) THEN
          PRINT *, "o número de colunas atribuído é", NOASS, '(muito
pequeno)'
        END IF
```

```
        TCOST = 0
        FAZER 300 I = 1, N
          J = ROWLAB (I)
          IF (J.EQ.0) THEN
            PRINT *, 'ROW', I, 'não é atribuído "
          END IF
          FSTARC = FOUT (I)
          LSTARC FOUT = (I + 1) -1
          MINCOST = ILARGE
          FAZER 310 ARC = FSTARC, LSTARC
           CURCOL = END (ARC)
           TMARG = PROW (I) -PCOL (CURCOL) + COST (ARC)
           IF (TMARG.LT.0) THEN
             PRINT *, 'COMPL. Frouxidão VIOLAÇÃO: ROW ', I,' COLUMN ',
CURCOL
           END IF
           IF (CURCOL.EQ.J) THEN
             IF (MINCOST.GT.COST (ARC)) THEN
                MINCOST = CUSTO (ARC)
                ROWMARG = TMARG
             END IF
           END IF
310 CONTINUAR
          IF (ROWMARG.NE.0) THEN
            PRINT *, 'COMPL. Folga. VIOLAÇÃO AT ATRIBUÍDO ARC DE ROW
', I
          END IF
          IF (MAXSET) THEN
            TCOST = TCOST-MINCOST
          MAIS
            TCOST = TCOST + MINCOST
          END IF
300 CONTINUAR

        ESCREVA (9,2100) TCOST
2100 FORMATO ('atribuição de custo =', F14.2)

        PRINT *, '****************************************** '
        PRINT *, 'programa terminou; PRESS <CR> '

        PAUSA
        PARE

        FIM



      SUBROUTINE SETRAN (ISEED)
      IMPLICIT real * 8 (AH, OZ), Integer * 4 (IN)
C ********************************************
**********************
C PORTABLE congruential (uniforme) RANDOM gerador de números:
NEXT_VALUE C = [(7 ** 5) * PREVIOUS_VALUE] MODULO [(2 ** 31) -1]
C
C Este gerador é composto por duas ROTINAS:
C (1) SETRAN - inicializa constantes e SEED
C (2) RRAN - gera um número aleatório REAIS
C
C O GERADOR DE RODA uma máquina com pelo menos 32 bits de precisão.
C DA SEED (ISEED) deve estar no intervalo (1, (2 ** 31 -1)).
```

```
C ***********************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      IF (ISEED.LT.1) PARADA 77
      MULT = 16807
      MODUL = 2147483647
      I15 = 2 ** 15
      I16 = 2 ** 16
      JRAN = ISEED
      RETURN
      FIM
C
      RAN função real ()
      IMPLICIT real * 4 (AH, OZ), Integer * 4 (IN)
C ***********************************************
**********************
C RAN gera um número aleatório real entre 0 e 1
C ***********************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      Ixhi = JRAN / I16
      IXLO = JRAN-Ixhi * I16
      IXALO = IXLO * MULT
      LEFTLO = IXALO / I16
      IXAHI = Ixhi * MULT
      IFULHI = IXAHI + LEFTLO
      IRTLO = IXALO-LEFTLO * I16
      IOVER = IFULHI / I15
      IRTHI = IFULHI-IOVER * I15
      JRAN = ((IRTLO-MODUL) + IRTHI * I16) + IOVER
      IF (JRAN.LT.0) JRAN = JRAN + MODUL
      RAN = FLOAT (JRAN) / FLOAT (MODUL)
      RETURN
      FIM
```

```
************************************************
************************
```

Códigos Max-Flow

```
************************************************
************************
```

FORD-Fulkerson

Este código implementa o método de Ford-Fulkerson para
resolver o problema de fluxo máximo (cf. \ Secção 1.2.2).

```
AMOSTRA C ******* programa de chamada para a Ford-Fulkerson *******

      PARÂMETROS (maxNodes = 10000, MAXARCS = 40000)
      Implícita INTEGER (AZ)
      COMUM / escalares / N, NA, GRANDE, SOURCE, pia
      COMUM / STATS / Niter
      COMUM / UBOUND / U
      COMUM / FLUXO / F
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
```

```
      COMUM / BLK3 / PRDCSR
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      COMUM / Etiquetas / LABEL
      COMUM / MARCAS / MARK

      INTEGER STARTN (MAXARCS)
      INTEGER ENDN (MAXARCS)
      INTEGER U (MAXARCS)
      INTEGER F (MAXARCS)
      INTEGER PRDCSR (maxNodes)
      INTEGER FIN (maxNodes)
      INTEGER FOUT (maxNodes)
      INTEGER NXTIN (MAXARCS)
      INTEGER NXTOU (MAXARCS)
      INTEGER LABEL (maxNodes)
      MARK LÓGICO (maxNodes)
      Real * 8 TT, TIMER


      PRINT *, 'FORD-Fulkerson MÉTODO DE MAX-FLOW'
      PRINT *, '**********************************'
      PRINT *, 'READING problema de dados'
      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' velho ')
      Rewind (13)

      GRANDE = 500000000


C LEIA número de nós e arcos

      LER (13,1010) N, NA

C LEIA início, fim, custo e capacidade de cada ARC

      FAZER 20 I = 1, NA
        READ (13,1020) STARTN (I), ENDN (I), MANEQUIM, U (I)
20 CONTINUAR

      ENDFILE (13)
      Rewind (13)

1000 FORMATO (1I8)
1010 FORMATO (2I8)
1020 FORMATO (4I8)


      PRINT *, "o número de nós IS = ', N

41 PRINT *, "ENTER o nó SUPERSOURCE '
      LER *, SOURCE
      IF ((SOURCE.LE.0) .OR. (SOURCE.GT.N)) GOTO 41

42 PRINT *, "ENTER o nó SUPERSINK '
      LER *, pia

      IF ((SINK.LE.0) .OR. (SINK.GT.N)) GOTO 42
      IF (SINK.EQ.R) GOTO 42
```

```
C DO fonte e dreno NÓS NÃO SERÁ Iterated ON



      PRINT *, "reestruturar os dados '
      CHAMADA INIDAT


C SET flui para ZERO

      FAZER 50 ARC = 1, NA
        F (ARC) = 0
50 CONTINUAR

      PRINT *, '********************************'
      TIMER = Long (362)
      CHAMADA FORD_FULK
      TT = FLOAT (LONG (362) - TIMER) / 60
      PRINT *, 'TIME TOTAL =', TT, 's.'

C ***** COMPUTE MAX-FLOW *****

      MAX_FLOW = 0
      FAZER 60 ARC = 1, NA
        IF (STARTN (ARC) .EQ.SOURCE) MAX_FLOW = MAX_FLOW + F (ARC)
60 CONTINUAR
      MAX_FLOW2 = 0
      FAZER 70 ARC = 1, NA
        IF (ENDN (ARC) .EQ.SINK) MAX_FLOW2 = MAX_FLOW2 + F (ARC)
70 CONTINUAR

      PRINT *, '# de iterações =', Niter
      PRINT *, "MAX-FLOW = ', MAX_FLOW2
      PRINT *, '********************************'

      IF (MAXFLOW.NE.MAXFLOW2) THEN
         PRINT *, 'fonte de fluxo de não igual a AFUNDE fluxo "
      ENDIF
      PRINT *, 'programa terminou; PRESS <CR> PARA SAIR '

      PAUSA
      FIM




C ********************************************** ***
      SUBROUTINE INIDAT
C Esta sub-rotina USAM O STARTN matrizes de dados E ENDN
C PARA CONSTRUIR AUXILIAR DE DADOS ARRAYS FOUT, NXTOU, FIN, E
C NXTIN. Nisto SUBROUTINE
C arbitrariamente ENCOMENDAR AS ARCS DEIXANDO cada nó e STORE
C esta informação em FOUT E NXTOU. Da mesma forma, arbitragem
C RILLY ORDER os arcos que entram em cada nó e armazenar esta
C INFORMAÇÃO EM FIN E NXTIN. NA CONCLUSÃO DO
C CONSTRUÇÃO, TEMOS
C
C FOUT (I) = Primeiro ARC DEIXANDO NODE I.
C NXTOU (J) = PRÓXIMO ARC saem do nó CHEFE DE ARC J.
C FIN (I) = Primeiro ARC Entrando no passo I.
C NXTIN (J) = PRÓXIMO ARC ENTRANDO NO NÓ DE CAUDA ARC J.
```

```
      PARÂMETROS (maxNodes = 10000, MAXARCS = 40000)
      Implícita INTEGER (AZ)
      Common / escalares / N, NA, GRANDE
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU

      INTEGER STARTN (MAXARCS)
      INTEGER ENDN (MAXARCS)
      INTEGER FIN (maxNodes)
      INTEGER FOUT (maxNodes)
      INTEGER NXTIN (MAXARCS)
      INTEGER NXTOU (MAXARCS)
      INTEGER FINALIN (maxNodes)
      INTEGER FINALOU (maxNodes)

      FAZER 20 NODE = 1, N
        FIN (nó) = 0
        FOUT (nó) = 0
        FINALIN (nó) = 0
        FINALOU (nó) = 0
20 CONTINUAR

      Fazer 30 ARC = 1, NA
        START = STARTN (ARC)
        END = ENDN (ARC)
        IF (FOUT (START) .NE.0) THEN
          NXTOU (FINALOU (START)) = ARC
        MAIS
          FOUT (START) = ARC
        END IF
        IF (FIN (END) .NE.0) THEN
          NXTIN (FINALIN (END)) = ARC
        MAIS
          FIN (END) = ARC
        END IF
        FINALOU (START) = ARC
        FINALIN (END) = ARC
        NXTIN (ARC) = 0
        NXTOU (ARC) = 0
30 CONTINUAR
C
      RETURN
      FIM



C ********************************************** **************
C
C BASIC FORD-Fulkerson MÉTODO DE MAX-FLOW.
C
C ********************************************** **************

      FORD_FULK SUBROUTINE
      Implícita INTEGER (AZ)
      COMUM / escalares / N, NA, GRANDE, SOURCE, pia
      COMUM / STATS / Niter
```

```
      COMUM / UBOUND / U
      COMUM / FLUXO / F
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / BLK3 / PRDCSR
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      COMUM / MARCAS / MARK
      COMUM / Etiquetas / LABEL

      INTEGER STARTN (1), ENDN (1), L (1), F (1), FIN (1), FOUT (1)
      INTEGER NXTIN (1), NXTOU (1), PRDCSR (1), na etiqueta (1)
      MARK LÓGICO (1)


      Niter = 0

      FAZER 10 I = 1, N
        MARK (I) =. FALSE.
10 CONTINUAR


C INÍCIO DE nova iteração

15 = 1 NLABEL
      NSCAN = 1
      MARK (SOURCE) =. TRUE.
      LABEL (1) = SOURCE


20 CONTINUAR

C SCAN um nó NOVO

      NÓ = LABEL (NSCAN)

C SCAN OUTGOING ARCS do nodo

      ARC = FOUT (nó)
30 IF (ARC.GT.0) THEN
         NODE2 = ENDN (ARC)
         IF ((.NOT.MARK (NODE2)). E. (F (ARC) .LT.U (ARC))) THEN
           PRDCSR (NODE2) = ARC
           IF (NODE2.EQ.SINK) THEN
             CHAMADA AUGMENT
             Niter = Niter + 1
             DO 40 I = 1, NLABEL
               MARK (LABEL (I)) =. FALSE.
40 CONTINUAR
             GOTO 15
           MAIS
             MARK (NODE2) =. TRUE.
             NLABEL = + 1 NLABEL
             LABEL (NLABEL) = NODE2
           END IF
         END IF
         ARC = NXTOU (ARC)
         GOTO 30
      END IF
```

```
C SCAN ENTRANTE ARCS do nodo

      ARC = FIN (nó)
50 IF (ARC.GT.0) THEN
         NODE2 = STARTN (ARC)
         IF ((.NOT.MARK (NODE2)). E. (F (ARC) .GT.0)) THEN
           PRDCSR (NODE2) = - ARC
           IF (NODE2.EQ.SINK) THEN
             CHAMADA AUGMENT
             Niter = Niter + 1
             FAZER 60 I = 1, NLABEL
               MARK (LABEL (I)) =. FALSE.
60 CONTINUAR
             GOTO 15
           MAIS
             MARK (NODE2) =. TRUE.
             NLABEL = + 1 NLABEL
             LABEL (NLABEL) = NODE2
           END IF
         END IF
         ARC = NXTIN (ARC)
         GOTO 50
      END IF

Check C para a rescisão; SCAN um nó NOVO

      IF (NSCAN.EQ.NLABEL) THEN
        RETURN
      END IF
      NSCAN = + 1 NSCAN
      GOTO 20

      FIM




      SUBROUTINE AUGMENT
      Implícita INTEGER (AZ)
      COMUM / escalares / N, NA, GRANDE, SOURCE, pia
      COMUM / UBOUND / U
      COMUM / FLUXO / F
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / BLK3 / PRDCSR

      INTEGER STARTN (1), ENDN (1), L (1), F (1), PRDCSR (1)


      DX = GRANDE
      CURNODE = SINK
10 IF (CURNODE.NE.SOURCE) THEN
        ARC = PRDCSR (CURNODE)
        IF (ARC.GT.0) THEN
          INCR = U (ARC) -F (ARC)
          IF (DX.GT.INCR) DX = INCR
          CURNODE = STARTN (ARC)
        MAIS
          ARC = -ARC
```

```
          INCR = F (ARC)
          IF (DX.GT.INCR) DX = INCR
          CURNODE = ENDN (ARC)
        END IF
        GOTO 10
      END IF

      CURNODE = SINK
20 IF (CURNODE.NE.SOURCE) THEN
        ARC = PRDCSR (CURNODE)
        IF (ARC.GT.0) THEN
          F (ARC) = F (ARC) + DX
          CURNODE = STARTN (ARC)
        MAIS
          ARC = -ARC
          F (ARC) = F (ARC) -DX
          CURNODE = ENDN (ARC)
        END IF
        GOTO 20
      END IF

      RETURN

      FIM



************************************************
************************

$ \ E $ -Relax-MF

************************************************
************************

Este código implementa o $ \ e $ -relaxation
método para o problema de fluxo máximo (cf. \ Seção 4.5). Aqui, $ \ e
= 1 $
em todo o algoritmo.

AMOSTRA C ******* CHAMADA PROGRAMA DE E-RELAX / MAX-FLOW *******
C
C esta versão usa um CYCLIC FILA, o caminho mais curto
C (em largura) INITIALIZATION
C e uma rotina que faz relabeling GLOBAL E TAMBÉM
C detecta um SATURATED CUT e termina EARLY
C TI em seguida, encontra um fluxo máximo utilizando um
"REVERSE'ALGORITHM
C
C Este código é uma versão um pouco diferentes do ONE
C no livro "REDE LINEAR OTIMIZAÇÃO: ALGORITMOS E
CÓDIGOS C, MIT Press, 1991, por Dimitri P. Bertsekas
C
C ********************************************** **
      PARÂMETROS (maxNodes = 20000, MAXARCS = 120000)
      Implícita INTEGER (AZ)
      COMUM / escalares / N, NA, GRANDE, SOURCE, pia
      Common / STATS / Niter, CUTEXIT, CUTSEARCH, CUTCOUNT, Naug
      COMUM / CHECK / CHECKTIME, TOT_TIME
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
```

```
      COMUM / UBOUND / U
      COMUM / FLUXO / F
      Common / PREÇOS / P
      COMUM / BLK3 / excedente
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      COMUM / BLK11 / FPUSHF
      COMUM / BLK12 / NXTPUSHF
      COMUM / BLK13 / FPUSHB
      COMUM / BLK14 / NXTPUSHB
      COMUM / Etiquetas / LABEL
      COMUM / MARCAS / MARK
      COMUM / Fila / NXTQUEUE
      COMUM / PRED / PRED

C gerador aleatório STUFF

      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      RAN REAIS

      INTEGER NXTQUEUE (maxNodes)
      INTEGER STARTN (MAXARCS)
      INTEGER ENDN (MAXARCS)
      INTEGER U (MAXARCS)
      INTEGER F (MAXARCS)
      INTEGER excedente (maxNodes)
      INTEGER FIN (maxNodes)
      INTEGER FOUT (maxNodes)
      INTEGER NXTIN (MAXARCS)
      INTEGER NXTOU (MAXARCS)
      INTEGER P (maxNodes)
      INTEGER FPUSHF (maxNodes)
      INTEGER NXTPUSHF (MAXARCS)
      INTEGER FPUSHB (maxNodes)
      INTEGER NXTPUSHB (MAXARCS)
      INTEGER LABEL (maxNodes)
      INTEGER PRED (maxNodes)
      MARK LÓGICO (maxNodes)
      Real * 8 TT, TOT_TIME, temporizador, TOTAL, TOTAL_C, CHECKTIME


      PRINT *, «método E-relaxamento para MAX-FLOW '
      PRINT *, 'usa FIFO fila e relabeling GLOBAL'
      PRINT *, "****************************************** '

      GRANDES = 1000000000


C ********************************************** ***********
C
C Leia os dados do problema
C
C O problema é especificado pelo variáveis a seguir:
C
CN: O número de nós
C NA: o número de arcos
C FONTE: o nó origem do problema de fluxo máximo
C Pia: o sink DO PROBLEMA MAX-FLOW
C STARTN (ARC): o nó INÍCIO DE ARC
```

```
C ENDN (ARC): o nó FIM DO ARCO
CU (ARC): a capacidade da ARC (IT for alterado posteriormente à
capacidade residual)
C
C
      PRINT *, 'READING problema de dados'

      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' velho ')
      Rewind (13)

C LEIA número de nós e arcos

      LEIA (13, *) N, NA

      FAZER 5 I = 1, NA
        LEIA (13, *) STARTN (I), ENDN (I), MANEQUIM, U (I)
5 CONTINUAR


      ENDFILE (13)
      Rewind (13)


      SOURCE = 1
      SINK = N

      PRINT *, "****************************************** '
      PRINT *, 'número de nós =', N ', número de arcos =', NA
C



C
C FIM DE LEITURA os dados do problema
C

C REINICIAR PREÇOS E EXCEDENTES

      FAZER 20 NODE = 1, N
        FPUSHF (nó) = 0
        FPUSHB (nó) = 0
        P (nó) = N
        EXCEDENTE (nó) = 0
20 CONTINUAR


C SET flui para limite superior OU limite inferior PARA SATISFAZER E-
CS

      FAZER 50 ARC = 1, NA
        IF (STARTN (ARC) .EQ.SOURCE) THEN
          F (ARC) = U (ARC)
          EXCEDENTE (ENDN (ARC)) = excedente (ENDN (ARC)) + F (ARC)
          EXCEDENTE (SOURCE) = excedente (SOURCE) -F (ARC)
        MAIS
          F (ARC) = 0
        END IF
50 CONTINUAR


C criam estruturas de dados necessários
```

```
      CHAMADA INIDAT


      TIMER = Long (362)
      CHAMADA E_RELAX_MF
      TT = FLOAT (LONG (362) - TIMER) / 60

      PRINT *, 'TIME TOTAL =', TT, 's'
      PRINT *, "tempo para encontrar um CUT MIN = ', TOT_TIME
      PRINT *, 'TIME PARA VERIFICAR SE A CUT MIN =', CHECKTIME

C ***** COMPUTE MAX-FLOW *****

      MAX_FLOW = 0
      ARC = FIN (PIA)
70 IF (ARC.GT.0) THEN
        MAX_FLOW = MAX_FLOW + F (ARC)
        ARC = NXTIN (ARC)
        GOTO 70
      END IF
      IF (MAX_FLOW.NE.SURPLUS (pia)) THEN
        PRINT *, 'ERRO NO CÁLCULO DO MAX-FLOW'
      END IF

      PRINT *, "MAX-FLOW = ', MAX_FLOW
      PRINT *, '# do fluxo de mudanças por ARC =', FLOAT (Naug) /
FLOAT (NA)
      PRINT *, '# dos aumentos de preços por nó =', FLOAT (Niter) /
FLOAT (N)
      PRINT *, '# de pesquisas para saturado CUT =', CUTCOUNT
      PRINT *, '******************************************** ** '

C ***** Verifique se todos os excedentes são ZERO ****

        FAZER 80 NODE = 1, N
          IF ((NODE.NE.SOURCE) .E. (NODE.NE.SINK)) THEN
            IF (EXCEDENTE (nó) .NE.0) THEN
              PRINT *, «excedente de NODE ', NODE,' é diferente de
zero '
            END IF
          END IF
80 CONTINUAR


      PRINT *, 'programa terminou; PRESS <CR> PARA SAIR '
      PAUSA

      FIM
```

```
C ********************************************** ***************
C
C BASIC E-relaxamento para MAX-FLOW.
C Esta versão usa uma fila para selecionar nós para fazer uma iteração
ON,
```

```
C E mantém listas PUSH. NÓS entrar na fila na parte inferior.
C esta versão usa um caminho mais curto (em largura) INITIALIZATION
C e uma rotina que detecta A CUT saturada e
C REALIZA relabeling GLOBAL
C
C Este código é uma versão um pouco diferentes do ONE
C no livro "REDE LINEAR OTIMIZAÇÃO: ALGORITMOS E
CÓDIGOS C, MIT Press, 1991, por Dimitri P. Bertsekas
C
C ********************************************** ***************

      E_RELAX_MF SUBROUTINE
      PARÂMETROS (maxNodes = 20000, MAXARCS = 120000)
      Implícita INTEGER (AZ)
      COMUM / escalares / N, NA, GRANDE, SOURCE, pia
      Common / STATS / Niter, CUTEXIT, CUTSEARCH, CUTCOUNT, Naug
      COMUM / CHECK / CHECKTIME, TOT_TIME
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / UBOUND / U
      COMUM / FLUXO / F
      Common / PREÇOS / P
      COMUM / BLK3 / excedente
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      COMUM / BLK11 / FPUSHF
      COMUM / BLK12 / NXTPUSHF
      COMUM / BLK13 / FPUSHB
      COMUM / BLK14 / NXTPUSHB
      COMUM / MARCAS / MARK
      COMUM / Etiquetas / LABEL
      COMUM / Fila / NXTQUEUE
      COMUM / PRED / PRED

      INTEGER NXTQUEUE (maxNodes)
      INTEGER STARTN (MAXARCS)
      INTEGER ENDN (MAXARCS)
      INTEGER U (MAXARCS)
      INTEGER F (MAXARCS)
      INTEGER excedente (maxNodes)
      INTEGER FIN (maxNodes)
      INTEGER FOUT (maxNodes)
      INTEGER NXTIN (MAXARCS)
      INTEGER NXTOU (MAXARCS)
      INTEGER P (maxNodes)
      INTEGER FPUSHF (maxNodes)
      INTEGER NXTPUSHF (MAXARCS)
      INTEGER FPUSHB (maxNodes)
      INTEGER NXTPUSHB (MAXARCS)
      INTEGER LABEL (maxNodes)
      INTEGER PRED (maxNodes)
      MARK LÓGICO (maxNodes)
      Real * 8 TEMP, TOT_TIME, TIMER1, CHECKTIME
      OUTFLAG lógico, relabel

      TIMER1 = longa (362)

C REINICIAR CONTAGEM DO NÚMERO DE CIMA ITERATIONS REALIZADAS
```

```
      Niter = 0
      Naug = 0
      COUNT = 0
      CUTCOUNT = 0
      CUTEXIT = 0
      THRESH = N
      CHECKTIME = 0
      TOT_TIME = 0


      Relabel = .TRUE.


C REINICIAR PREÇOS POR UM MÉTODO PESQUISA BOCA PRIMEIRA

        FAZER 51 NODE = 1, N
          MARK (nó) =. FALSE.
51 CONTINUAR

        MARK (SOURCE) =. TRUE.
        MARK (pia) =. TRUE.
        P (FONTE) = N
        P (pia) = 0

        NLABEL = 1
        LABEL (1) = SINK
        NSCAN = 0

55 CONTINUAR
        NSCAN = + 1 NSCAN
        NÓ = LABEL (NSCAN)
        PREÇO = P (nó) +1

        ARC = FIN (nó)
58 IF (ARC.GT.0) THEN
          START = STARTN (ARC)
          IF (.NOT.MARK (START)) THEN
            NLABEL = + 1 NLABEL
            LABEL (NLABEL) = INÍCIO
            P (START) = PREÇO
            MARK (START) =. TRUE.
          END IF
          ARC = NXTIN (ARC)
          GOTO 58
        END IF

        IF (NLABEL.GT.NSCAN) GOTO 55

C ***** ***** INITIALIZATION FILA

70 CONTINUAR

        FIRST = 0
        FAZER 82 I = 1, NLABEL
          NÓ = LABEL (I)
          IF ((NODE.NE.SOURCE) .E. (NODE.NE.SINK)) THEN
            IF (EXCEDENTE (nó) .GT.0) THEN
              IF (FIRST.EQ.0) THEN
                FIRST = NÓ
              MAIS
                NXTQUEUE (LAST) = NÓ
```

```
              END IF
              LAST = NÓ
            MAIS
              NXTQUEUE (nó) = 0
            END IF
          MAIS
            NXTQUEUE (nó) = - 1
          END IF
82 CONTINUAR
        NXTQUEUE (LAST) = PRIMEIRA

        IF (FIRST.EQ.0) RETURN

        I = PRIMEIRA
        PrevNode = LAST

C ********************************************** **********
C
C INICIAR do algoritmo MAIN.
C Execute ITERATIONS único nó até o término.
C
C ********************************************** **********

100 CONTINUAR

C passamos para o próximo nó (Nó I) para a iteração

        SURPI = excedente (I)
        PREÇO = P (I)

        IF ((SURPI .gt. 0) .E. (PRICE.LT.N)) THEN
C PRINT *, 'NODE', I, 'INÍCIO PREÇO =', PREÇO
           ARCF = FPUSHF (I)
           ARCB = FPUSHB (I)

C ********************** D - EMPURRA ************************

C começar por tentar afastar FLUXO EM ARCS que foram
C admissível a final da última iteração (SE HOUVER)
C neste nodo. WE deve verificar se eles ainda são admissíveis.

120 IF ((SURPI .gt. 0) .E. (ARCF .gt. 0)) THEN
                     RESID = L (ARCF) - F (ARCF)
                     J = ENDN (ARCF)
                    IF ((PREÇO-P (J) .EQ.1) .E. (RESID.GT.0)) THEN
                        Naug = Naug + 1
                        IF (SURPI .GE. RESID) THEN
                           F (ARCF) = L (ARCF)
                           SURPI = SURPI - RESID
                           EXCEDENTE (J) = excedente (J) + RESID
                           IF (NXTQUEUE (J) .EQ.0) THEN
                            IF (EXCEDENTE (J) .GT.0) THEN
                             NXTQUEUE (PrevNode) = J
                             NXTQUEUE (J) = I
                             PrevNode = J
                            END IF
                           END IF
                           ARCF = NXTPUSHF (ARCF)
                        MAIS
                           F (ARCF) = F (ARCF) + SURPI
                           EXCEDENTE (J) = excedente (J) + SURPI
```

```
                           IF (NXTQUEUE (J) .EQ.0) THEN
                            IF (EXCEDENTE (J) .GT.0) THEN
                             NXTQUEUE (PrevNode) = J
                             NXTQUEUE (J) = I
                             PrevNode = J
                            END IF
                           END IF
                           SURPI = 0
                        ENDIF
                     MAIS
                        ARCF = NXTPUSHF (ARCF)
                     ENDIF
                     GOTO 120
                  ENDIF
121 IF ((SURPI .gt. 0) .E. (ARCB .gt. 0)) THEN
                     J = STARTN (ARCB)
                    IF ((PREÇO-P (J) .EQ.1) .E (F (ARCB) .GT.0.)) THEN
                        Naug = Naug + 1
                        IF (SURPI .GE. F (ARCB)) THEN
                           SURPI = SURPI - F (ARCB)
                           EXCEDENTE (J) = excedente (J) + F (ARCB)
                           IF (NXTQUEUE (J) .EQ.0) THEN
                            IF (EXCEDENTE (J) .GT.0) THEN
                             NXTQUEUE (PrevNode) = J
                             NXTQUEUE (J) = I
                             PrevNode = J
                            END IF
                           END IF
                           F (ARCB) = 0
                           ARCB = NXTPUSHB (ARCB)
                        MAIS
                           F (ARCB) = F (ARCB) - SURPI
                           EXCEDENTE (J) = excedente (J) + SURPI
                           IF (NXTQUEUE (J) .EQ.0) THEN
                            IF (EXCEDENTE (J) .GT.0) THEN
                             NXTQUEUE (PrevNode) = J
                             NXTQUEUE (J) = I
                             PrevNode = J
                            END IF
                           END IF
                           SURPI = 0
                        ENDIF
                     MAIS
                        ARCB = NXTPUSHB (ARCB)
                     ENDIF
                     GOTO 121
                  ENDIF

C Agora entramos A REPEAT ... ATÉ tipo de círculo até que
C têm DO EXCESSO reduzida a zero.

C ********************* aumento de preço *********************

129 CONTINUAR

C Primeiro, tente A (POSSIVELMENTE DEGENERATE) aumento de preços.

130 IF ((ARCF .EQ. 0) .E. (ARCB .EQ. 0)) THEN
                     Niter = Niter + 1
                     PREÇO = GRANDE
                     ARC = FOUT (I)
```

```
131 SE (ARC .gt. 0) THEN
                        IF (F (ARC) .lt. U (ARC)) THEN
                           XP = P (ENDN (ARC)) + 1
                           IF (XP .lt. PREÇO) THEN
                              PREÇO = XP
                              ARCF = ARC
                              NXTPUSHF (ARC) = 0
                           MAIS
                            IF (XP .EQ. PREÇO) THEN
                              NXTPUSHF (ARC) = ARCF
                              ARCF = ARC
                            END IF
                           ENDIF
                        ENDIF
                        ARC = NXTOU (ARC)
                        GOTO 131
                     ENDIF
                     ARC = FIN (I)
132 SE (ARC .gt. 0) THEN
                        IF (F (ARC) .gt. 0) THEN
                           XP = P (STARTN (ARC)) + 1
                           IF (XP .lt. PREÇO) THEN
                              PREÇO = XP
                              ARCB = ARC
                              ARCF = 0
                              NXTPUSHB (ARC) = 0
                           MAIS
                            IF (XP .EQ. PREÇO) THEN
                              NXTPUSHB (ARC) = ARCB
                              ARCB = ARC
                            END IF
                           ENDIF
                        ENDIF
                        ARC = NXTIN (ARC)
                        GOTO 132
                     ENDIF

                  ENDIF

C ********************** D - EMPURRA ********************

C Se a etapa NÃO FOI degenerar, e tentar fazer alguma
C empurrar. NÓS NÃO reutilizar o código acima, porque IS IT
Não é necessário c para verificar se o ARCS utilizados são
admissíveis.

                  IF (SURPI .gt. 0) THEN
141 IF (ARCF .gt. 0) THEN
                        RESID = L (ARCF) - F (ARCF)
                        Naug = Naug + 1
                        J = ENDN (ARCF)
                        IF (SURPI .GE. RESID) THEN
                           F (ARCF) = L (ARCF)
                           SURPI = SURPI - RESID
                           EXCEDENTE (J) = excedente (J) + RESID
                           IF (NXTQUEUE (J) .EQ.0) THEN
                            IF (EXCEDENTE (J) .GT.0) THEN
                             NXTQUEUE (PrevNode) = J
                             NXTQUEUE (J) = I
                             PrevNode = J
                            END IF
```

```
                              END IF
                              ARCF = NXTPUSHF (ARCF)
                              IF (SURPI .gt. 0) GOTO 141
                           MAIS
                              F (ARCF) = F (ARCF) + SURPI
                              EXCEDENTE (J) = excedente (J) + SURPI
                              IF (NXTQUEUE (J) .EQ.0) THEN
                               IF (EXCEDENTE (J) .GT.0) THEN
                                NXTQUEUE (PrevNode) = J
                                NXTQUEUE (J) = I
                                PrevNode = J
                               END IF
                              END IF
                              SURPI = 0
                           ENDIF
                        END IF

142 IF ((SURPI .gt. 0) .E. (ARCB .gt. 0)) THEN
                           J = STARTN (ARCB)
                           Naug = Naug + 1
                           IF (SURPI .GE. F (ARCB)) THEN
                              SURPI = SURPI - F (ARCB)
                              EXCEDENTE (J) = excedente (J) + F (ARCB)
                              IF (NXTQUEUE (J) .EQ.0) THEN
                                NXTQUEUE (PrevNode) = J
                                NXTQUEUE (J) = I
                                PrevNode = J
                              END IF
                              F (ARCB) = 0
                              ARCB = NXTPUSHB (ARCB)
                           MAIS
                              F (ARCB) = F (ARCB) - SURPI
                              EXCEDENTE (J) = excedente (J) + SURPI
                              IF (NXTQUEUE (J) .EQ.0) THEN
                                NXTQUEUE (PrevNode) = J
                                NXTQUEUE (J) = I
                                PrevNode = J
                              END IF
                              SURPI = 0
                           ENDIF
                           GOTO 142
                        ENDIF

C nós fizemos todo o EMPURRAR WE CAN a esse preço
C LEVEL. Tentar fazer um (possivelmente DEGENERATE) PREÇO
C aumentar.

                        GOTO 129
                     ENDIF

C ESTE É O FIM DO ITERATION UP. Se chegarmos aqui, o
C EXCEDENTE DE I é 0 e fizemos o nosso aumento de preço do ano
passado.

                  EXCEDENTE (I) = 0
                  P (I) = PREÇO
                  FPUSHF (I) = ARCF
                  FPUSHB (I) = ARCB

               ENDIF
```

```
C ADVANCE fila.
C Se a fila estiver retornam vazios todo o excedente à fonte e SAIR

        NXTNODE = NXTQUEUE (I)
        IF (I.EQ.NXTNODE) THEN
          IF (CUTEXIT.EQ.1) THEN
            RETURN
          MAIS
            TOT_TIME = TOT_TIME + FLOAT (LONG (362) - TIMER1) / 60
            GOTO 600
          END IF
        MAIS
          NXTQUEUE (PrevNode) = NXTNODE
          NXTQUEUE (I) = 0
          I = NXTNODE
        END IF

        IF (COUNT.LT.THRESH) THEN
          COUNT = COUNT + 1
          Ir para 100
        END IF


C ***************** PERIODICALLY Procure um corte saturando
***************
C e recalcular os preços a ser igual ao distâncias mais próximas

500 CONTINUAR

X PRINT *, '** VERIFICAÇÃO de rescisão antecipada **'

        TIMER = Long (362)

C Atualizar LIMIAR

        THRESH = INT (1,1 * THRESH)

C reinicializar o COUNT de iterações para a próxima verificação para
um corte MIN

        COUNT = 0
        CUTCOUNT = + 1 CUTCOUNT
        FAZER 510 L = 1, NLABEL
          MARK (LABEL (L)) =. FALSE.
510 CONTINUAR

        OUTFLAG = .FALSE.

        NLABEL = 1
        LABEL (1) = SINK
        NSCAN = 0

550 CONTINUAR
        NSCAN = + 1 NSCAN
        NÓ = LABEL (NSCAN)
        MARK (nó) =. TRUE.
        ARC = FIN (nó)
580 IF (ARC.GT.0) THEN
           START = STARTN (ARC)
```

```
C Se este é um arco ao longo do qual o fluxo pode ser empurrado para o
nó
C Verifique se o OPP. Nó tem 0 EXCEDENTE E se assim ROTULAR o nó

           IF (F (ARC) .LT.U (ARC)) THEN
             IF (.NOT.MARK (START)) THEN
               IF (etiquetagem) THEN
                 IF (EXCEDENTE (START) .GT.0) THEN
                   OUTFLAG = .TRUE.
                 END IF
                 P (START) = P (nó) +1
               MAIS
                 IF (EXCEDENTE (START) .GT.0) THEN
                   CHECKTIME = CHECKTIME + FLOAT (LONG (362) - TIMER)
/ 60
                   GOTO 100
                 END IF
               END IF
               NLABEL = + 1 NLABEL
               LABEL (NLABEL) = INÍCIO
               MARK (START) =. TRUE.
             END IF
           END IF
           ARC = NXTIN (ARC)
           GOTO 580
         END IF

         ARC = FOUT (nó)
585 IF (ARC.GT.0) THEN
           END = ENDN (ARC)

C Se este é um arco ao longo do qual o fluxo pode ser empurrado para o
nó
C Verifique se o OPP. Nó tem 0 EXCEDENTE E se assim ROTULAR o nó

           IF (F (ARC) .GT.0) THEN
             IF (.NOT.MARK (END)) THEN
               IF (etiquetagem) THEN
                 IF (excedente (END) .GT.0) THEN
                   OUTFLAG = .TRUE.
                 END IF
                 P (END) = P (nó) +1
               MAIS
                 IF (excedente (END) .GT.0) THEN
                   CHECKTIME = CHECKTIME + FLOAT (LONG (362) - TIMER)
/ 60
                   GOTO 100
                 END IF
               END IF
               NLABEL = + 1 NLABEL
               LABEL (NLABEL) = END
               MARK (END) =. TRUE.
             END IF
           END IF
           ARC = NXTOU (ARC)
           GOTO 585
         END IF

         IF (NLABEL.GT.NSCAN) GOTO 550

C Se um nó W / excedente positivo é acessível a partir da PIA
```

```
C relabel os nós inacessível a N e voltar para mais ITERATION

        IF (OUTFLAG) THEN

          IF (etiquetagem) THEN
            FAZER 595 NÓ = 1, N
              IF (.NOT.MARK (nó)) THEN
                IF (P (nó) .LT.N) P (nó) = N
              END IF
595 CONTINUAR
          END IF

X PRINT *, 'para sair do check de rescisão antecipada'
          CHECKTIME = CHECKTIME + FLOAT (LONG (362) - TIMER) / 60
          GOTO 100
        MAIS
X PRINT *, 'SATURATED CUT encontrado'
X PRINT *, '# de nós em SINK lado do corte =', NLABEL
          CHECKTIME = CHECKTIME + FLOAT (LONG (362) - TIMER) / 60
          TOT_TIME = TOT_TIME + FLOAT (LONG (362) - TIMER1) / 60
          GOTO 600
        END IF

C
C ********************** ENCONTRAR A MAX-FLOW **********************
****
C
C SATURATED CUT FOI ENCONTRADO mas alguns nós possam ter EXCEDENTE
diferente de zero.
C PARA ENCONTRAR A MAX-FLOW, retornamos o fluxo adicional PARA A fonte
por
C utilizando o algoritmo SAME
C

600 CONTINUAR

       CUTEXIT = 1
       Relabel = .FALSE.
       COUNT = 0
       THRESH = GRANDE

        620nm NODE = 1, N
          MARK (nó) =. FALSE.
620 CONTINUAR

        MARK (SOURCE) =. TRUE.
        MARK (pia) =. TRUE.
        P (FONTE) = 0

        NLABEL = 1
        LABEL (1) = SOURCE
        NSCAN = 0

650 CONTINUAR
        NSCAN = + 1 NSCAN
        NÓ = LABEL (NSCAN)
        PREÇO = P (nó) +1

        ARC = FIN (nó)
660 IF (ARC.GT.0) THEN
          IF (U (ARC) .GT.0) THEN
```

```
            START = STARTN (ARC)
            IF (.NOT.MARK (START)) THEN
              NLABEL = + 1 NLABEL
              LABEL (NLABEL) = INÍCIO
              P (START) = PREÇO
              MARK (START) =. TRUE.
            END IF
          END IF
          ARC = NXTIN (ARC)
          GOTO 660
        END IF

        ARC = FOUT (nó)
670 IF (ARC.GT.0) THEN
          IF (F (ARC) .GT.0) THEN
            END = ENDN (ARC)
            IF (.NOT.MARK (END)) THEN
              NLABEL = + 1 NLABEL
              LABEL (NLABEL) = END
              P (END) = PREÇO
              MARK (END) =. TRUE.
            END IF
          END IF
          ARC = NXTOU (ARC)
          GOTO 670
        END IF

        IF (NLABEL.GT.NSCAN) GOTO 650

        GOTO 70

        FIM

C ********************************************** ***
      SUBROUTINE INIDAT
C Esta sub-rotina USAM O STARTN matrizes de dados E ENDN
C PARA CONSTRUIR AUXILIAR DE DADOS ARRAYS FOUT, NXTOU, FIN, E
C NXTIN que são exigidos por E-Relax-MF. Nisto SUBROUTINE
C arbitrariamente ENCOMENDAR AS ARCS DEIXANDO cada nó e STORE
C esta informação em FOUT E NXTOU. Da mesma forma, arbitragem
C RILLY ORDER os arcos que entram em cada nó e armazenar esta
C INFORMAÇÃO EM FIN E NXTIN. NA CONCLUSÃO DO
C CONSTRUÇÃO, TEMOS
C
C FOUT (I) = Primeiro ARC DEIXANDO NODE I.
C NXTOU (J) = PRÓXIMO ARC saem do nó CHEFE DE ARC J.
C FIN (I) = Primeiro ARC Entrando no passo I.
C NXTIN (J) = PRÓXIMO ARC ENTRANDO NO NÓ DE CAUDA ARC J.

      PARÂMETROS (maxNodes = 20000, MAXARCS = 120000)
      Implícita INTEGER (AZ)
      COMUM / escalares / N, NA, GRANDE, SOURCE, pia
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      INTEGER STARTN (MAXARCS)
      INTEGER ENDN (MAXARCS)
      INTEGER FIN (maxNodes)
```

```
      INTEGER FOUT (maxNodes)
      INTEGER NXTIN (MAXARCS)
      INTEGER NXTOU (MAXARCS)
      INTEGER FINALIN (maxNodes), FINALOU (maxNodes)
C
      FAZER 20 NODE = 1, N
        FIN (nó) = 0
        FOUT (nó) = 0
        FINALIN (nó) = 0
        FINALOU (nó) = 0
20 CONTINUAR
C
      Fazer 30 ARC = 1, NA
        START = STARTN (ARC)
        END = ENDN (ARC)
        IF (FOUT (START) .NE.0) THEN
          NXTOU (FINALOU (START)) = ARC
        MAIS
          FOUT (START) = ARC
        END IF
        IF (FIN (END) .NE.0) THEN
          NXTIN (FINALIN (END)) = ARC
        MAIS
          FIN (END) = ARC
        END IF
        FINALOU (START) = ARC
        FINALIN (END) = ARC
        NXTIN (ARC) = 0
        NXTOU (ARC) = 0
30 CONTINUAR
C
      RETURN
      FIM

C ********************************************** ***


      SUBROUTINE SETRAN (ISEED)
      IMPLICIT real * 8 (AH, OZ), Integer * 4 (IN)
C **********************************************
**********************
C PORTABLE congruential (uniforme) RANDOM gerador de números:
NEXT_VALUE C = [(7 ** 5) * PREVIOUS_VALUE] MODULO [(2 ** 31) -1]
C
C Este gerador é composto por duas ROTINAS:
C (1) SETRAN - inicializa constantes e SEED
C (2) RRAN - gera um número aleatório REAIS
C
C O GERADOR DE RODA uma máquina com pelo menos 32 bits de precisão.
C DA SEED (ISEED) deve estar no intervalo (1, (2 ** 31 -1)).
C **********************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      IF (ISEED.LT.1) PARADA 77
      MULT = 16807
      MODUL = 2147483647
      I15 = 2 ** 15
      I16 = 2 ** 16
      JRAN = ISEED
      RETURN
      FIM
```

```
C
      RAN função real ()
      IMPLICIT real * 4 (AH, OZ), Integer * 4 (IN)
C ***********************************************
**********************
C RAN gera um número aleatório real entre 0 e 1
C ***********************************************
**********************
      COMUM / RANDM / MULT, MODUL, I15, I16, JRAN
      Ixhi = JRAN / I16
      IXLO = JRAN-Ixhi * I16
      IXALO = IXLO * MULT
      LEFTLO = IXALO / I16
      IXAHI = Ixhi * MULT
      IFULHI = IXAHI + LEFTLO
      IRTLO = IXALO-LEFTLO * I16
      IOVER = IFULHI / I15
      IRTHI = IFULHI-IOVER * I15
      JRAN = ((IRTLO-MODUL) + IRTHI * I16) + IOVER
      IF (JRAN.LT.0) JRAN = JRAN + MODUL
      RAN = FLOAT (JRAN) / FLOAT (MODUL)
      RETURN
      FIM


***********************************************
************************

$ \ E $ -Relaxation Codes

***********************************************
************************


$ \ E $ -Relax

Este código implementa o método $ \ e $ -relaxation com
$ \ E $ scaling para o problema de fluxo de custo mínimo (cf. \ Seção
4.5).

C ** AMOSTRA programa de chamada para o e-RELAX **
C
C Este programa vai LER UM PROBLEMA arquivo no formato PADRÃO
C e resolvê-lo usando o e-RELAX.
C
C Esta é uma versão em escala E QUE utiliza uma fila CYCLIC
C
C ********************************************** **
C
      PARÂMETROS (maxNodes = 8000, MAXARCS = 25000)
      Implícita INTEGER (AZ)
      Common / escalares / N, NA, GRANDES, fator, EPS,
     & THRESHSURPLUS, Sfactor
      COMUM / STATS / NCYC, Niter, totaliter
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / UBOUND / U
      COMUM / FLUXO / F
      Common / PREÇOS / P
      COMUM / BLK3 / excedente
      COMUM / BLK4 / FIN
```

```
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      COMUM / BLK11 / FPUSHF
      COMUM / BLK12 / NXTPUSHF
      COMUM / BLK13 / FPUSHB
      COMUM / BLK14 / NXTPUSHB
      COMUM / CUSTO / COST
      COMUM / Fila / NXTQUEUE
      INTEGER NXTQUEUE (maxNodes)
      INTEGER STARTN (MAXARCS)
      INTEGER ENDN (MAXARCS)
      INTEGER U (MAXARCS)
      INTEGER F (MAXARCS)
      INTEGER excedente (maxNodes)
      INTEGER FIN (maxNodes)
      INTEGER FOUT (maxNodes)
      INTEGER NXTIN (MAXARCS)
      INTEGER NXTOU (MAXARCS)
      INTEGER P (maxNodes)
      INTEGER FPUSHF (maxNodes)
      INTEGER NXTPUSHF (MAXARCS)
      INTEGER FPUSHB (maxNodes)
      INTEGER NXTPUSHB (MAXARCS)
      INTEGER COST (MAXARCS)
      Real * 8 TT, TIMER
      DOUBLE PRECISION TCOST, PRODUTOS


      PRINT *, «método E-relaxamento para MIN CUSTO DE FLUXO '
      PRINT *, '************************************'
      PRINT *, 'READING problema de dados'
      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' velho ')
      Rewind (13)

C LEIA número de nós e arcos

      LER (13,1010) N, NA

C LEIA início, fim, custo e capacidade de cada ARC
C e gerar MAX CUSTO VALOR

      MAXCOST = 0

      FAZER 5 I = 1, NA
        READ (13,1020) STARTN (I), ENDN (I), CUSTO (I), L (I)
        COST (I) = CUSTO (I) * (N + 1)
        IF (ABS (COST (I)). GT.MAXCOST) MAXCOST = ABS (COST (I))
5 CONTINUAR

C LEIA SUPPLY de cada nó

      FAZER 8 I = 1, N
        READ (13,1000) EXCEDENTE (I)
8 CONTINUAR

      ENDFILE (13)
      Rewind (13)

      PRINT *, 'FIM DE LEITURA'
```

```
1000 FORMATO (1I8)
1010 FORMATO (2I8)
1020 FORMATO (4I8)

C
      GRANDES = 1500000000

C REINICIAR PREÇOS E LISTAS DE PRESSÃO

      Faça 10 NODE = 1, N
        FPUSHF (nó) = 0
        FPUSHB (nó) = 0
        P (nó) = - INT (GRANDE / 10)
10 CONTINUAR

C nos seguintes afirmações, os parâmetros E-dimensionamento estão
definidos.
C as seguintes faixas são recomendadas:
C EPSILON PARTIDA: ENTRE MAXCOST / 20 a MAXCOST / 2
C FACTOR DE ESCALA: entre quatro a dez

25 CONTINUAR
      PRINT *, "ENTER PARTIDA EPSILON '
      PRINT *, 'deve ser entre 1 e', MAXCOST
      LER *, EPS
      IF (EPS.LT.1) Vá para 25
      PRINT *, "ENTER o fator de escala '
30 CONTINUAR
      LER *, FACTOR
      IF ((EPS.GT.1) .E. (FACTOR.LE.1)) THEN
        PRINT *, "ENTER o fator de escala; Deve ser superior a 1 '
        IR PARA 30
      END IF
C
        MAXSURPLUS = -Grande
        FAZER 920 NÓ = 1, N
          IF (EXCEDENTE (nó) .GT.MAXSURPLUS) THEN
            MAXSURPLUS = excedente (nó)
          END IF
920 CONTINUAR

C CONFIGURAR O SSCALE PARÂMETROS como 1 para permitir EXCEDENTE DE
ESCALA
CA HEURISTIC esquema é usado para definir o Sfactor E
C PARÂMETROS THRESHSURPLUS que controlam EXCEDENTE DE ESCALA

      SSCALE = 1

      IF (SSCALE.EQ.1) THEN
        Sfactor = 2 + INT (FACTOR * MAXSURPLUS / MAXCOST)
        IF (SFACTOR.LT.4) Sfactor = 4
        THRESHSURPLUS = INT (MAXSURPLUS / Sfactor)
        IF (EPS.EQ.1) THRESHSURPLUS = 0
      MAIS
        THRESHSURPLUS = 0
      END IF

      PRINT *, 'INICIALIZANDO estruturas de dados'
      CHAMADA INIDAT

      PRINT *, '*********************************'
```

```
      PRINT *, 'chamado E-relaxar para MCF (+ NÓS DE EXCEDENTES
Iterated ONLY)'
      TIMER = Long (362)
      CHAMADA EPS_RELAX
      TT = (LONG (362) - TIMER) / 60
      PRINT *, 'TIME TOTAL =', TT, 's.'

      TCOST = 0
      FAZER 330 I = 1, NA
        COST (I) = CUSTO (I) / (N + 1)
        PRODUTO = FLOAT (F (I) * CUSTO (I))
        TCOST = TCOST + PRODUTOS
330 CONTINUAR

      ESCREVA (9,1100) TCOST
1100 FORMATO ('', 'custo ótimo =', F14.2)

      PRINT *, '*********************************'
      PRINT *, '# de iterações =', totaliter
      PRINT *, '# DE SN preço sobe =', Niter
      PRINT *, '*********************************'

C Verifica a exactidão da ANSWER

      IF (EPS.NE.1) THEN
        PRINT *, '* ATENÇÃO * THE EPSILON final é igual', EPS
      END IF
      FAZER 80 NODE = 1, N
        IF (EXCEDENTE (nó) .NE.0) THEN
          PRINT *, "excedente diferente de zero AT NODE ', NODE
        ENDIF
80 CONTINUAR
      FAZER 90 ARC = 1, NA
        COST (ARC) = CUSTO (ARC) * (N + 1)
        IF (F (ARC) .GT.0) THEN
          IF (P (STARTN (ARC).) - P (ENDN (ARC)) o tenente-EPS + COST
(ARC)) THEN
            PRINT *, 'E-CS VIOLADOS AT ARC ", ARC
          ENDIF
        ENDIF
        IF (F (ARC) .LT.U (ARC)) THEN
          IF (P (STARTN (ARC).) - P (ENDN (ARC)) GT.EPS + COST (ARC))
THEN
            PRINT *, 'E-CS VIOLADOS AT ARC ", ARC
          ENDIF
        ENDIF
90 CONTINUAR
      PRINT *, ''
      PRINT *, 'programa terminou; PRESS <CR> '
      PAUSA
      PARE
      FIM



C ********************************************** ***
      SUBROUTINE INIDAT
C Esta sub-rotina USAM O STARTN matrizes de dados E ENDN
C PARA CONSTRUIR AUXILIAR DE DADOS ARRAYS FOUT, NXTOU, FIN, E
C NXTIN que são exigidos por E-RELAX. Nisto SUBROUTINE
C arbitrariamente ENCOMENDAR AS ARCS DEIXANDO cada nó e STORE
```

```
C esta informação em FOUT E NXTOU. Da mesma forma, arbitragem
C RILLY ORDER os arcos que entram em cada nó e armazenar esta
C INFORMAÇÃO EM FIN E NXTIN. NA CONCLUSÃO DO
C CONSTRUÇÃO, TEMOS QUE
C
C FOUT (I) = Primeiro ARC DEIXANDO NODE I.
C NXTOU (J) = PRÓXIMO ARC saem do nó CHEFE DE ARC J.
C FIN (I) = Primeiro ARC Entrando no passo I.
C NXTIN (J) = PRÓXIMO ARC ENTRANDO NO NÓ DE CAUDA ARC J.

      PARÂMETROS (maxNodes = 8000, MAXARCS = 25000)
      Implícita INTEGER (AZ)
      Common / escalares / N, NA, GRANDE, FACTOR, EPS
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      INTEGER STARTN (MAXARCS)
      INTEGER ENDN (MAXARCS)
      INTEGER FIN (maxNodes)
      INTEGER FOUT (maxNodes)
      INTEGER NXTIN (MAXARCS)
      INTEGER NXTOU (MAXARCS)
      INTEGER FINALIN (maxNodes), FINALOU (maxNodes)

      FAZER 20 NODE = 1, N
        FIN (nó) = 0
        FOUT (nó) = 0
        FINALIN (nó) = 0
        FINALOU (nó) = 0
20 CONTINUAR

      Fazer 30 ARC = 1, NA
        START = STARTN (ARC)
        END = ENDN (ARC)
        IF (FOUT (START) .NE.0) THEN
          NXTOU (FINALOU (START)) = ARC
        MAIS
          FOUT (START) = ARC
        END IF
        IF (FIN (END) .NE.0) THEN
          NXTIN (FINALIN (END)) = ARC
        MAIS
          FIN (END) = ARC
        END IF
        FINALOU (START) = ARC
        FINALIN (END) = ARC
        NXTIN (ARC) = 0
        NXTOU (ARC) = 0
30 CONTINUAR

      RETURN
      FIM



C ********************************************** ***************
C
C SCALED E-RELAXAMENTO
```

```
C Esta versão usa uma fila para selecionar nós.
NÓS DE C entrar na fila na parte inferior.
C
C ********************************************** ***************

      EPS_RELAX SUBROUTINE
      NONE IMPLICIT
      INTEGER N, NA, GRANDE, FACTOR, EPS, ARC, ARCF, ARCB
      INTEGER NCYC, Niter, NODE, PRICE, PRICEINCR
      INTEGER I, J, K, LN, CAPOUT, Capin, PrevNode, LASTQUEUE
      INTEGER SURPI, Reji, NEXT, início, fim Pinicial, PEND, maxprice
      INTEGER RESID, DEL, FLOW, NXTNODE, totaliter
      INTEGER REFPRICE, PRJ, XP, REFPRJ, PRICEJ, THRESHSURPLUS,
Sfactor
      INTEGER NXTQUEUE (1)
      INTEGER STARTN (1), ENDN (1), L (1), F (1), EXCEDENTE (1), FIN
(1), FOUT (1)
      INTEGER NXTIN (1), NXTOU (1), P (1)
      INTEGER FPUSHF (1), NXTPUSHF (1), FPUSHB (1), NXTPUSHB (1)
      CUSTO INTEIRO (1)

      Common / escalares / N, NA, GRANDES, fator, EPS,
     & THRESHSURPLUS, Sfactor
      COMUM / STATS / NCYC, Niter, totaliter
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / UBOUND / U
      COMUM / FLUXO / F
      Common / PREÇOS / P
      COMUM / BLK3 / excedente
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      COMUM / BLK11 / FPUSHF
      COMUM / BLK12 / NXTPUSHF
      COMUM / BLK13 / FPUSHB
      COMUM / BLK14 / NXTPUSHB
      COMUM / CUSTO / COST
      COMUM / Fila / NXTQUEUE


C REINICIAR CONTAGEM DO NÚMERO DE CIMA ITERATIONS REALIZADAS

      Niter = 0
      Totaliter = 0
      NCYC = 0
      Maxprice = GRANDE

C reduzir as capacidades ARC

         DO 40 NODE = 1, N
            CAPOUT = 0
            ARC = FOUT (nó)
41 IF (ARC.GT.0) THEN
               CAPOUT = MIN (grande, CAPOUT + U (ARC))
               ARC = NXTOU (ARC)
               IR PARA 41
            END IF
            CAPOUT = MIN (grande, CAPOUT-EXCEDENTE (nó))
            IF (CAPOUT.LT.0) GOTO 400
```

```
            Capin = 0
            ARC = FIN (nó)
43 IF (ARC.GT.0) THEN
               IF (U (ARC) .gt. CAPOUT) THEN
                  U (ARC) = CAPOUT
               ENDIF
               Capin = MIN (grande, Capin + U (ARC))
               ARC = NXTIN (ARC)
               IR PARA 43
            END IF
            Capin = MIN (grande, Capin + excedente (nó))
            IF (CAPIN.LT.0) GOTO 400
            ARC = FOUT (nó)
45 IF (ARC.GT.0) THEN
               IF (U (ARC) .gt. Capin) THEN
                  U (ARC) = Capin
               ENDIF
               ARC = NXTOU (ARC)
               IR PARA 45
            END IF
40 CONTINUAR

C SET ARC FLUXO PARA SATISFAZER E-CS

       FAZER 49 ARC = 1, NA
         START = STARTN (ARC)
         END = ENDN (ARC)
         Pinicial = P (START)
         PEND = P (END)
         IF (PSTART.GE.PEND + COST (ARC) + EPS) THEN
           EXCEDENTE (START) = excedente (START) -U (ARC)
           EXCEDENTE (END) = excedente (END) + U (ARC)
           F (ARC) = U (ARC)
         MAIS
           F (ARC) = 0
         END IF
49 CONTINUAR

C *********** início de uma nova fase ESCALA ***********

60 CONTINUAR

C ***** ***** INITIALIZATION FILA

        FAZER 82 NODE = 1, N-1
            NXTQUEUE (nó) = NODE + 1
82 CONTINUAR
        NXTQUEUE (N) = 1
        I = 1
        PrevNode = N
        LASTQUEUE = N

C ************ iniciar um novo ciclo de até ITERATIONS ************

100 CONTINUAR

C passamos para o próximo nó (Nó I) para a iteração

        SURPI = excedente (I)
        IF (SURPI.GT.THRESHSURPLUS) THEN
```

```
C & ARCF ARCB são os valores atuais dos ARCS de partida da
Aperte LISTAS DE I

           Totaliter = totaliter + 1
           ARCF = FPUSHF (I)
           ARCB = FPUSHB (I)

           PREÇO = P (I)
C PRINT *, 'NODE', I, 'INÍCIO PREÇO =', PREÇO

115 IF ((ARCF .gt. 0) .OR. (ARCB .gt. 0)) THEN

C começar por tentar afastar FLUXO EM ARCS que foram
C admissível a final da última iteração (SE HOUVER)
C neste nodo. WE deve verificar se eles ainda estão
C admissível.


120 IF ((SURPI .gt. 0) .E. (ARCF .gt. 0)) THEN
                  J = ENDN (ARCF)
                  IF (PREÇO-P (J) -Custo (ARCF) .EQ.EPS) THEN
                     RESID = L (ARCF) - F (ARCF)
                     IF (RESID.GT.0) THEN
                      IF (SURPI .GE. RESID) THEN
                         F (ARCF) = L (ARCF)
                         SURPI = SURPI - RESID
                         EXCEDENTE (J) = excedente (J) + RESID
                         ARCF = NXTPUSHF (ARCF)
                      MAIS
                         F (ARCF) = F (ARCF) + SURPI
                         EXCEDENTE (J) = excedente (J) + SURPI
                         SURPI = 0
                      ENDIF
                      IF (EXCEDENTE (J) .GT.THRESHSURPLUS) THEN
                       IF (NXTQUEUE (J) .EQ.0) THEN
                        NXTQUEUE (PrevNode) = J
                        NXTQUEUE (J) = I
                        PrevNode = J
                       END IF
                      END IF
                    MAIS
                      ARCF = NXTPUSHF (ARCF)
                    ENDIF
                  MAIS
                     ARCF = NXTPUSHF (ARCF)
                  ENDIF
                  GOTO 120
                ENDIF

121 IF ((SURPI .gt. 0) .E. (ARCB .gt. 0)) THEN
               J = STARTN (ARCB)
               IF (PREÇO-P (J) + COST (ARCB) .EQ.EPS) THEN
                  RESID = F (ARCB)
                  IF (RESID.GT.0) THEN
                     IF (SURPI .GE. RESID) THEN
                        SURPI = SURPI - RESID
                        EXCEDENTE (J) = excedente (J) + RESID
                        F (ARCB) = 0
                        ARCB = NXTPUSHB (ARCB)
                     MAIS
                        F (ARCB) = F (ARCB) - SURPI
```

```
                        EXCEDENTE (J) = excedente (J) + SURPI
                        SURPI = 0
                      ENDIF
                      IF (EXCEDENTE (J) .GT.THRESHSURPLUS) THEN
                       IF (NXTQUEUE (J) .EQ.0) THEN
                        NXTQUEUE (PrevNode) = J
                        NXTQUEUE (J) = I
                        PrevNode = J
                       END IF
                      END IF
                    MAIS
                      ARCB = NXTPUSHB (ARCB)
                    ENDIF
                  MAIS
                    ARCB = NXTPUSHB (ARCB)
                  ENDIF
                  GOTO 121
                ENDIF
             ENDIF

C ***** FIM DE D-EMPURRA; VERIFIQUE SE PREÇO RISE É NECESSÁRIO *****

            IF ((ARCF .EQ. 0) .E. (ARCB .EQ. 0)) THEN

C ****************** FAZER UM AUMENTO DE PREÇO *********************

                   REFPRICE = PREÇO
                   PREÇO = GRANDE
                   Niter = Niter + 1
                   ARC = FOUT (I)
111 SE (ARC .gt. 0) THEN
                    IF (F (ARC) .lt. U (ARC)) THEN
                       XP = P (ENDN (ARC)) + COST (ARC)
                       IF (XP.LT.PRICE) THEN
                          PREÇO = XP
                          ARCF = ARC
                          NXTPUSHF (ARC) = 0
                       MAIS
                        IF (XP.EQ.PRICE) THEN
                          NXTPUSHF (ARC) = ARCF
                          ARCF = ARC
                        END IF
                       ENDIF
                    ENDIF
                    ARC = NXTOU (ARC)
                    GOTO 111
                  ENDIF
                  ARC = FIN (I)

112 SE (ARC .gt. 0) THEN
                   IF (F (ARC) .gt. 0) THEN
                      XP = P (STARTN (ARC)) - COST (ARC)
                      IF (XP.LT.PRICE) THEN
                         PREÇO = XP
                         ARCB = ARC
                         ARCF = 0
                         NXTPUSHB (ARC) = 0
                      MAIS
                       IF (XP.EQ.PRICE) THEN
                         NXTPUSHB (ARC) = ARCB
                         ARCB = ARC
```

```
                    END IF
                   ENDIF
                 ENDIF
                 ARC = NXTIN (ARC)
                 GOTO 112
               ENDIF

C Se o preço = GRANDE, a lista TOQUE É PROVÁVEL VAZIO, preço tão
definida como pelo menos THE
C NÍVEL DO MELHOR ARC

           IF (PRICE.EQ.LARGE) THEN
                ARCF = 0
                ARCB = 0

                ARC = FOUT (I)
211 IF (ARC.GT.0) THEN
                 XP = P (ENDN (ARC)) + COST (ARC)
                 IF (XP.LT.PRICE) THEN
                   PREÇO = XP
                 MAIS
                   IF (XP.GE.LARGE) THEN
                    PRINT *, 'PREÇO DO', I, "excedeu o INT. GAMA '
                    PAUSA
                    PARE
                   END IF
                 ENDIF
                 ARC = NXTOU (ARC)
                 GOTO 211
               ENDIF
               ARC = FIN (I)

212 IF (ARC.GT.0) THEN
                 XP = P (STARTN (ARC)) - COST (ARC)
                 IF (XP.LT.PRICE) THEN
                    PREÇO = XP
                 MAIS
                   IF (XP.GE.LARGE) THEN
                    PRINT *, 'PREÇO DO', I, "excedeu o INT. GAMA '
                    PAUSA
                    PARE
                   END IF
                 ENDIF
                 ARC = NXTIN (ARC)
                 GOTO 212
               ENDIF

               IF (SURPI.GT.0) GOTO 400
               IF (PRICE.LT.INT (GRANDE / 10)) PREÇO = INT (GRANDE /
10)

           END IF


C SET PREÇO do nó i


           Preço = Preço + EPS
           P (I) = PREÇO
           FPUSHF (I) = ARCF
           FPUSHB (I) = ARCB
```

```
X PRICEINCR = PREÇO-REFPRICE
X IF (PRICEINCR.LE.0) THEN
X PRINT *, 'não positivo INCREMENTO PREÇO =', PRICEINCR
X PAUSE
X PARAR
X ENDIF

C FIM DO PREÇO RISE; SE HÁ AINDA UM LOTE DE EXCEDENTE,
C tentar fazer tanto insistir.

           IF (SURPI.GT.THRESHSURPLUS) GOTO 115

        MAIS

C SE NO PREÇO RISE OCORREU REAJUSTE O início do LISTAS DE PRESSÃO

           FPUSHF (I) = ARCF
           FPUSHB (I) = ARCB
        ENDIF

C Execute a escrituração FINAL da iteração


        EXCEDENTE (I) = SURPI

C Se o preço é muito alto, então algo está errado

X IF (PRICE.GT.MAXPRICE) THEN
X GO TO 400
X ENDIF

       ENDIF

C ********** END OF UP ITERATION COMEÇAR no nó i *********

C de seleção para o final da fila. Este passo não é NECESSÁRIO PARA
C ALGORITMO; É INCLUÍDO de recolha de estatísticas SOBRE
COMPORTAMENTO DO C algoritmo do, EG CONTANDO O número de ciclos

X IF (I.EQ.LASTQUEUE) THEN
X = LASTQUEUE PrevNode
X = NCYC NCYC + 1
X PRINT *, 'CICLO #', NCYC, '# dos aumentos de preços ", Niter
X END IF


C VERIFICAÇÃO DE RESCISÃO DE ESCALA FASE. Se o escalonamento fase é
NÃO TERMINOU C, o avanço da fila e voltar para levar outro nó.

        NXTNODE = NXTQUEUE (I)
        IF (I.NE.NXTNODE) THEN
          NXTQUEUE (I) = 0
          NXTQUEUE (PrevNode) = NXTNODE
          I = NXTNODE
          Ir para 100
        END IF

C ************** FIM DE subproblema (escalonamento FASE)
**************
```

```
X PRINT *, 'FIM DE ESCALA FASE'

C Execute uma verificação de diagnóstico

X = 0 LN
X FAZER 500 NÓ = 1, N
X IF (EXCEDENTE (nó) .NE.0) THEN
X = LN LN + 1
X ENDIF
500 CONTINUAR
X PRINT *, "excedente diferente de zero AT ', LN," nós "
X FAZER 600 ARC = 1, NA
X IF (F (ARC) .GT.0) THEN
X IF (P (STARTN (ARC).) - P (ENDN (ARC)) o tenente-EPS + COST (ARC))
THEN
X PRINT *, 'E-CS VIOLADOS AT ARC ", ARC
X ENDIF
X ENDIF
X IF (F (ARC) .LT.U (ARC)) THEN
X IF (P (STARTN (ARC)) - P (ENDN (ARC)) GT.EPS + COST (ARC).) THEN
X PRINT *, 'E-CS VIOLADOS AT ARC ", ARC
X ENDIF
X ENDIF
600 CONTINUAR
X PRINT *, 'FIM DE VERIFICAÇÃO DE FASE OLD'

C ****** SE EPSILON É uma RESCINDIR; MAIS REDUZIR EPSILON ******

        IF (EPS.EQ.1) THEN
           RETURN
        MAIS
           EPS = INT (EPS / FACTOR)
           IF (EPS.LT.1) EPS = 1
        END IF
        THRESHSURPLUS = INT (THRESHSURPLUS / Sfactor)
        IF (EPS.EQ.1) THRESHSURPLUS = 0

C redefinir o FLUXO & LISTAS DE PRESSÃO; ENCONTRE O preço mínimo

       XP = GRANDE
       FAZER 800 NÓ = 1, N
         FPUSHF (nó) = 0
         FPUSHB (nó) = 0
         IF (P (nó) .LT.XP) XP = P (nó)
800 CONTINUAR

C REDUZIR OS PREÇOS PARA REDUZIR RISCO DE ESTOURO

       DEL = XP + INT (GRANDE / 10)

       IF (DEL.LT.0) DEL = 0
       FAZER 810 NÓ = 1, N
         P (nó) = P (nó) -Del
810 CONTINUAR

X PRINT *, «modificar as ARC FLUXO PARA SATISFAZER E-CS '
       FAZER 900 ARC = 1, NA
         START = STARTN (ARC)
         END = ENDN (ARC)
         Pinicial = P (START)
         PEND = P (END)
```

```
          IF (PSTART.GT.PEND + EPS + COST (ARC)) THEN
            RESID = U (ARC) -F (ARC)
            IF (RESID.GT.0) THEN
              EXCEDENTE (START) = excedente (START) -RESID
              EXCEDENTE (END) = excedente (END) + RESID
              F (ARC) = U (ARC)
            END IF
          MAIS
           IF (PSTART.LT.PEND-EPS + COST (ARC)) THEN
             FLUXO = F (ARC)
             IF (FLOW.GT.0) THEN
               EXCEDENTE (START) = excedente (START) + FLUXO
               EXCEDENTE (END) = excedente (END) -FLOW
               F (ARC) = 0
             END IF
           END IF
          END IF
900 CONTINUAR


C voltar para outra FASE

X PRINT *, "VERIFICAÇÃO E-CS ANTES DE NOVA FASE '
X FAZER 700 ARC = 1, NA
X IF (F (ARC) .GT.0) THEN
X IF (P (STARTN (ARC).) - P (ENDN (ARC)) o tenente-EPS + COST (ARC))
THEN
X PRINT *, 'E-CS VIOLADOS AT ARC ", ARC
X ENDIF
X ENDIF
X IF (F (ARC) .LT.U (ARC)) THEN
X IF (P (STARTN (ARC)) - P (ENDN (ARC)) GT.EPS + COST (ARC).) THEN
X PRINT *, 'E-CS VIOLADOS AT ARC ", ARC
X ENDIF
X ENDIF
700 CONTINUAR
       IR PARA 60

400 PRINT *, "o problema é inviável '
       PAUSA
       PARE

       FIM



$ \ E $ -Relax-N

Este código é o mesmo que $ \ $ e -Relax mas permite
iterações de relaxamento a partir de nós com tanto negativos como
positivos
excedente.


C ** AMOSTRA programa de chamada para o e-RELAX **
C
C Este programa vai LER UM PROBLEMA arquivo no formato PADRÃO
C e resolvê-lo usando o e-RELAX.
C
C Esta é uma versão em escala E QUE utiliza uma fila CYCLIC
C
```

```
ITERATIONS C são feitas a partir positivos e negativos
NÓS DE C em excesso. Um mecanismo adaptativo é usado para decidir
C QUER fazer uma iteração de um nó EXCEDENTE NEGATIVO
C
C ********************************************** **
C
      PARÂMETROS (maxNodes = 8000, MAXARCS = 25000)
      Implícita INTEGER (AZ)
      Common / escalares / N, NA, GRANDES, fator, EPS,
     & THRESHSURPLUS, Sfactor
      COMUM / STATS / NCYC, Niter, totaliter
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / UBOUND / U
      COMUM / FLUXO / F
      Common / PREÇOS / P
      COMUM / BLK3 / excedente
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      COMUM / BLK11 / FPUSHF
      COMUM / BLK12 / NXTPUSHF
      COMUM / BLK13 / FPUSHB
      COMUM / BLK14 / NXTPUSHB
      COMUM / CUSTO / COST
      COMUM / Fila / NXTQUEUE
      COMUM / STATUS / STATUS
      INTEGER NXTQUEUE (maxNodes)
      INTEGER STARTN (MAXARCS)
      INTEGER ENDN (MAXARCS)
      INTEGER U (MAXARCS)
      INTEGER F (MAXARCS)
      INTEGER excedente (maxNodes)
      INTEGER FIN (maxNodes)
      INTEGER FOUT (maxNodes)
      INTEGER NXTIN (MAXARCS)
      INTEGER NXTOU (MAXARCS)
      INTEGER P (maxNodes)
      INTEGER FPUSHF (maxNodes)
      INTEGER NXTPUSHF (MAXARCS)
      INTEGER FPUSHB (maxNodes)
      INTEGER NXTPUSHB (MAXARCS)
      INTEGER COST (MAXARCS)
      INTEGER STATUS (maxNodes)
      Real * 8 TCOST, TT, TIMER


      PRINT *, «método E-relaxamento para MIN CUSTO DE FLUXO '
      PRINT *, '***********************************'
      PRINT *, 'READING problema de dados'
      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' velho ')
      Rewind (13)

C LEIA número de nós e arcos

      LER (13,1010) N, NA

C LEIA início, fim, custo e capacidade de cada ARC
C e gerar MAX CUSTO VALOR
```

```
      MAXCOST = 0

      FAZER 5 I = 1, NA
        READ (13,1020) STARTN (I), ENDN (I), CUSTO (I), L (I)
        COST (I) = CUSTO (I) * (N + 1)
        IF (ABS (COST (I)). GT.MAXCOST) MAXCOST = ABS (COST (I))
5 CONTINUAR

C LEIA SUPPLY de cada nó

      FAZER 8 I = 1, N
        READ (13,1000) EXCEDENTE (I)
8 CONTINUAR

      ENDFILE (13)
      Rewind (13)

      PRINT *, 'FIM DE LEITURA'

1000 FORMATO (1I8)
1010 FORMATO (2I8)
1020 FORMATO (4I8)

C
      GRANDES = 1500000000

C REINICIAR PREÇOS de nó, EXCEDENTES ETC.

      Faça 10 NODE = 1, N
        FPUSHF (nó) = 0
        FPUSHB (nó) = 0
        P (nó) = 0
        STATUS (nó) = 0
10 CONTINUAR

C nos seguintes afirmações, os parâmetros E-dimensionamento estão
definidos.
C as seguintes faixas são recomendadas:
C EPSILON PARTIDA: ENTRE MAXCOST / 20 a MAXCOST / 2
C FACTOR DE ESCALA: entre quatro a dez

25 CONTINUAR
      PRINT *, "ENTER PARTIDA EPSILON '
      PRINT *, 'deve ser entre 1 e', MAXCOST
      LER *, EPS
      IF (EPS.LT.1) Vá para 25
      PRINT *, "ENTER o fator de escala '
30 CONTINUAR
      LER *, FACTOR
      IF ((EPS.GT.1) .E. (FACTOR.LE.1)) THEN
        PRINT *, "ENTER o fator de escala; Deve ser superior a 1 '
        IR PARA 30
      END IF
C
        MAXSURPLUS = -Grande
        FAZER 920 NÓ = 1, N
          IF (EXCEDENTE (nó) .GT.MAXSURPLUS) THEN
            MAXSURPLUS = excedente (nó)
          END IF
920 CONTINUAR
```

```
C CONFIGURAR O SSCALE PARÂMETROS como 1 para permitir EXCEDENTE DE
ESCALA
CA HEURISTIC esquema é usado para definir o Sfactor E
C PARÂMETROS THRESHSURPLUS que controlam EXCEDENTE DE ESCALA

      SSCALE = 1

      IF (SSCALE.EQ.1) THEN
        Sfactor = 2 + INT (FACTOR * MAXSURPLUS / MAXCOST)
        IF (SFACTOR.LT.4) Sfactor = 4
        THRESHSURPLUS = INT (MAXSURPLUS / Sfactor)
        IF (EPS.EQ.1) THRESHSURPLUS = 0
      MAIS
        THRESHSURPLUS = 0
      END IF

      PRINT *, 'INICIALIZANDO estruturas de dados'
      CHAMADA INIDAT

      PRINT *, '*********************************'
      PRINT *, 'chamado E-relaxar para MCF (+ E - EXCEDENTE NODES
Iterated)'
      TIMER = Long (362)
      CHAMADA EPS_RELAX_N
      TT = (LONG (362) - TIMER) / 60
      PRINT *, 'TIME TOTAL =', TT, 's.'

      TCOST = 0
      FAZER 330 I = 1, NA
330 TCOST TCOST = + F (I) * CUSTO (I) / (N + 1)
      ESCREVA (9,1100) TCOST
1100 FORMATO ('', 'custo ótimo =', F14.2)

      PRINT *, '*********************************'
      PRINT *, '# de iterações =', totaliter
      PRINT *, '# DE SN preço sobe =', Niter
      PRINT *, '*********************************'

C Verifica a exactidão da ANSWER

      IF (EPS.NE.1) THEN
        PRINT *, '* ATENÇÃO * THE EPSILON final é igual', EPS
      END IF
      FAZER 80 NODE = 1, N
        IF (EXCEDENTE (nó) .NE.0) THEN
          PRINT *, "excedente diferente de zero AT NODE ', NODE
        ENDIF
80 CONTINUAR
      FAZER 90 ARC = 1, NA
        IF (F (ARC) .GT.0) THEN
          IF (P (STARTN (ARC).) - P (ENDN (ARC)) o tenente-EPS + COST
(ARC)) THEN
            PRINT *, 'E-CS VIOLADOS AT ARC ", ARC
          ENDIF
        ENDIF
        IF (F (ARC) .LT.U (ARC)) THEN
          IF (P (STARTN (ARC).) - P (ENDN (ARC)) GT.EPS + COST (ARC))
THEN
            PRINT *, 'E-CS VIOLADOS AT ARC ", ARC
          ENDIF
        ENDIF
```

```
90 CONTINUAR
      PRINT *, ''
      PRINT *, 'programa terminou; PRESS <CR> '
      PAUSA
      PARE
      FIM



C ********************************************** ***
      SUBROUTINE INIDAT
C Esta sub-rotina USAM O STARTN matrizes de dados E ENDN
C PARA CONSTRUIR AUXILIAR DE DADOS ARRAYS FOUT, NXTOU, FIN, E
C NXTIN que são exigidos por E-Relax-N. Nisto SUBROUTINE
C arbitrariamente ENCOMENDAR AS ARCS DEIXANDO cada nó e STORE
C esta informação em FOUT E NXTOU. Da mesma forma, arbitragem
C RILLY ORDER os arcos que entram em cada nó e armazenar esta
C INFORMAÇÃO EM FIN E NXTIN. NA CONCLUSÃO DO
C CONSTRUÇÃO, TEMOS QUE
C
C FOUT (I)    = Primeiro ARC DEIXANDO NODE I.
C NXTOU (J) = PRÓXIMO ARC saem do nó CHEFE DE ARC J.
C FIN (I) = Primeiro ARC Entrando no passo I.
C NXTIN (J) = PRÓXIMO ARC ENTRANDO NO NÓ DE CAUDA ARC J.

      PARÂMETROS (maxNodes = 8000, MAXARCS = 25000)
      Implícita INTEGER (AZ)
      Common / escalares / N, NA, GRANDE, FACTOR, EPS
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      INTEGER STARTN (MAXARCS)
      INTEGER ENDN (MAXARCS)
      INTEGER FIN (maxNodes)
      INTEGER FOUT (maxNodes)
      INTEGER NXTIN (MAXARCS)
      INTEGER NXTOU (MAXARCS)
      INTEGER FINALIN (maxNodes), FINALOU (maxNodes)

      FAZER 20 NODE = 1, N
        FIN (nó) = 0
        FOUT (nó) = 0
        FINALIN (nó) = 0
        FINALOU (nó) = 0
20 CONTINUAR

      Fazer 30 ARC = 1, NA
        START = STARTN (ARC)
        END = ENDN (ARC)
        IF (FOUT (START) .NE.0) THEN
          NXTOU (FINALOU (START)) = ARC
        MAIS
          FOUT (START) = ARC
        END IF
        IF (FIN (END) .NE.0) THEN
          NXTIN (FINALIN (END)) = ARC
        MAIS
          FIN (END) = ARC
```

```
        END IF
        FINALOU (START) = ARC
        FINALIN (END) = ARC
        NXTIN (ARC) = 0
        NXTOU (ARC) = 0
30 CONTINUAR

      RETURN
      FIM



C ********************************************** ***************
C
C SCALED E-RELAXAMENTO
C Esta versão usa uma fila para selecionar nós.
NÓS DE C entrar na fila na parte inferior.
C usa tanto ITERATIONS EXCEDENTES positivos e negativos.
C
C ********************************************** ***************

      EPS_RELAX_N SUBROUTINE
      NONE IMPLICIT
      INTEGER N, NA, GRANDE, FACTOR, EPS, ARC, ARCF, ARCB
      INTEGER NCYC, Niter, NODE, PRICE, PRICEINCR
      INTEGER I, J, K, LN, CAPOUT, Capin, PrevNode, LASTQUEUE
      INTEGER SURPI, Reji, NEXT, início, fim Pinicial, PEND, maxprice
      INTEGER RESID, DEL, FLOW, NXTNODE, totaliter, PARSTAT
      INTEGER REFPRICE, PRJ, XP, REFPRJ, PRICEJ, THRESHSURPLUS,
Sfactor
      INTEGER NXTQUEUE (1)
      INTEGER STARTN (1), ENDN (1), L (1), F (1), EXCEDENTE (1), FIN
(1), FOUT (1)
      INTEGER NXTIN (1), NXTOU (1), P (1)
      INTEGER FPUSHF (1), NXTPUSHF (1), FPUSHB (1), NXTPUSHB (1)
      CUSTO INTEIRO (1), STATUS (1)

      Common / escalares / N, NA, GRANDES, fator, EPS,
     & THRESHSURPLUS, Sfactor
      COMUM / STATS / NCYC, Niter, totaliter
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / UBOUND / U
      COMUM / FLUXO / F
      Common / PREÇOS / P
      COMUM / BLK3 / excedente
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      COMUM / BLK11 / FPUSHF
      COMUM / BLK12 / NXTPUSHF
      COMUM / BLK13 / FPUSHB
      COMUM / BLK14 / NXTPUSHB
      COMUM / CUSTO / COST
      COMUM / Fila / NXTQUEUE
      COMUM / STATUS / STATUS


C
C REINICIAR CONTAGEM DO NÚMERO DE CIMA ITERATIONS REALIZADAS
```

```
C
      Niter = 0
      Totaliter = 0
      NCYC = 0
      Maxprice = GRANDE
C
C reduzir as capacidades ARC
C
         DO 40 NODE = 1, N
            CAPOUT = 0
            ARC = FOUT (nó)
41 IF (ARC.GT.0) THEN
               CAPOUT = MIN (grande, CAPOUT + U (ARC))
               ARC = NXTOU (ARC)
               IR PARA 41
            END IF
            CAPOUT = MIN (grande, CAPOUT-EXCEDENTE (nó))
            IF (CAPOUT.LT.0) GOTO 400
            Capin = 0
            ARC = FIN (nó)
43 IF (ARC.GT.0) THEN
               IF (U (ARC) .gt. CAPOUT) THEN
                  U (ARC) = CAPOUT
               ENDIF
               Capin = MIN (grande, Capin + U (ARC))
               ARC = NXTIN (ARC)
               IR PARA 43
            END IF
            Capin = MIN (grande, Capin + excedente (nó))
            IF (CAPIN.LT.0) GOTO 400
            ARC = FOUT (nó)
45 IF (ARC.GT.0) THEN
               IF (U (ARC) .gt. Capin) THEN
                  U (ARC) = Capin
               ENDIF
               ARC = NXTOU (ARC)
               IR PARA 45
            END IF
40 CONTINUAR
C
C SET ARC FLUXO PARA SATISFAZER E-CS
C
       FAZER 49 ARC = 1, NA
         START = STARTN (ARC)
         END = ENDN (ARC)
         Pinicial = P (START)
         PEND = P (END)
         IF (PSTART.GE.PEND + COST (ARC) + EPS) THEN
           EXCEDENTE (START) = excedente (START) -U (ARC)
           EXCEDENTE (END) = excedente (END) + U (ARC)
           F (ARC) = U (ARC)
         MAIS
           F (ARC) = 0
         END IF
49 CONTINUAR

C *********** início de uma nova fase ESCALA ***********

60 CONTINUAR

C ***** ***** INITIALIZATION FILA
```

```
        FAZER 82 NODE = 1, N-1
            NXTQUEUE (nó) = NODE + 1
82 CONTINUAR
        NXTQUEUE (N) = 1
        I = 1
        PrevNode = N
        LASTQUEUE = N

C Ajuste o parâmetro de estado para permitir que um NEG. ITERATION
EXCEDENTE

      PARSTAT = 0

C ************ iniciar um novo ciclo de até ITERATIONS ************

100 CONTINUAR

C passamos para o próximo nó (Nó I) para a iteração

        SURPI = excedente (I)
        IF (SURPI.GT.THRESHSURPLUS) THEN

C & ARCF ARCB são os valores atuais dos ARCS de partida da
Aperte LISTAS DE I

           Totaliter = totaliter + 1
           ARCF = FPUSHF (I)
           ARCB = FPUSHB (I)
           STATUS (I) = STATUS (I) +1

           PREÇO = P (I)

115 IF ((ARCF .gt. 0) .OR. (ARCB .gt. 0)) THEN

C começar por tentar afastar FLUXO EM ARCS que foram
C admissível a final da última iteração (SE HOUVER)
C neste nodo. WE deve verificar se eles ainda estão
C admissível.


120 IF ((SURPI .gt. 0) .E. (ARCF .gt. 0)) THEN
                  J = ENDN (ARCF)
                  IF (PREÇO-P (J) -Custo (ARCF) .EQ.EPS) THEN
                     RESID = L (ARCF) - F (ARCF)
                     IF (RESID.GT.0) THEN
                      IF (EXCEDENTE (J) .LT.0) PARSTAT = PARSTAT + 1
                      IF (SURPI .GE. RESID) THEN
                         F (ARCF) = L (ARCF)
                         SURPI = SURPI - RESID
                         EXCEDENTE (J) = excedente (J) + RESID
                         ARCF = NXTPUSHF (ARCF)
                      MAIS
                         F (ARCF) = F (ARCF) + SURPI
                         EXCEDENTE (J) = excedente (J) + SURPI
                         SURPI = 0
                      ENDIF
                      IF (ABS (excedente (J)). GT.THRESHSURPLUS) THEN
                       IF (NXTQUEUE (J) .EQ.0) THEN
                        NXTQUEUE (PrevNode) = J
                        NXTQUEUE (J) = I
```

```
                        PrevNode = J
                       END IF
                      END IF
                    MAIS
                      ARCF = NXTPUSHF (ARCF)
                    ENDIF
                  MAIS
                     ARCF = NXTPUSHF (ARCF)
                  ENDIF
                  GOTO 120
                ENDIF

121 IF ((SURPI .gt. 0) .E. (ARCB .gt. 0)) THEN
              J = STARTN (ARCB)
              IF (PREÇO-P (J) + COST (ARCB) .EQ.EPS) THEN
                 RESID = F (ARCB)
                 IF (RESID.GT.0) THEN
                    IF (EXCEDENTE (J) .LT.0) PARSTAT = PARSTAT + 1
                    IF (SURPI .GE. RESID) THEN
                       SURPI = SURPI - RESID
                       EXCEDENTE (J) = excedente (J) + RESID
                       F (ARCB) = 0
                       ARCB = NXTPUSHB (ARCB)
                    MAIS
                       F (ARCB) = F (ARCB) - SURPI
                       EXCEDENTE (J) = excedente (J) + SURPI
                       SURPI = 0
                    ENDIF
                    IF (ABS (excedente (J)). GT.THRESHSURPLUS) THEN
                     IF (NXTQUEUE (J) .EQ.0) THEN
                      NXTQUEUE (PrevNode) = J
                      NXTQUEUE (J) = I
                      PrevNode = J
                     END IF
                    END IF
                   MAIS
                     ARCB = NXTPUSHB (ARCB)
                   ENDIF
                MAIS
                   ARCB = NXTPUSHB (ARCB)
                ENDIF
                GOTO 121
             ENDIF
          ENDIF

C ***** FIM DE D-EMPURRA; VERIFIQUE SE PREÇO RISE É NECESSÁRIO *****

         IF ((ARCF .EQ. 0) .E. (ARCB .EQ. 0)) THEN

C ****************** FAZER UM AUMENTO DE PREÇO *********************

                REFPRICE = PREÇO
                PREÇO = GRANDE
                Niter = Niter + 1
                ARC = FOUT (I)
111 SE (ARC .gt. 0) THEN
                 IF (F (ARC) .lt. U (ARC)) THEN
                    XP = P (ENDN (ARC)) + COST (ARC)
                    IF (XP.LT.PRICE) THEN
                       PREÇO = XP
                       ARCF = ARC
```

```
                        NXTPUSHF (ARC) = 0
                     MAIS
                      IF (XP.EQ.PRICE) THEN
                        NXTPUSHF (ARC) = ARCF
                        ARCF = ARC
                      END IF
                     ENDIF
                  ENDIF
                  ARC = NXTOU (ARC)
                  GOTO 111
               ENDIF
               ARC = FIN (I)

112 SE (ARC .gt. 0) THEN
               IF (F (ARC) .gt. 0) THEN
                  XP = P (STARTN (ARC)) - COST (ARC)
                  IF (XP.LT.PRICE) THEN
                     PREÇO = XP
                     ARCB = ARC
                     ARCF = 0
                     NXTPUSHB (ARC) = 0
                  MAIS
                   IF (XP.EQ.PRICE) THEN
                     NXTPUSHB (ARC) = ARCB
                     ARCB = ARC
                   END IF
                  ENDIF
               ENDIF
               ARC = NXTIN (ARC)
               GOTO 112
            ENDIF

C Se o preço = GRANDE, a lista TOQUE É PROVÁVEL VAZIO, preço tão
definida como pelo menos THE
C NÍVEL DO MELHOR ARC

         IF (PRICE.EQ.LARGE) THEN
              ARCF = 0
              ARCB = 0

              ARC = FOUT (I)
211 IF (ARC.GT.0) THEN
               XP = P (ENDN (ARC)) + COST (ARC)
               IF (XP.LT.PRICE) THEN
                  PREÇO = XP
               MAIS
                  PRINT *, 'PREÇO DO', I, "excedeu o INT. GAMA '
                  PAUSA
                  PARE
               ENDIF
               ARC = NXTOU (ARC)
               GOTO 211
            ENDIF
            ARC = FIN (I)

212 IF (ARC.GT.0) THEN
               XP = P (STARTN (ARC)) - COST (ARC)
               IF (XP.LT.PRICE) THEN
                  PREÇO = XP
               MAIS
                  PRINT *, 'PREÇO DO', I, "excedeu o INT. GAMA '
```

```
                    PAUSA
                    PARE
                  ENDIF
                  ARC = NXTIN (ARC)
                  GOTO 212
                ENDIF

                IF (SURPI.GT.0) GOTO 400
                IF (PRICE.LT.INT (GRANDE / 10)) PREÇO = INT (GRANDE /
10)

           END IF



C SET PREÇO do nó i

           Preço = Preço + EPS
           P (I) = PREÇO
           FPUSHF (I) = ARCF
           FPUSHB (I) = ARCB

C FIM DO PREÇO RISE; SE HÁ AINDA UM LOTE DE EXCEDENTE,
C tentar fazer tanto insistir.

           IF (SURPI.GT.THRESHSURPLUS) GOTO 115

        MAIS

C SE NO PREÇO RISE OCORREU REAJUSTE O início do LISTAS DE PRESSÃO

           FPUSHF (I) = ARCF
           FPUSHB (I) = ARCB
        ENDIF

C Execute a escrituração FINAL da iteração


        EXCEDENTE (I) = SURPI

C Se o preço é muito alto, então algo está errado

X IF (PRICE.GT.MAXPRICE) THEN
X GO TO 400
X ENDIF

       ENDIF


C Verifique se há um ITERATION EXCEDENTE NEGATIVO

       IF (SURPI.LT (-. THRESHSURPLUS)) THEN

           Totaliter = totaliter + 1
           ARCF = FPUSHF (I)
           ARCB = FPUSHB (I)

           PREÇO = P (I)

1115 IF ((ARCF .gt. 0) .OR. (ARCB .gt. 0)) THEN
```

```
C começar por tentar afastar FLUXO EM ARCS que foram
C admissível a final da última iteração (SE HOUVER)
C neste nodo. WE deve verificar se eles ainda estão
C admissível.


1120 IF ((SURPI .lt. 0) .E. (ARCF .gt. 0)) THEN
                  J = ENDN (ARCF)
                  IF (PREÇO-P (J) -Custo (ARCF) .EQ (-. EPS)) THEN
                    RESID = F (ARCF)
                    IF (RESID.GT.0) THEN
                     IF (EXCEDENTE (J) .GT.0) PARSTAT = PARSTAT + 1
                     IF ((-SURPI) .GE. RESID) THEN
                        F (ARCF) = 0
                        SURPI = SURPI + RESID
                        EXCEDENTE (J) = excedente (J) - RESID
                        ARCF = NXTPUSHF (ARCF)
                     MAIS
                        F (ARCF) = F (ARCF) + SURPI
                        EXCEDENTE (J) = excedente (J) + SURPI
                        SURPI = 0
                     ENDIF
                     IF (ABS (excedente (J)). GT.THRESHSURPLUS) THEN
             IF ((NXTQUEUE (J) .EQ.0) .E. (STATUS (J) .LE. (PARSTAT /
2))) THEN
                        NXTQUEUE (PrevNode) = J
                        NXTQUEUE (J) = I
                        PrevNode = J
                      END IF
                     END IF
                    MAIS
                     ARCF = NXTPUSHF (ARCF)
                    ENDIF
                  MAIS
                     ARCF = NXTPUSHF (ARCF)
                  ENDIF
                  GOTO 1120
                ENDIF

1121 IF ((SURPI .lt. 0) .E. (ARCB .gt. 0)) THEN
               J = STARTN (ARCB)
               IF (PREÇO-P (J) + COST (ARCB) .EQ (-. EPS)) THEN
                  RESID = U (ARCB) -F (ARCB)
                  IF (RESID.GT.0) THEN
                     IF (EXCEDENTE (J) .GT.0) PARSTAT = PARSTAT + 1
                     IF ((-SURPI) .GE. RESID) THEN
                        SURPI = SURPI + RESID
                        EXCEDENTE (J) = excedente (J) - RESID
                        F (ARCB) = L (ARCB)
                        ARCB = NXTPUSHB (ARCB)
                     MAIS
                        F (ARCB) = F (ARCB) - SURPI
                        EXCEDENTE (J) = excedente (J) + SURPI
                        SURPI = 0
                     ENDIF
                     IF (ABS (excedente (J)). GT.THRESHSURPLUS) THEN
           IF ((NXTQUEUE (J) .EQ.0) .E. (STATUS (J) .LE. (PARSTAT /
2))) THEN
                       NXTQUEUE (PrevNode) = J
                       NXTQUEUE (J) = I
                       PrevNode = J
```

```
                    END IF
                  END IF
                MAIS
                  ARCB = NXTPUSHB (ARCB)
                ENDIF
              MAIS
                 ARCB = NXTPUSHB (ARCB)
              ENDIF
              GOTO 1121
            ENDIF
          ENDIF

C ***** FIM DE D-EMPURRA; VERIFIQUE SE PREÇO RISE É NECESSÁRIO *****

        IF ((ARCF .EQ. 0) .E. (ARCB .EQ. 0)) THEN

C ****************** FAZER UM AUMENTO DE PREÇO *********************

               REFPRICE = PREÇO
               PREÇO = -Grande
               Niter = Niter + 1
               ARC = FOUT (I)
1111 IF (ARC .gt. 0) THEN
                IF (F (ARC) .gt. 0) THEN
                   XP = P (ENDN (ARC)) + COST (ARC)
                   IF (XP.GT.PRICE) THEN
                      PREÇO = XP
                      ARCF = ARC
                      NXTPUSHF (ARC) = 0
                   MAIS
                    IF (XP.EQ.PRICE) THEN
                      NXTPUSHF (ARC) = ARCF
                      ARCF = ARC
                    END IF
                   ENDIF
                ENDIF
                ARC = NXTOU (ARC)
                GOTO 1111
              ENDIF
              ARC = FIN (I)

1112 IF (ARC .gt. 0) THEN
               IF (F (ARC) .lt. U (ARC)) THEN
                  XP = P (STARTN (ARC)) - COST (ARC)
                  IF (XP.GT.PRICE) THEN
                     PREÇO = XP
                     ARCB = ARC
                     ARCF = 0
                     NXTPUSHB (ARC) = 0
                  MAIS
                   IF (XP.EQ.PRICE) THEN
                     NXTPUSHB (ARC) = ARCB
                     ARCB = ARC
                   END IF
                  ENDIF
               ENDIF
               ARC = NXTIN (ARC)
               GOTO 1112
            ENDIF
```

```
C Se o preço = -Grande, a lista IMPULSO é provável vazio, assim
definir o preço ao máximo de THE
C NÍVEL DO MELHOR ARC

           IF (PRICE.EQ.-GRANDE) THEN
                ARCF = 0
                ARCB = 0

                ARC = FOUT (I)
1211 IF (ARC .gt. 0) THEN
                  XP = P (ENDN (ARC)) + COST (ARC)
                  IF (XP.GT.PRICE) THEN
                   PREÇO = XP
                  MAIS
                   IF (XP.LE.-GRANDE) THEN
                    PRINT *, 'PREÇO DO', I, "excedeu o INT. GAMA '
                    PAUSA
                    PARE
                   END IF
                  END IF
                  ARC = NXTOU (ARC)
                  GOTO 1211
                END IF
                ARC = FIN (I)

1212 IF (ARC .gt. 0) THEN
                 XP = P (STARTN (ARC)) - COST (ARC)
                 IF (XP.GT.PRICE) THEN
                   PREÇO = XP
                 MAIS
                   IF (XP.LE.-GRANDE) THEN
                    PRINT *, 'PREÇO DO', I, "excedeu o INT. GAMA '
                    PAUSA
                    PARE
                   END IF
                 END IF
                 ARC = NXTIN (ARC)
                 GOTO 1212
                ENDIF

                IF (SURPI.LT.0) GOTO 400
                IF (PRICE.GT.-INT (GRANDE / 10)) PREÇO = -INT (GRANDE
/ 10)

           END IF



C SET PREÇO do nó i

           Preço = PREÇO-EPS
           P (I) = PREÇO
           FPUSHF (I) = ARCF
           FPUSHB (I) = ARCB

C FIM DO PREÇO RISE; SE HÁ AINDA UM LOTE DE EXCEDENTE,
C tentar fazer tanto insistir.

           IF (SURPI.LT (-. THRESHSURPLUS)) GOTO 1115

        ENDIF
```

```
C Execute a escrituração FINAL da iteração


         EXCEDENTE (I) = SURPI

       END IF



C ********** FIM DE PARTIDA ITERATION no nó i *********

C de seleção para o final da fila. Este passo não é NECESSÁRIO PARA
C ALGORITMO; É INCLUÍDO de recolha de estatísticas SOBRE
COMPORTAMENTO DO C algoritmo do, EG CONTANDO O número de ciclos

         IF (I.EQ.LASTQUEUE) THEN
          LASTQUEUE = PrevNode
          NCYC = + 1 NCYC
X PRINT *, 'CICLO #', NCYC, '# dos aumentos de preços ", Niter
         END IF

C VERIFICAÇÃO DE RESCISÃO DE ESCALA FASE. Se o escalonamento fase é
NÃO TERMINOU C, o avanço da fila e voltar para levar outro nó.

        NXTNODE = NXTQUEUE (I)
        IF (I.NE.NXTNODE) THEN
          NXTQUEUE (I) = 0
          NXTQUEUE (PrevNode) = NXTNODE
          I = NXTNODE
          Ir para 100
        END IF

C ************** FIM DE subproblema (escalonamento FASE)
**************

X PRINT *, 'FIM DE ESCALA FASE'

C Execute uma verificação de diagnóstico

X = 0 LN
X FAZER 500 NÓ = 1, N
X IF (EXCEDENTE (nó) .NE.0) THEN
X = LN LN + 1
X ENDIF
500 CONTINUAR
X PRINT *, "excedente diferente de zero AT ', LN," nós "
X FAZER 600 ARC = 1, NA
X IF (F (ARC) .GT.0) THEN
X IF (P (STARTN (ARC).) - P (ENDN (ARC)) o tenente-EPS + COST (ARC))
THEN
X PRINT *, 'E-CS VIOLADOS AT ARC ", ARC
X ENDIF
X ENDIF
X IF (F (ARC) .LT.U (ARC)) THEN
X IF (P (STARTN (ARC)) - P (ENDN (ARC)) GT.EPS + COST (ARC).) THEN
X PRINT *, 'E-CS VIOLADOS AT ARC ", ARC
X ENDIF
X ENDIF
600 CONTINUAR
```

```
X PRINT *, 'FIM DE VERIFICAÇÃO DE FASE OLD'

C ****** SE EPSILON É uma RESCINDIR; MAIS REDUZIR EPSILON ******

        IF (EPS.EQ.1) THEN
           RETURN
        MAIS
           EPS = INT (EPS / FACTOR)
           IF (EPS.LT.1) EPS = 1
        END IF
        THRESHSURPLUS = INT (THRESHSURPLUS / Sfactor)
        IF (EPS.EQ.1) THRESHSURPLUS = 0

C redefinir o FLUXO & LISTAS DE PRESSÃO

       FAZER 800 NÓ = 1, N
         FPUSHF (nó) = 0
         FPUSHB (nó) = 0
         STATUS (nó) = 0
800 CONTINUAR

       FAZER 900 ARC = 1, NA
         START = STARTN (ARC)
         END = ENDN (ARC)
         Pinicial = P (START)
         PEND = P (END)
         IF (PSTART.GT.PEND + EPS + COST (ARC)) THEN
           RESID = U (ARC) -F (ARC)
           IF (RESID.GT.0) THEN
             EXCEDENTE (START) = excedente (START) -RESID
             EXCEDENTE (END) = excedente (END) + RESID
             F (ARC) = U (ARC)
           END IF
         MAIS
          IF (PSTART.LT.PEND-EPS + COST (ARC)) THEN
            FLUXO = F (ARC)
            IF (FLOW.GT.0) THEN
              EXCEDENTE (START) = excedente (START) + FLUXO
              EXCEDENTE (END) = excedente (END) -FLOW
              F (ARC) = 0
            END IF
          END IF
         END IF
900 CONTINUAR


C voltar para outra FASE

X PRINT *, "VERIFICAÇÃO E-CS ANTES DE NOVA FASE '
X FAZER 700 ARC = 1, NA
X IF (F (ARC) .GT.0) THEN
X IF (P (STARTN (ARC).) - P (ENDN (ARC)) o tenente-EPS + COST (ARC))
THEN
X PRINT *, 'E-CS VIOLADOS AT ARC ", ARC
X ENDIF
X ENDIF
X IF (F (ARC) .LT.U (ARC)) THEN
X IF (P (STARTN (ARC)) - P (ENDN (ARC)) GT.EPS + COST (ARC).) THEN
X PRINT *, 'E-CS VIOLADOS AT ARC ", ARC
X ENDIF
X ENDIF
```

```
700 CONTINUAR
       IR PARA 60

400 PRINT *, "o problema é inviável '
       PAUSA
       PARE
       FIM


SLF

Pequena etiqueta First (SLF) de código para encontrar o caminho mais
curto de um
origem para todos os destinos, de acordo com o jornal "A etiqueta
simples e rápida Corrigindo
Método para caminhos mais curtos, "Redes, Vol. 23, 1993, pp. 703-709,
por DP
Bertsekas.

C ********************************************** ***********
C SLF algoritmo para encontrar o caminho mais curto DE UMA ORIGEM para
todos os destinos
C ********************************************** ***********
C
C Todos os parâmetros são INTEIRO
C
C a única máquina CONSTANTE dependente usada É INF
C
C **********************************************
**********************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      INTEGER FOUT, NXTOUT, D, P, Q, R, HP, Y, X, T2, DEST, COUNT
      TT1 REAL, TT2, TCOST
      DIMENSÃO FOUT (maxNodes), D (maxNodes), P (maxNodes), HP
(maxNodes)
      DIMENSÃO Q (maxNodes)
      DIMENSÃO LNGT (MAXARCS), ND (MAXARCS), NXTOUT (MAXARCS)
      DATA INF / 999999999 /

      CHAMADA READ (N, M, FOUT, NXTOUT, ND, LNGT)

      FAZER 10 I = 1, N
        Q (I) = 0
        P (I) = 0
10 CONTINUAR
C
C REINICIAR ORIGEM E FILA DE DESTINOS
C

      PRINT *, "o número de nós IS = ', N
      PRINT *, "ENTER o nó origem»
      LEIA *, R



      PRINT *, "CHAMANDO SLF algoritmo para resolver o problema"
      PRINT *, "****************************************** '
C
```

```
C GET hora de início para o Mac II
C
      TT1 = longa (362) /60.0
C

        CHAME SLF (FOUT, NXTOUT, ND, LNGT, D, P, Q, N, INF, R, COUNT)
C
C GET TÉRMINO TEMPO PARA O MAC II
C
      TT2 = longa (362) /60.0 - TT1
      PRINT *, "acabado --- tempo de CPU TOTAL ', TT2,' s '
      PRINT *, "****************************************** '
      PRINT *, "O nó ORIGEM É ', R
C

      PRINT *, 'número de iterações', COUNT
      PRINT *, 'SH. DIST. De destino », N, 'é', D (N)

45 PRINT *, "entrar em outros nó de destino (0 se for feito)"
      LEIA *, DEST
      IF ((DEST.LT.0) .OR. (DEST.GT.N)) GOTO 45
      IF (DEST.EQ.R) GOTO 45
      IF (DEST.NE.0) THEN
        PRINT *, 'distância mais curta para', DEST, '=', D (DEST)
        GOTO 45
      END IF



      PARE
      FIM
C
C
      SUBROUTINE SLF (FOUT, NXTOUT, ND, LNGT, D, P, Q, N, INF, R,
COUNT)
C ********************************************** *****************
C
C ROTINA SLF
C
C SLF algoritmo baseado no papel:
C "Um algoritmo SIMLPLE E RÓTULO RÁPIDO corrigindo para caminhos mais
curtos"
C NETWORKS, VOL. 23, 1993, PP. 703-709, BY
C DIMITRI P. Bertsekas
C
C ********************************************** *****************
C 1) encontra um MENOR caminho árvore com raiz em NODE R eo menor
C DISTÂNCIAS
C 2) é baseado no método de SLF com o conjunto Q REALIZAR
C AS uma única fila Q (.)
C
C significado dos parâmetros de entrada:
C
C FOUT (I) = POINTER de Arco-LIST do nó i, i = 1,2, ..., N + 1
C ND (J) = TÉRMINO NÓ DE ARC J, J = 1,2, ..., M
C LNGT (J) = comprimento do arco J, J = 1,2, ..., M
C NMAX = dimensão do ARRAYS A (.), D (.), P (.), Q (.)
C Mmáx = dimensão do ARRAYS ND (.), LNGT (.)
CN = número de nós
C INF = MUITO GRANDE VALOR INTEIRO (INFINITY)
CR = ROOT
```

```
C
C significado dos parâmetros de saída:
C
CD (I) = distância mais curta de R a I, I = 1,2, ..., N
CP (I) = antecessor nó I no menor caminho árvore, I = 1,2, ..., N
C
C DOS PARÂMETROS interna principal:
C
CQ (I) = lista de nós candidatos; Q (I) = -1 SE EU NÃO ESTÁ EM Q e tem
C JÁ sido digitalizados
C = 0 SE EU NÃO ESTÁ EM Q e tem
C não foi digitalizado
C = J IF I PRECEDE NÓ NA J
LISTA C
C = NN N + 1
CU = nó atual
CV = ENDING do nó da ARC CURRENT
C INIT = START-ponteiro para a ARC-LIST do nó atual
C IFIN = END-ponteiro para a ARC-LIST do nó atual
C DV = LABEL TENTATIVA DE NÓ V
C LAST = ponteiro para o último nó Q (.)
C
C Todos os parâmetros são INTEIRO
C
C ********************************************** ******************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      INTEGER FOUT, NXTOUT, D, P, Q, R, U, V, DV, COUNT, primeiro,
segundo, terceiro
      DIMENSÃO FOUT (maxNodes), D (maxNodes), P (maxNodes), Q
(maxNodes)
      DIMENSÃO ND (MAXARCS), LNGT (MAXARCS), NXTOUT (MAXARCS)
C
C INITIALIZE
C
      FAZER 10 I = 1, N
        Q (I) = 0
        D (I) = INF
  10 CONTINUAR
      Q (R) = - 1
      D (R) = 0
      P (R) = 0
      NN = N + 1
      FIRST = NN
      LAST = NN
      COUNT = 0
      U = R
C
C
C EXPLORAR A ESTRELA DE FRENTE U
C
20 CONTINUAR


      COUNT = COUNT + 1
      J = FOUT (L)
25 IF (J.GT.0) THEN
        V = ND (J)
        DV = D (L) + LNGT (J)
```

```
C
C Verifique se o rótulo de V PODE SER MELHORADO
C
        IF (D (V) .gt. DV) THEN
          D (V) = DV
          P (V) = L
          IF (Q (V)) 30,30,50
C
C IF V NÃO ESTÁ EM Q e com a etiqueta é menor do que o rótulo do nó
superior,
C está inserida NO INÍCIO DA Q
C
30 IF (LAST.EQ.NN) THEN
           Q (V) = NN
           FIRST = V
           LAST = V
           IR PARA 50
          END IF

           IF (DV.LE.D (FIRST)) THEN
             Q (V) = PRIMEIRA
             FIRST = V
           MAIS
             Q (LAST) = V
             Q (V) = NN
             LAST = V
           END IF

50 CONTINUAR
         END IF
         J = NXTOUT (J)
         GOTO 25
       END IF
C
C Remova a nova corrente NODE U
C
60 U = PRIMEIRA
       FIRST = Q (U)
       Q (U) = - 1
       IF (LAST .EQ. U) LAST = NN
C
C Verifique se a lista estiver vazia
C
       IF (U .LE. N) Vá para 20

       RETURN
       FIM

      SUBROUTINE READ (N, M, FOUT, NXTOUT, END, LNGT)
C ***********************************************
******************
C lê os dados do gráfico (armazenado como uma lista adjacence) e as
origens
LISTA C.
C ********************************************** ******************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      Implícita INTEGER (AZ)
      DIMENSÃO FOUT (maxNodes), LAST (maxNodes)
```

```
      DIMENSÃO START (MAXARCS), END (MAXARCS), LNGT (MAXARCS)
      DIMENSÃO NXTOUT (MAXARCS)
C
      PRINT *, "leitura mais curto DATA PATH problema"
      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' velho ')
      Rewind (13)

C LEIA número de nós e arcos

      LER (13,1010) N, M

C LEIA ENDNODE eo comprimento de cada ARC

      FAZER 20 I = 1, M
        READ (13,1020) START (I), END (I), LNGT (I), MANEQUIM
20 CONTINUAR

      FAZER 30 I = 1, N
        READ (13,1000) MANEQUIM
30 CONTINUAR

      ENDFILE (13)
      Rewind (13)

      PRINT *, 'FIM DE LEITURA'

1000 FORMATO (1I8)
1010 FORMATO (2I8)
1020 FORMATO (4I8)


      PRINT *, "reestruturar os dados '
C
C GERAR A FRENTE E ESTRELAS com versões anteriores para cada nó
C
      FAZER 60 I = 1, N
        FOUT (I) = 0
        LAST (I) = 0
60 CONTINUAR

      FAZER 65 ARC = 1, M
        NXTOUT (ARC) = 0
        NÓ = START (ARC)
        IF (FOUT (nó) .NE.0) THEN
          NXTOUT (LAST (nó)) = ARC
        MAIS
          FOUT (nó) = ARC
        END IF
        LAST (nó) = ARC
65 CONTINUAR

      RETURN

      FIM


SLFT

Combinação de pequeno selo primeiro código (SLF) com o algoritmo de
limiar para encontrar
```

caminho mais curto a partir de uma origem para todos os destinos, de acordo com o jornal "A Simple e
Etiqueta Rápido Correção Método para caminhos mais curtos, "Redes, Vol. 23, 1993, pp.
703-709, por DP Bertsekas.

```
C ********************************************** ***********
C COMBINADO SLF-LIMIAR algoritmo para encontrar caminhos mais curtos
C DE UMA ORIGEM para todos os destinos
C ********************************************** ***********
C
C Todos os parâmetros são INTEIRO
C
C a única máquina CONSTANTE dependente usada É INF
C
C **********************************************
**********************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      INTEGER FOUT, NXTOUT, D, P, HP, Y, X, T2, DEST, COUNT, CC
      INTEIRO A, Q1, Q2, T, R
      TT1 REAL, TT2, TCOST
      DIMENSÃO FOUT (maxNodes), D (maxNodes), P (maxNodes)
      DIMENSÃO Q1 (maxNodes), Q2 (maxNodes)
      DIMENSÃO LNGT (MAXARCS), ND (MAXARCS), NXTOUT (MAXARCS)
      DATA INF / 999999999 /


      CHAMADA READ (N, M, FOUT, NXTOUT, ND, LNGT)

C LMAX é o comprimento máximo ARC utilizados na definição da LIMIAR



      LMAX = -INF

      FAZER 5 ARC = 1, M
        IF (LNGT (ARC) .GT.LMAX) LMAX = LNGT (ARC)
5 CONTINUAR


      CHAME THINCR (N, M, LMAX, T)

      FAZER 10 I = 1, N
        P (I) = 0
        Q1 (I) = 0
        Q2 (I) = 0
10 CONTINUAR
C
C REINICIAR ORIGEM E FILA DE DESTINOS
C

      PRINT *, "o número de nós IS = ', N
      PRINT *, "ENTER o nó origem»
      LEIA *, R
```

```
      PRINT *, "CHAMANDO SLF-LIMIAR algoritmo para resolver o
problema"
      PRINT *, '******************************************** *** '
C
C GET hora de início para o Mac II
C
      TT1 = longa (362) /60.0
C

        CHAME SLFT (FOUT, NXTOUT, ND, LNGT, D, P, Q1, Q2, N, INF, T,
R, COUNT, CC)
C
C GET TÉRMINO TEMPO PARA O MAC II
C
      TT2 = longa (362) /60.0 - TT1
      PRINT *, "acabado --- tempo de CPU TOTAL ', TT2,' s '
      PRINT *, '******************************************** *** '
      PRINT *, "O nó ORIGEM É ', R
C

      PRINT *, 'número de iterações', COUNT
      PRINT *, "número de repartições", CC
      PRINT *, 'SH. DIST. De destino », N, 'é', D (N)

45 PRINT *, "entrar em outros nó de destino (0 se for feito)"
      LEIA *, DEST
      IF ((DEST.LT.0) .OR. (DEST.GT.N)) GOTO 45
      IF (DEST.EQ.R) GOTO 45
      IF (DEST.NE.0) THEN
        PRINT *, 'distância mais curta para', DEST, '=', D (DEST)
        GOTO 45
      END IF



      PARE
      FIM
C
C



      SUBROUTINE SLFT (FOUT, NXTOUT, ND, LNGT, D, P, Q1, Q2, N, INF,
T, R, COUNT, CC)
C ********************************************** *****************
C
C SLFT ROTINA
C
C COMBINADO SLF-LIMIAR algoritmo baseado no papel:
C "Um algoritmo SIMLPLE E RÓTULO RÁPIDO corrigindo para caminhos mais
curtos"
C NETWORKS, VOL. 23, 1993, PP. 703-709, BY
C DIMITRI P. Bertsekas
C
C ********************************************** *****************
C 1) encontra um MENOR caminho árvore com raiz em NODE R eo menor
C DISTÂNCIAS
C 2) é baseado no método de SLF com o conjunto Q
C implementado como um PAR DE LISTAS: (.) Q1 AS uma fila e
C Q2 (.) Como uma lista encadeada. A partição é baseado em um
C valor limite
```

```
C
C significado dos parâmetros de entrada:
C
C FOUT (I) = POINTER de Arco-LIST do nó i, i = 1,2, ..., N + 1
C ND (J) = TÉRMINO NÓ DE ARC J, J = 1,2, ..., M
C LNGT (J) = comprimento do arco J, J = 1,2, ..., M
C NMAX = dimensão do ARRAYS A (.), D (.), P (.), Q (.)
C Mmáx = dimensão do ARRAYS ND (.), LNGT (.)
CN = número de nós
C INF = MUITO GRANDE VALOR INTEIRO (INFINITY)
CR = ROOT
C
C significado dos parâmetros de saída:
C
CD (I) = distância mais curta de R a I, I = 1,2, ..., N
CP (I) = antecessor nó I no menor caminho árvore, I = 1,2, ..., N
C
C significado dos parâmetros MAIN INTERNOS:
C
C Q1 (I) = lista de nós CANDIDATOS Q1 (I) = 0 SE EU NÃO ESTÁ EM Q1 (.)
C que tenham a sua LABEL LESS THAN = J IF I precede nó j
C ou igual ao atual na lista
C LIMIAR
C
C Q2 (I) = lista de outros Q2 CANDIDATO (I) = 0 SE EU NÃO ESTÁ EM Q2
(.)
C nós e cópias antigas de nós = J IF I PRECEDE nó j
C inserido no Q1 (.) Na lista
C
C = NN N + 1
CU = nó atual
CV = ENDING do nó da ARC CURRENT
C INIT = START-ponteiro para a ARC-LIST do nó atual
C IFIN = END-ponteiro para a ARC-LIST do nó atual
C DV = LABEL TENTATIVA DE NÓ V
C LAST = ponteiro para o último nó Q (.)
C PNTR = ponteiro para o último nó da fila PRIMEIRO DE Q (.)
C
C Todos os parâmetros são INTEIRO
C
C ********************************************** ******************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      INTEGER Q1, Q2, T, THRS, T1, PRIMEIRA
      INTEGER FOUT, NXTOUT, D, P, R, U, V, DV, COUNT, CC, LAST, LAST2
      DIMENSÃO FOUT (maxNodes), D (maxNodes), P (maxNodes)
      DIMENSÃO ND (MAXARCS), LNGT (MAXARCS), NXTOUT (MAXARCS)
      DIMENSÃO Q1 (maxNodes), Q2 (maxNodes)
C
C INITIALIZE
C
      FAZER 10 I = 1, N
        D (I) = INF
10 CONTINUAR
      D (R) = 0
      P (R) = 0
      NN = N + 1
      Q1 (NN) = NN
      Q2 (NN) = NN
```

```
      THRS = -1
      LAST = NN
      LAST2 = NN
      COUNT = 0
      CC = 0
      U = R
C
C EXPLORAR A ESTRELA DE FRENTE U
C
20 CONTINUAR

      COUNT = COUNT + 1
      J = FOUT (L)
25 IF (J.GT.0) THEN
        V = ND (J)
        DV = D (L) + LNGT (J)
C
C Verifique se o rótulo de V PODE SER MELHORADO
C
        IF (D (V) .LE. DV) para 50
        IF (DV .gt. THRS) Vá para 30
        IF (Q1 (V) .gt. 0) Ir para 40
C
C INSERÇÃO NO V pé nem cabeça Q1 (.)
C
        IF (LAST.EQ.NN) THEN
          Q1 (V) = NN
          FIRST = V
          LAST = V
          IR PARA 40
         END IF

         IF (DV.LE.D (FIRST)) THEN
          Q1 (V) = PRIMEIRA
          FIRST = V
        MAIS
          Q1 (LAST) = V
          Q1 (V) = NN
          LAST = V
        END IF

        IR PARA 40

30 IF (Q2 (V) .gt. 0) Ir para 40
C
C IF V NÃO ESTÁ EM Q2 (.), Ele é inserido na cabeça ou a cauda do Q2
(.)
C
        IF (LAST2.EQ.NN) THEN
          Q2 (V) = NN
          Q2 (NN) = V
          V = LAST2
          IR PARA 40
         END IF

         IF (DV.LE.D (Q2 (NN))) THEN
          Q2 (V) = Q2 (NN)
          Q2 (NN) = V
        MAIS
          Q2 (LAST2) = V
          Q2 (V) = NN
```

```
          V = LAST2
        END IF

C
C Atualizar o rótulo de V
C

40 D (V) = DV
        P (V) = L

50 CONTINUAR

        J = NXTOUT (J)
        GOTO 25
       END IF
C
C Verifique se Q1 (.) ESTÁ VAZIO
C
60 IF (LAST .EQ. NN) Vá para 80
C
C Remova a nova corrente NODE U da cabeça de Q1 (.)
C
70 U = PRIMEIRA
      FIRST = Q1 (U)
      Q1 (U) = 0
      IF (LAST .EQ. U) LAST = NN
      IR PARA 20
C
C Verifique se TAMBÉM Q2 (.) ESTÁ VAZIO
C
80 IF (LAST2 .EQ. NN) RETURN


C
C calcular o NOVO TENTATIVA limiar T1
C
      MIN = INF
      T1 = THRS + 1 + T
      LAST2 = NN
      I = NN
      J = Q2 (I)
C
C SCAN Q2 (.) A fim de calcular MIN e remover CÓPIAS de nós
C já foi removido
C


90 IF (D (J) .LE. T1) ir para 100
C
ATUALIZAÇÃO C MIN
C
      LAST2 = J
      MIN = MIN0 (MIN, D (J))
      I = J
      IR PARA 110
C
C Retire J DE Q2 (.)
C
Q2 100 (I) = Q2 (J)
      Q2 (J) = 0
C
```

```
C Verifique se J deve ser inserido no Q1 (.)
C
      IF (D (J) .LE. THRS) ir para 110

      IF (LAST.EQ.NN) THEN
        Q1 (J) = NN
        FIRST = J
        LAST = J
        IR PARA 110
       END IF

      IF (D (J) .LE.D (FIRST)) THEN
        Q1 (J) = PRIMEIRA
        FIRST = J
      MAIS
        Q1 (LAST) = J
        Q1 (J) = NN
        LAST = J
      END IF

110 J = Q2 (I)
      IF (J .NE. NN) Vá para 90
C
C Atualizar o valor limite
C
      THRS = T1
C
C Verifique se Q1 (.) Ainda é VAZIO
C
      IF (LAST .NE. NN) THEN
        CC = CC + 1
        IR PARA 70
      END IF
C
C IF Q2 (.) Foi esvaziado em seguida, retornar
C
      IF (LAST2 .EQ. NN) RETURN
C
C aumentar o THRS limiar e digitalize novamente Q2 (.)
C
      THRS = MIN + T
      LAST2 = NN
      I = NN
      J = Q2 (I)

120 SE (D (J) .gt. THRS) ir para 130

C
C Mova J DE Q2 (.) Para Q1 (.)
C

      Q2 (I) = Q2 (J)
      Q2 (J) = 0

      IF (LAST.EQ.NN) THEN
        Q1 (J) = NN
        FIRST = J
        LAST = J
        IR PARA 140
      END IF
```

```
      IF (D (J) .LE.D (FIRST)) THEN
        Q1 (J) = PRIMEIRA
        FIRST = J
      MAIS
        Q1 (LAST) = J
        Q1 (J) = NN
        LAST = J
      END IF

      IR PARA 140

C
C continuam sendo a digitalização de Q2 (.)
C

130 I = J
      LAST2 = J

140 J = Q2 (I)
      IF (J .NE. NN) ir para 120
      CC = CC + 1
      IR PARA 70
      FIM




      SUBROUTINE THINCR (N, M, LMAX, T)
C ***********************************************
**********************
C calcular o incremento do limite, conforme proposto em "F.GLOVER; R.
C GLOVER; D.KLINGMAN, CENTRO DE ESTUDOS cibernético, UNIVERSITY OF
C Texas, em Austin TX, EUA. RELATÓRIO DE PESQUISA CCS 419, Junho de
1981.
C ***********************************************
**********************
      INTEGER T, S

C Escolha X2 = 1,5 para GRID gráficos e X2 = 0,25 para Random GRÁFICOS

      X2 = 0,25

      S = MIN0 (35, M / N)
      ALMAX = LMAX
      T = X2 * ALMAX
      AT = T
      AS = S
      IF (S .gt. 7) T = AT * 7. / AS
      IF (T .LE. 0) T = 1
      RETURN
100 FORMATO (F4.2)
      FIM


      SUBROUTINE READ (N, M, FOUT, NXTOUT, END, LNGT)
C ***********************************************
******************
C lê os dados do gráfico (armazenado como uma lista adjacence) e as
origens
LISTA C.
```

```
C ********************************************** ******************


      PARÂMETROS (maxNodes = 10000, MAXARCS = 100000)

      Implícita INTEGER (AZ)
      DIMENSÃO FOUT (maxNodes), LAST (maxNodes)
      DIMENSÃO START (MAXARCS), END (MAXARCS), LNGT (MAXARCS)
      DIMENSÃO NXTOUT (MAXARCS)
C
      PRINT *, "leitura mais curto DATA PATH problema"
      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' velho ')
      Rewind (13)

C LEIA número de nós e arcos

      LER (13,1010) N, M

C LEIA ENDNODE eo comprimento de cada ARC

      FAZER 20 I = 1, M
        READ (13,1020) START (I), END (I), LNGT (I), MANEQUIM
20 CONTINUAR

      FAZER 30 I = 1, N
        READ (13,1000) MANEQUIM
30 CONTINUAR

      ENDFILE (13)
      Rewind (13)

      PRINT *, 'FIM DE LEITURA'

1000 FORMATO (1I8)
1010 FORMATO (2I8)
1020 FORMATO (4I8)


      PRINT *, "reestruturar os dados '
C
C GERAR A FRENTE E ESTRELAS com versões anteriores para cada nó
C
      FAZER 60 I = 1, N
        FOUT (I) = 0
        LAST (I) = 0
60 CONTINUAR

      FAZER 65 ARC = 1, M
        NXTOUT (ARC) = 0
        NÓ = START (ARC)
        IF (FOUT (nó) .NE.0) THEN
          NXTOUT (LAST (nó)) = ARC
        MAIS
          FOUT (nó) = ARC
        END IF
        LAST (nó) = ARC
65 CONTINUAR

      RETURN

      FIM
```

```
************************************************
************************

Código Leilão de Max-Flow Problemas

************************************************
************************

Algoritmo de Leilão para o problema de fluxo máximo, de acordo com o
papel
"Um leilão Algoritmo para o Max-Flow Problem", JOTA, Vol. 87, 1995,
pp. 69-101, pela DP
Bertsekas.

AUCTION_MAX_FLOW


LEILÃO C ******* / MAX-FLOW CODE *******
C
C contém algumas afirmações de tempo que têm de ser adaptados ao
sistema de destino
C
C ********************************************** **
      PARÂMETROS (maxNodes = 20001, MAXARCS = 120000)
      Implícita INTEGER (AZ)
      COMUM / escalares / N, NA, GRANDE, SOURCE, pia
      COMUM / STATS / Niter
      COMUM / STATS1 / Naug, EXT_ATT, EXT_SUCC, SB_SUCC, NUM_EXT
      COMUM / CHECK / CH_TIME, EXT_TIME, SB_TIME, AUG_TIME, MF_TIME
      COMUM / check1 / IT_TIME, INIT_TIME, CEL_TIME, PL_TIME, TOT_TIME
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / UBOUND / U
      COMUM / FLUXO / F
      Common / PREÇOS / P
      COMUM / BLK3 / excedente
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      COMUM / PRED / PRDARC



      INTEGER STARTN (MAXARCS)
      INTEGER ENDN (MAXARCS)
      INTEGER U (MAXARCS)
      INTEGER F (MAXARCS)
      INTEGER excedente (maxNodes)
      INTEGER FIN (maxNodes)
      INTEGER FOUT (maxNodes)
      INTEGER NXTIN (MAXARCS)
      INTEGER NXTOU (MAXARCS)
      INTEGER P (maxNodes)
      INTEGER PRDARC (maxNodes)
      Real * 8 TT, temporizador, TOTAL, Totalt, AVERAGE_P
```

```
      Real * 8 CH_TIME, EXT_TIME, SB_TIME, AUG_TIME, MF_TIME, IT_TIME
      Real * 8 INIT_TIME, CEL_TIME, PL_TIME, TOT_TIME


      PRINT *, 'ALGORITHM LEILÃO PARA MAX-FLOW'
      PRINT *, "****************************************** '

      TOTAL = 0
      Totalt = 0
      TOT_ITER = 0
      TOT_AUG = 0

      GRANDES = 1000000000


C ********************************************** ***********
C
C INÍCIO DE PROBLEMA DE ENTRADA CODE
C
C
C O problema é especificado pelo variáveis a seguir:
C
CN: O número de nós
C NA: o número de arcos
C FONTE: o nó origem do problema de fluxo máximo
C Pia: o sink DO PROBLEMA MAX-FLOW
C STARTN (ARC): o nó INÍCIO DE ARC
C ENDN (ARC): o nó FIM DO ARCO
CU (ARC): a capacidade da ARC (IT for alterado posteriormente à
capacidade residual)
C
      PRINT *, 'READING problema de dados'

      ABERTO (13, FILE = "FOR013.DAT ', STATUS =' velho ')
      Rewind (13)

C LEIA número de nós e arcos

      LEIA (13, *) N, NA

      FAZER 5 I = 1, NA
        LEIA (13, *) STARTN (I), ENDN (I), MANEQUIM, U (I)
5 CONTINUAR


      ENDFILE (13)
      Rewind (13)


      SOURCE = 1
      SINK = N

C



C
C FIM DO PROBLEMA entrada de código
C ********************************************** ***********
C
```

```
C INITIALIZE FLUXO

      Fazer 30 ARC = 1, NA
        F (ARC) = 0
X IF (U (ARC) .EQ.0) THEN
X PRINT *, "ARC", ARC, "TEM CAPACIDADE DE ZERO '
X PAUSE
X PARAR
X END IF
30 CONTINUAR

C REINICIAR PREÇOS E EXCEDENTES

        FAZER 50 NODE = 1, N
          P (nó) = N
          EXCEDENTE (nó) = 0
50 CONTINUAR

        P (pia) = 0


      CHAMADA INIDAT

      TIMER = Long (362)


C ********** CHAMAR ******** algoritmo principal

      CHAMADA AUCT_MF

C ***********************************************


      TT = FLOAT (LONG (362) - TIMER) / 60

      PRINT *, "tempo para encontrar um CUT MIN = ', TOT_TIME
      PRINT *, '******************************************** ** '
      PRINT *, 'TIME TOTAL =', TT, 's'
X PRINT *, 'TIME para inicialização =', INIT_TIME
X PRINT *, "tempo para construir LISTAS DE PRESSÃO = ', PL_TIME
X PRINT *, 'TIME FOR EXTENSIONS =', EXT_TIME
X PRINT *, 'TEMPO PARA SEGUNDA MELHOR LOGIC =', SB_TIME
X PRINT *, 'TIME FOR SCANS Node =', IT_TIME
X PRINT *, 'TIME FOR CONTR / EXT LOGIC =', CEL_TIME
X PRINT *, 'TIME FOR aumentos =', AUG_TIME
X PRINT *, 'TIME PARA VERIFICAR SE A CUT MIN =', CH_TIME

C ***** COMPUTE MAX-FLOW *****

      MAX_FLOW = 0
      ARC = FIN (PIA)
70 IF (ARC.GT.0) THEN
        MAX_FLOW = MAX_FLOW + F (ARC)
        ARC = NXTIN (ARC)
       GOTO 70
      END IF
X IF (MAX_FLOW.NE.SURPLUS (pia)) THEN
X PRINT *, 'ERRO NO CÁLCULO DO MAX-FLOW'
X END IF

      PRINT *, "MAX-FLOW = ', MAX_FLOW
```

```
      PRINT *, '# DE FLUXO DE EMPURRA / ARC =', FLOAT (Naug) / FLOAT
(NA)
      PRINT *, '# uma subida de preços / NODE =', FLOAT (Niter) /
FLOAT (N)
X PRINT *, '# de extensões / NODE =', FLOAT (NUM_EXT) / FLOAT (N)
X PRINT *, '# DE SPEC. TENTATIVAS DE EXTENSÃO = ', EXT_ATT
X PRINT *, '# DE SPEC. SUCESSOS DE EXTENSÃO = ', EXT_SUCC
X PRINT *, '# de segundas maiores sucessos =', SB_SUCC
      PRINT *, '****************************************** ** '


C ***** Verifique se todos os excedentes são ZERO ****

        FAZER 80 NODE = 1, N
          IF ((NODE.NE.SOURCE) .E. (NODE.NE.SINK)) THEN
            IF (EXCEDENTE (nó) .NE.0) THEN
              PRINT *, «excedente de NODE ', NODE,' é diferente de zero
'
            END IF
          END IF
80 CONTINUAR

Hora de gravação C

      ABERTO (16, FILE = "AUCTION.OUT ', STATUS =' Novo ')
      Rewind (16)

      ESCREVA (16,1020) TOT_TIME
1020 FORMATO ('TIME FOR CUT MIN =', F11.4, 's')
      ESCREVA (16,1021) TT
1021 FORMATO ('TOTAL TIME =', F11.4, 's')
X WRITE (16,1022) CH_TIME
X1022 formato ('TEMPO PARA VERIFICAR SE A CUT MIN =', F9.3)
X WRITE (16,1023) EXT_TIME
X1023 formato ('TIME FOR EXTENSIONS =', F9.3)
X WRITE (16,1024) IT_TIME
X1024 formato ('TIME FOR SCANS Node =', F9.3)
X WRITE (16,1025) SB_TIME
X1025 formato ('TEMPO PARA SEGUNDA MELHOR LOGIC =', F9.3)
X WRITE (16,1026) CEL_TIME
X1026 formato ('TIME FOR LOGIC CONTR / EXT =', F9.3)
X WRITE (16,1027) AUG_TIME
X1027 formato ('TIME FOR aumentos =', F9.3)


      ENDFILE (16)
      Rewind (16)

      PRINT *, 'programa terminou; PRESS <CR> PARA SAIR '
      PAUSA
      FIM




C ********************************************** ***************
C
C algoritmo principal LEILÃO PARA MAX-FLOW.
C Esta versão usa uma amplitude PRIMEIRA INICIALIZAÇÃO
C e uma rotina que detecta A CUT saturada.
```

```
C
C ESTE CÓDIGO foi escrito por
C DIMITRI P. Bertsekas
C fev 1993
C
C ********************************************** ***************

      AUCT_MF SUBROUTINE
      PARÂMETROS (maxNodes = 20001, MAXARCS = 120000)
      NONE IMPLICIT
      INTEGER N, NA, L, grande, FACTOR, EPS, ARC
      INTEGER Naug, Niter, EXT_ATT, EXT_SUCC, SB_SUCC, NODE
      INTEGER I, J, K, LN, CAPOUT, Capin
      INTEGER início, fim
      INTEGER FLOW, INCR, PREÇO, LIMIT, F_LIMIT, ARC_COUNT, NUM_EXT
      INTEGER BSTLEVEL, NEW_LEVEL, LAST, o excesso, NEWINCR
      INTEGER ROOT, TERM, PTERM, EXTARC, PREVARC, PRD
      INTEGER SECLEVEL, SECARC, N_PURG, cur_level, CUR_NODE
      INTEGER PrevNode, NXTNODE, PR_TERM, PREVLEVEL, TWO_NA
      INTEGER ISUCC, IPRED, GAPDIST, INEXT, IFIRST, IDIST, LEVEL,
PLEVEL
      FONTE INTEIRO, pia
      INTEGER NLIST, NSCAN
      Real * 8 TEMP, TIMER1, CH_TIME, EXT_TIME, SB_TIME, AUG_TIME,
MF_TIME
      Real * 8 IT_TIME, INIT_TIME, CEL_TIME, PL_TIME, TOT_TIME


      INTEGER STARTN (MAXARCS), ENDN (MAXARCS)
      INTEGER L (MAXARCS), F (MAXARCS)
      INTEGER FIN (maxNodes), FOUT (maxNodes)
      INTEGER NXTIN (MAXARCS), NXTOU (MAXARCS), P (maxNodes)
      INTEGER FPUSHF (maxNodes), NXTPUSHF (MAXARCS)
      INTEGER FPUSHB (maxNodes), NXTPUSHB (MAXARCS)
      INTEGER PRDARC (maxNodes), EXTEND_ARC (maxNodes)
      INTEGER SB_ARC (maxNodes)
      INTEGER excedente (maxNodes)
      LISTA INTEIRO (maxNodes), SB_LEVEL (maxNodes)
      INTEGER primeiro (0: maxNodes), SUCC (0: maxNodes), PRED (0:
maxNodes)
      INTEGER PFIRST (0: maxNodes), PSUCC (0: maxNodes), PPRED (0:
maxNodes)


      COMUM / escalares / N, NA, GRANDE, SOURCE, pia
      COMUM / STATS / Niter
      COMUM / STATS1 / Naug, EXT_ATT, EXT_SUCC, SB_SUCC, NUM_EXT
      COMUM / CHECK / CH_TIME, EXT_TIME, SB_TIME, AUG_TIME, MF_TIME
      COMUM / check1 / IT_TIME, INIT_TIME, CEL_TIME, PL_TIME, TOT_TIME
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / UBOUND / U
      COMUM / FLUXO / F
      Common / PREÇOS / P
      COMUM / BLK3 / excedente
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      COMUM / PRED / PRDARC
```

```
ESTRUTURA DE DADOS C ******* que detecta uma lacuna na PREÇOS
*********

C FIRST (D): primeiro nó no Label D
C SUCC (nó): Sucessor de NÓ NA DISTÂNCIA SAME (= 0 se o nó é a última)
C PRED (nó): antecessor NÓ NA DISTÂNCIA SAME (= 0 se o nó é o
primeiro)

C ********************************************** *************

C PLEVEL: A etiqueta da mais alta rótulo do nó W / POS. EXCEDENTE

C ********************************************** *************


      TIMER1 = longa (362)

      TOT_TIME = 0

C REINICIAR CONTAGEM DO NÚMERO DE CIMA ITERATIONS REALIZADAS


X TIMER = longa (362)


      Niter = 0
      Naug = 0
X = 0 NUM_EXT

X = 0 CH_TIME
X = 0 EXT_TIME
X = 0 SB_TIME
X = 0 AUG_TIME
X = 0 MF_TIME
X = 0 IT_TIME
X = 0 INIT_TIME
X = 0 CEL_TIME
X = 0 PL_TIME


X = 0 EXT_ATT
X = 0 EXT_SUCC
X = 0 SB_SUCC

C definir parâmetros PATH limitação de tempo

      F_LIMIT = 1

C INITIALIZE FLUXO

X fazer 30 ARC = 1, NA
XF (ARC) = 0
X IF (U (ARC) .EQ.0) THEN
X PRINT *, "ARC", ARC, "TEM CAPACIDADE DE ZERO '
X PAUSE
X PARAR
X END IF
X30 CONTINUAR

C REINICIAR estruturas de dados
```

```
        FAZER 50 NODE = 1, N
CP (nó) = N
C excedente (nó) = 0
          FIRST (nó) = 0
          PFIRST (nó) = 0
          PPRED (nó) = 0
          PSUCC (nó) = 0
          SB_LEVEL (nó) = - GRANDE
50 CONTINUAR

C REINICIAR PREÇOS POR UM MÉTODO PESQUISA BOCA PRIMEIRA

        P (FONTE) = N
        P (pia) = 0
        LEVEL = 0

        NLIST = 1
        LIST (1) = SINK
        NSCAN = 0

60 CONTINUAR
        NSCAN = + 1 NSCAN
        NÓ = LIST (NSCAN)
        PREÇO = P (nó) +1
        ARC = FIN (nó)
65 IF (ARC.GT.0) THEN
          START = STARTN (ARC)
          IF (P (START) .EQ.N) THEN
            NLIST = NLIST + 1
            LIST (NLIST) = INÍCIO
            EXTEND_ARC (START) = 0
            P (START) = PREÇO
            IF (LEVEL.LT.PRICE) LEVEL = PREÇO
            IFIRST = PRIMEIRO (preço)
            FIRST (PRICE) = INÍCIO
            SUCC (START) = IFIRST
            PRED (START) = 0
            PRED (IFIRST) = INÍCIO
          END IF
          ARC = NXTIN (ARC)
          GOTO 65
        END IF


        IF (NLIST.GT.NSCAN) GOTO 60


        P (FONTE) = N
X = 1 N_PURG


C INITIALIZE PLEVEL, fluxos e EXCEDENTES

        PLEVEL = 0

        ARC = FOUT (SOURCE)
70 IF (ARC.GT.0) THEN
          END = ENDN (ARC)
          PREÇO = P (END)
          IF (PLEVEL.LT.PRICE) PLEVEL = PREÇO
          IF (excedente (END) .EQ.0) THEN
```

```
            IFIRST = PFIRST (preço)
            PFIRST (preço) = END
            PSUCC (END) = IFIRST
            PPRED (IFIRST) = END
          END IF
          INCR = U (ARC)
          F (ARC) = INCR
          U (ARC) = 0
          EXCEDENTE (END) = excedente (END) + INCR
          ARC = NXTOU (ARC)
          GOTO 70
        END IF


X = INIT_TIME INIT_TIME + FLOAT (LONG (362) - TIMER) / 60


C *** aumentos COMEÇO ***


200 CONTINUAR

      ROOT = PFIRST (PLEVEL)

X IF (excedente (ROOT) .EQ.0) THEN
X PRINT *, «excedente de raiz é ZERO '
X PAUSE
X PARAR
X END IF

       TERM = ROOT

       ARC_COUNT = 0

C ********************************************** ********
C
C *** algoritmo principal com a raiz como a origem ***
C
C ********************************************** ********

500 CONTINUAR


X TIMER = longa (362)

C INÍCIO DE UMA NOVA FRENTE ITERATION

      PTERM = P (TERM)
      EXTARC = EXTEND_ARC (TERM)

X IF (PTERM.GE.N) THEN
X PRINT *, '*** ATENÇÃO: NÓ W / PREÇO GRANDE é terminal'
X PAUSE
X PARAR
X END IF

      IF (EXTARC.EQ.0) THEN

X TIMER = longa (362)

C construir a lista de ARCS W / QUARTO PARA EMPURRAR FLUXO
```

```
        FPUSHF (TERM) = 0
        ARC = FOUT (TERM)
510 IF (ARC.GT.0) THEN
          IF (U (ARC) .GT.0) THEN
            IF (FPUSHF (TERM) .EQ.0) THEN
              FPUSHF (TERM) = ARC
              NXTPUSHF (ARC) = 0
              LAST = ARC
            MAIS
              NXTPUSHF (LAST) = ARC
              NXTPUSHF (ARC) = 0
              LAST = ARC
            END IF
          END IF
          ARC = NXTOU (ARC)
          IR PARA 510
        END IF

        FPUSHB (TERM) = 0
        ARC = FIN (TERM)
520 IF (ARC.GT.0) THEN
          IF (F (ARC) .GT.0) THEN
            IF (FPUSHB (TERM) .EQ.0) THEN
              FPUSHB (TERM) = ARC
              NXTPUSHB (ARC) = 0
              LAST = ARC
            MAIS
              NXTPUSHB (LAST) = ARC
              NXTPUSHB (ARC) = 0
              LAST = ARC
            END IF
          END IF
          ARC = NXTIN (ARC)
          IR PARA 520
        END IF

X = PL_TIME PL_TIME + FLOAT (LONG (362) - TIMER) / 60
        IR PARA 600
      END IF


C PATH ESPECULATIVO EXTENSÃO TENTATIVA
C NOTA: ARC> 0 significa que ARC é orientado, desde a nascente até a
pia
C ARC <0 significa que ARC é orientada da pia AO SOURCE

X = EXT_ATT EXT_ATT + 1

      IF (EXTARC.GT.0) THEN
        IF (U (EXTARC) .EQ.0) THEN
           SECLEVEL = SB_LEVEL (TERM)
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
           IR PARA 580
        END IF
        END = ENDN (EXTARC)
        BSTLEVEL = P (END)
        IF (PTERM.GE.BSTLEVEL) THEN
X = EXT_SUCC EXT_SUCC + 1
X = NUM_EXT NUM_EXT + 1
          TERM = END
```

```
          PRDARC (TERM) = EXTARC

C Se o caminho é longo, FAZER UM AUMENTO

          IF (ARC_COUNT.GE.F_LIMIT) THEN
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
            GOTO 2000
          END IF

C IF pia é encontrado, FAZER UM AUMENTO

          IF (TERM.EQ.SINK) THEN
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
            GOTO 2000
          END IF

C voltar para outra ITERATION

          ARC_COUNT = + 1 ARC_COUNT
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
          IR PARA 500
        END IF
      MAIS
        EXTARC = -EXTARC
        IF (F (EXTARC) .EQ.0) THEN
           SECLEVEL = SB_LEVEL (TERM)
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
           IR PARA 580
        END IF
        START = STARTN (EXTARC)
        BSTLEVEL = P (START)
        IF (PTERM.GE.BSTLEVEL) THEN
X = EXT_SUCC EXT_SUCC + 1
X = NUM_EXT NUM_EXT + 1
          TERM = INÍCIO
          PRDARC (TERM) = - EXTARC

C Se o caminho é longo, FAZER UM AUMENTO

          IF (ARC_COUNT.GE.F_LIMIT) THEN
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
            GOTO 2000
          END IF

C IF pia é encontrado, FAZER UM AUMENTO

          IF (TERM.EQ.SINK) THEN
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
            GOTO 2000
          END IF

C voltar para outra ITERATION

          ARC_COUNT = + 1 ARC_COUNT
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
          IR PARA 500
        END IF
      END IF

C segundo melhor TEST lógica aplicada para salvar um SCAN nó completo
```

```
C Se o velho melhor nível continua a ser melhor ir para outra
contração

550 CONTINUAR

X TIMER = longa (362)

         SECLEVEL = SB_LEVEL (TERM)
         IF (BSTLEVEL.LE.SECLEVEL) THEN
X = SB_SUCC SB_SUCC + 1
X = SB_TIME SB_TIME + FLOAT (LONG (362) - TIMER) / 60
           GOTO 800
         END IF

C IF segundo melhor PODE SER USADO, siga um uma contração
C ou começar de novo com uma extensão ESPECULATIVO

580 IF (SECLEVEL.GT.-GRANDE) THEN
          EXTARC = SB_ARC (TERM)
          IF (EXTARC.GT.0) THEN
             IF (U (EXTARC) .EQ.0) THEN
X = SB_TIME SB_TIME + FLOAT (LONG (362) - TIMER) / 60
               GOTO 600
             END IF
             BSTLEVEL = P (ENDN (EXTARC))
          MAIS
             IF (F (-EXTARC) .EQ.0) THEN
X = SB_TIME SB_TIME + FLOAT (LONG (362) - TIMER) / 60
               GOTO 600
             END IF
             BSTLEVEL = P (STARTN (-EXTARC))
          END IF
          IF (BSTLEVEL.EQ.SECLEVEL) THEN
X = SB_SUCC SB_SUCC + 1
            SB_LEVEL (TERM) = - GRANDE
            EXTEND_ARC (TERM) = EXTARC
X = SB_TIME SB_TIME + FLOAT (LONG (362) - TIMER) / 60
            GOTO 800
          END IF
       END IF

C
C TENTATIVA extensão foi vencida, SO SCAN nodo terminal
C

600 CONTINUAR

X TIMER = longa (362)


       BSTLEVEL = GRANDE
       SECLEVEL = GRANDE

       ARC = FPUSHF (TERM)
700 IF (ARC.GT.0) THEN
         NEW_LEVEL = P (ENDN (ARC))
         IF (NEW_LEVEL.LT.SECLEVEL) THEN
           IF (NEW_LEVEL.LT.BSTLEVEL) THEN
             SECLEVEL = BSTLEVEL
             BSTLEVEL = NEW_LEVEL
             SECARC = EXTARC
```

```
            EXTARC = ARC
           MAIS
            SECLEVEL = NEW_LEVEL
            SECARC = ARC
           END IF
         END IF
         ARC = NXTPUSHF (ARC)
         GOTO 700
       END IF

       ARC = FPUSHB (TERM)
710 IF (ARC.GT.0) THEN
         NEW_LEVEL = P (STARTN (ARC))
         IF (NEW_LEVEL.LT.SECLEVEL) THEN
           IF (NEW_LEVEL.LT.BSTLEVEL) THEN
              SECLEVEL = BSTLEVEL
              BSTLEVEL = NEW_LEVEL
              SECARC = EXTARC
              EXTARC = -ARC
            MAIS
              SECLEVEL = NEW_LEVEL
              SECARC = -ARC
           END IF
         END IF
         ARC = NXTPUSHB (ARC)
         GOTO 710
       END IF

       SB_LEVEL (TERM) = SECLEVEL
       SB_ARC (TERM) = SECARC
       EXTEND_ARC (TERM) = EXTARC

X = IT_TIME IT_TIME + FLOAT (LONG (362) - TIMER) / 60


C ************* FIM DO NÓ SCAN *************

C Se o nó terminal é o ROOT, ajuste o seu preço e CHANGE ROOT

800 CONTINUAR


           NEW_LEVEL = + 1 BSTLEVEL

           P (TERM) = NEW_LEVEL

C VERIFICAÇÃO DE MUDANÇA DE PREÇOS E A GAP

           IF (PTERM.LT.NEW_LEVEL) THEN

             Niter = Niter + 1

X TIMER = longa (362)

C IF termo tem POSITIVO SURPPLUS AJUSTE O POS. BUCKETS EXCEDENTES

             IF (excedente (TERM) .GT.0) THEN

               ISUCC = PSUCC (TERM)
               IPRED = PPRED (TERM)
               PSUCC (IPRED) = ISUCC
```

```
                PPRED (ISUCC) = IPRED
                 IF (IPRED.EQ.0) PFIRST (PTERM) = ISUCC

                IF (NEW_LEVEL.LT.N) THEN

C Ajuste nível e local TERMO NO TOPO DO NOVO PREÇO BALDE

                  IF (PLEVEL.LT.NEW_LEVEL) PLEVEL = NEW_LEVEL
                  NÓ = PFIRST (NEW_LEVEL)
                  PFIRST (NEW_LEVEL) = TERM
                  PPRED (nó) = TERM
                  PSUCC (TERM) = NÓ
                  PPRED (TERM) = 0
                MAIS
                  IF (PLEVEL.EQ.PTERM) THEN
                     IF ((IPRED.EQ.0) .E. (ISUCC.EQ.0)) THEN

C termo é purgado e é o único node no POS MAX. EXCEDENTE Nível de
Preços
C encontrar o novo PLEVEL

850 PLEVEL = PLEVEL-1
                      IF (PLEVEL.EQ.0) THEN
X = CH_TIME CH_TIME + FLOAT (LONG (362) - TIMER) / 60
                        TOT_TIME = TOT_TIME + FLOAT (LONG (362) -
TIMER1) / 60
                        GOTO 3000
                      END IF
                      IF (PFIRST (PLEVEL) .EQ.0) GOTO 850
                    END IF
                  END IF
                END IF
              END IF

C ***********************************************

C Retire TERMO do balde preço atual

              ISUCC = SUCC (TERM)
              IPRED = PRED (TERM)
              SUCC (IPRED) = ISUCC
              PRED (ISUCC) = IPRED

              IF (NEW_LEVEL.LT.N) THEN

C Ajuste nível e local TERMO NO TOPO DO NOVO PREÇO BALDE

                IF (LEVEL.LT.NEW_LEVEL) LEVEL = NEW_LEVEL
                NÓ = FIRST (NEW_LEVEL)
                FIRST (NEW_LEVEL) = TERM
                PRED (nó) = TERM
                SUCC (TERM) = NÓ
                PRED (TERM) = 0
              MAIS
X = N_PURG N_PURG + 1
X PRINT *, 'NÚMERO purgado =', N_PURG
              END IF

              IF (IPRED.EQ.0) THEN
                FIRST (PTERM) = ISUCC
                IF (ISUCC.EQ.0) THEN
```

```
C ************ GAP foi obtida; AJUSTE estruturas de dados

                 IF (F_LIMIT.LT.5) F_LIMIT = F_LIMIT + 1

C purgar a nós acima THE GAP fixando o preço TO N
C e limpar o BUCKETS ACIMA DA GAP

                 IF (LEVEL.GT.PTERM) THEN
                   FAZER 920 cur_level = LEVEL, PTERM + 1, -1
                    CUR_NODE = FIRST (cur_level)
900 IF (CUR_NODE.GT.0) THEN
X = N_PURG N_PURG + 1
                      P (CUR_NODE) = N
                      CUR_NODE = SUCC (CUR_NODE)
                      GOTO 900
                    END IF
                    FIRST (cur_level) = 0
                    PFIRST (cur_level) = 0
920 CONTINUAR

X PRINT *, 'GAP AT PREÇO ", PTERM,' # purgado = ', N_PURG
                 END IF

C encontrar o novo NÍVEL E NEW PLEVEL, e iniciar outro PATH

                 PLEVEL = PTERM

950 PLEVEL = PLEVEL-1
               IF (PLEVEL.EQ.0) THEN
X = CH_TIME CH_TIME + FLOAT (LONG (362) - TIMER) / 60
                  TOT_TIME = TOT_TIME + FLOAT (LONG (362) - TIMER1) /
60
                  GOTO 3000
               END IF
               IF (PFIRST (PLEVEL) .EQ.0) GOTO 950

                LEVEL = PTERM

980 LEVEL = LEVEL-1
               IF (LEVEL.EQ.0) THEN
X = CH_TIME CH_TIME + FLOAT (LONG (362) - TIMER) / 60
                 TOT_TIME = TOT_TIME + FLOAT (LONG (362) - TIMER1) /
60
                 GOTO 3000
               END IF
               IF (FIRST (LEVEL) .EQ.0) GOTO 980

X = CH_TIME CH_TIME + FLOAT (LONG (362) - TIMER) / 60
                GOTO 200
              END IF
            END IF
          END IF


C ********** INÍCIO DE CONTRAÇÃO / extensão LOGIC **********

X TIMER = longa (362)

C FAZER uma contração na raiz IF TERM = ROOT
```

```
            IF (TERM.EQ.ROOT) THEN
X = CEL_TIME CEL_TIME + FLOAT (LONG (362) - TIMER) / 60
              IR PARA 200
            END IF

C Verifique se a extensão ou contração

            PRD = PRDARC (TERM)

            IF (PRD.GT.0) THEN
               PR_TERM = STARTN (PRD)
            MAIS
               PR_TERM = ENDN (-PRD)
            END IF
            PREVLEVEL = P (PR_TERM)

            IF (PREVLEVEL.GT.BSTLEVEL) THEN

C EXTENSÃO PATH


X = NUM_EXT NUM_EXT + 1
              IF (EXTARC.GT.0) THEN
                END = ENDN (EXTARC)
                TERM = END
              MAIS
                START = STARTN (-EXTARC)
                TERM = INÍCIO
              END IF
              PRDARC (TERM) = EXTARC

C IF pia é encontrado, FAZER UM AUMENTO

              IF (TERM.EQ.SINK) THEN
X = CEL_TIME CEL_TIME + FLOAT (LONG (362) - TIMER) / 60
                GOTO 2000
              END IF

              ARC_COUNT = + 1 ARC_COUNT

C voltar para outra ITERATION

X = CEL_TIME CEL_TIME + FLOAT (LONG (362) - TIMER) / 60
              IR PARA 500
          MAIS

C CONTRAÇÃO PATH

              TERM = PR_TERM
              ARC_COUNT = ARC_COUNT-1
              PTERM = P (TERM)
              EXTARC = PRD
              BSTLEVEL = + 1 BSTLEVEL

C DO SEGUNDO A melhor teste e se isso falhar, fazer uma varredura nó
completo

X = CEL_TIME CEL_TIME + FLOAT (LONG (362) - TIMER) / 60
              GOTO 550
           END IF
```

```
C ************************************************
**************************

C DO AUMENTO DE RAIZ, LISTAS DE PRESSÃO CORRETO, e baldes com
resultados positivos

2000 CONTINUAR

X TIMER = longa (362)

C inicializar o AUGMENTATION

        INCR = 0
       NÓ = ROOT

2100 EXTARC = EXTEND_ARC (nó)
        Naug = Naug + 1
        IF (EXTARC.GT.0) THEN
          EXCESSO = excedente (nó)
          IF (EXCESS.GT.U (EXTARC)) THEN
            NEWINCR = L (EXTARC)
          MAIS
            NEWINCR = excesso
          END IF

C INSERT / REMOVER nó no BUCKETS com resultados positivos

          IF ((NEWINCR.LT.INCR) .E. (INCR.EQ.EXCESS)) THEN

C NODE INSERIR no balde de seu preço

             PREÇO = P (nó)
             IFIRST = PFIRST (preço)
             PFIRST (preço) = NÓ
             PPRED (nó) = 0
             PSUCC (nó) = IFIRST
             PPRED (IFIRST) = NÓ
             IF (PLEVEL.LT.PRICE) PLEVEL = PREÇO
           MAIS
              IF ((NEWINCR.EQ.EXCESS) .E. (INCR.LT.EXCESS)) THEN

C Retire nó a partir do POS CURRENT. BALDE EXCEDENTE

               ISUCC = PSUCC (nó)
               IPRED = PPRED (nó)
               PSUCC (IPRED) = ISUCC
               PPRED (ISUCC) = IPRED

               IF (IPRED.EQ.0) THEN
                 PFIRST (P (nó)) = ISUCC
                 IF (ISUCC.EQ.0) THEN

C lacuna no POS. BUCKETS EXCEDENTES foi obtida
C encontrar o novo POS. EXCEDENTE nível de preços e iniciar outro PATH

                   IF (PLEVEL.EQ.P (nó)) THEN
2170 PLEVEL = PLEVEL-1
                    IF (PFIRST (PLEVEL) .EQ.0) GOTO 2170
                  END IF
```

```
                END IF
              END IF
            END IF
          END IF
          INCR = NEWINCR
          END = ENDN (EXTARC)

C Adicionar ARC ao gráfico REDUZIDA

          IF (F (EXTARC) .EQ.0) THEN
            NXTPUSHB (EXTARC) = FPUSHB (END)
            FPUSHB (END) = EXTARC
            NEW_LEVEL = P (nó)
            IF (SB_LEVEL (END) .GT.NEW_LEVEL) THEN
              SB_LEVEL (END) = NEW_LEVEL
              SB_ARC (END) = - EXTARC
            END IF
          END IF

          F (EXTARC) = F (EXTARC) + INCR
          L (EXTARC) = L (EXTARC) -INCR

          EXCEDENTE (END) = excedente (END) + INCR
          EXCEDENTE (nó) = excedente (nó) -INCR

C Retire ARC a partir do gráfico REDUZIDA

          IF (U (EXTARC) .EQ.0) THEN
            ARC = FPUSHF (nó)
            IF (ARC.EQ.EXTARC) THEN
              FPUSHF (nó) = NXTPUSHF (ARC)
            MAIS
              PREVARC = ARC
              ARC = NXTPUSHF (ARC)
2200 IF (ARC.GT.0) THEN
                IF (ARC.EQ.EXTARC) THEN
                  NXTPUSHF (PREVARC) = NXTPUSHF (ARC)
                  IR PARA 2250
                END IF
                PREVARC = ARC
                ARC = NXTPUSHF (ARC)
                GOTO 2200
              END IF
            END IF
X IF (SB_LEVEL (nó) .gt (-. GRANDE)) THEN
X IF (EXTARC.EQ.SB_ARC (nó)) THEN
X SB_LEVEL (nó) = - GRANDE
X PRINT *, '** ATENÇÃO ** SECARC = EXTARC no aumento'
X END IF
X END IF
          END IF

2250 NODE = END

        MAIS
          EXTARC = -EXTARC
          EXCESSO = excedente (nó)
          IF (EXCESS.GT.F (EXTARC)) THEN
            NEWINCR = F (EXTARC)
          MAIS
            NEWINCR = excesso
```

```
          END IF

C INSERT / REMOVE nó na baldes com resultados positivos

          IF ((NEWINCR.LT.INCR) .E. (INCR.EQ.EXCESS)) THEN

C NODE INSERIR no balde de seu preço

             PREÇO = P (nó)
             IFIRST = PFIRST (preço)
             PFIRST (preço) = NÓ
             PPRED (nó) = 0
             PSUCC (nó) = IFIRST
             PPRED (IFIRST) = NÓ
             IF (PLEVEL.LT.PRICE) PLEVEL = PREÇO
           MAIS
              IF ((NEWINCR.EQ.EXCESS) .E. (INCR.LT.EXCESS)) THEN

C Retire nó a partir do POS CURRENT. BALDE EXCEDENTE

               ISUCC = PSUCC (nó)
               IPRED = PPRED (nó)
               PSUCC (IPRED) = ISUCC
               PPRED (ISUCC) = IPRED

               IF (IPRED.EQ.0) THEN
                 PFIRST (P (nó)) = ISUCC
                 IF (ISUCC.EQ.0) THEN

C lacuna no POS. BUCKETS EXCEDENTES foi obtida
C encontrar o novo POS. EXCEDENTE nível de preços e iniciar outro PATH

                   IF (PLEVEL.EQ.P (nó)) THEN
2270 PLEVEL = PLEVEL-1
                     IF (PFIRST (PLEVEL) .EQ.0) GOTO 2270
                   END IF
                 END IF
               END IF
             END IF
           END IF
           INCR = NEWINCR
           START = STARTN (EXTARC)

C Adicionar ARC ao gráfico REDUZIDA

           IF (U (EXTARC) .EQ.0) THEN
             NXTPUSHF (EXTARC) = FPUSHF (START)
             FPUSHF (START) = EXTARC
             NEW_LEVEL = P (nó)
             IF (SB_LEVEL (START) .GT.NEW_LEVEL) THEN
               SB_LEVEL (START) = NEW_LEVEL
               SB_ARC (START) = EXTARC
             END IF
           END IF

           L (EXTARC) = L (EXTARC) + œINCR
           F (EXTARC) = F (EXTARC) -INCR

           EXCEDENTE (START) = excedente (START) + INCR
           EXCEDENTE (nó) = excedente (nó) -INCR
```

```
C Retire ARC a partir do gráfico REDUZIDA

          IF (F (EXTARC) .EQ.0) THEN
            ARC = FPUSHB (nó)
            IF (ARC.EQ.EXTARC) THEN
              FPUSHB (nó) = NXTPUSHB (ARC)
            MAIS
              PREVARC = ARC
              ARC = NXTPUSHB (ARC)
2300 IF (ARC.GT.0) THEN
               IF (ARC.EQ.EXTARC) THEN
                 NXTPUSHB (PREVARC) = NXTPUSHB (ARC)
                 IR PARA 2350
               END IF
               PREVARC = ARC
               ARC = NXTPUSHB (ARC)
               GOTO 2300
             END IF
           END IF
X IF (SB_LEVEL (nó) .gt (-. GRANDE)) THEN
X IF ((-EXTARC) .EQ.SB_ARC (nó)) THEN
X SB_LEVEL (nó) = - GRANDE
X PRINT *, '** ATENÇÃO ** SECARC = EXTARC no aumento'
X END IF
X END IF
          END IF

2350 NODE = INÍCIO

        END IF

        IF (NODE.NE.TERM) GOTO 2100

X IF (INCR.LE.0) THEN
X PRINT *, "aumento inválido '
X PAUSE
X PARAR
X END IF


C Verifique se TERMINAL NODE MUDOU DE ZERO AO POS. EXCEDENTE

        IF (EXCEDENTE (nó) .EQ.INCR) THEN

C NODE INSERIR no balde de seu preço

          PREÇO = P (nó)
          IFIRST = PFIRST (preço)
          PFIRST (preço) = NÓ
          PPRED (nó) = 0
          PSUCC (nó) = IFIRST
          PPRED (IFIRST) = NÓ
          IF (PLEVEL.LT.PRICE) PLEVEL = PREÇO
        END IF

X = AUG_TIME AUG_TIME + FLOAT (LONG (362) - TIMER) / 60

C *************** FIM DE CICLO DE AUMENTO **************

C VERIFICAÇÃO DE ENCERRAMENTO
```

```
        IF (PLEVEL.EQ.0) THEN
          TOT_TIME = TOT_TIME + FLOAT (LONG (362) - TIMER1) / 60
          GOTO 3000
        END IF

C voltar a começar um outro caminho


        GOTO 200

C ********************************************** ***********
C
C ********* INÍCIO DE MAX-FLOW RECTIFICATION ***********
C ******** fluxo de retorno excesso à SOURCE ***********
C
C ********************************************* *** ***********


3000 CONTINUAR

X PRINT *, «MÍNIMO CUT encontrado '

        F_LIMIT = 2

C REINICIAR estruturas de dados

        FAZER 3010 NODE = 1, N
          P (nó) = N
          PFIRST (nó) = 0
3010 CONTINUAR

C REINICIAR PREÇOS POR UMA BOCA DE PRIMEIRA PESQUISA SINK

        NLIST = 1
        P (pia) = 0
        LIST (1) = SINK
        NSCAN = 0

3020 CONTINUAR
        NSCAN = + 1 NSCAN
        NÓ = LIST (NSCAN)

        ARC = FIN (nó)
3030 IF (ARC.GT.0) THEN
          IF (U (ARC) .GT.0) THEN
            START = STARTN (ARC)
            IF (P (START) .EQ.N) THEN
              NLIST = NLIST + 1
              LIST (NLIST) = INÍCIO
              P (START) = 0
            END IF
          END IF
          ARC = NXTIN (ARC)
          GOTO 3030
        END IF

        ARC = FOUT (nó)
3040 IF (ARC.GT.0) THEN
         IF (F (ARC) .GT.0) THEN
          END = ENDN (ARC)
          IF (P (END) .EQ.N) THEN
```

```
            NLIST = NLIST + 1
            LIST (NLIST) = END
            P (END) = 0
          END IF
         END IF
         ARC = NXTOU (ARC)
         GOTO 3040
        END IF

        IF (NLIST.GT.NSCAN) GOTO 3020

X PRINT *, '# de nós em SINK lado do corte =', NLIST

C REINICIAR preços em uma amplitude primeira pesquisa a partir da
fonte

        P (FONTE) = 0
        PLEVEL = 0

        NLIST = 1
        LIST (1) = SOURCE
        NSCAN = 0

3060 CONTINUAR
        NSCAN = + 1 NSCAN
        NÓ = LIST (NSCAN)
        SB_LEVEL (nó) = - GRANDE
        PREÇO = P (nó) +1

        ARC = FIN (nó)
3065 IF (ARC.GT.0) THEN
         IF (U (ARC) .GT.0) THEN
          START = STARTN (ARC)
          IF (P (START) .EQ.N) THEN
            NLIST = NLIST + 1
            LIST (NLIST) = INÍCIO
            IF (EXTEND_ARC (START) .NE.0) EXTEND_ARC (START) = ARC
            P (START) = PREÇO
            IF (EXCEDENTE (START) .GT.0) THEN
               IF (PLEVEL.LT.PRICE) PLEVEL = PREÇO
               IFIRST = PFIRST (preço)
               PFIRST (preço) = INÍCIO
               PSUCC (START) = IFIRST
               PPRED (START) = 0
               PPRED (IFIRST) = INÍCIO
            MAIS
               PSUCC (START) = 0
               PPRED (START) = 0
            END IF
          END IF
         END IF
         ARC = NXTIN (ARC)
         GOTO 3065
        END IF

        ARC = FOUT (nó)
3070 IF (ARC.GT.0) THEN
         IF (F (ARC) .GT.0) THEN
          END = ENDN (ARC)
          IF (P (END) .EQ.N) THEN
            NLIST = NLIST + 1
```

```
            LIST (NLIST) = END
            IF (EXTEND_ARC (END) .NE.0) EXTEND_ARC (END) = - ARC
            P (END) = PREÇO
            IF (excedente (END) .GT.0) THEN
               IF (PLEVEL.LT.PRICE) PLEVEL = PREÇO
               IFIRST = PFIRST (preço)
               PFIRST (preço) = END
               PSUCC (END) = IFIRST
               PPRED (END) = 0
               PPRED (IFIRST) = END
            MAIS
               PSUCC (END) = 0
               PPRED (END) = 0
            END IF
          END IF
         END IF
         ARC = NXTOU (ARC)
         GOTO 3070
        END IF


        IF (NLIST.GT.NSCAN) GOTO 3060


C VERIFICAÇÃO DE ENCERRAMENTO

        IF (PLEVEL.EQ.0) RETURN

X = INIT_TIME INIT_TIME + FLOAT (LONG (362) - TIMER) / 60


C *** aumentos COMEÇO ***


3200 CONTINUAR

       ROOT = PFIRST (PLEVEL)

X IF (excedente (ROOT) .EQ.0) THEN
X PRINT *, «excedente de raiz é ZERO '
X PAUSE
X PARAR
X END IF

       TERM = ROOT

       ARC_COUNT = 0

C ********************************************** ********
C
C *** algoritmo principal com a raiz como a origem ***
C
C ********************************************** ********

3500 CONTINUAR


X TIMER = longa (362)

C INÍCIO DE UMA NOVA FRENTE ITERATION
```

```
      PTERM = P (TERM)
      EXTARC = EXTEND_ARC (TERM)

X IF (PTERM.GE.N) THEN
X PRINT *, '*** ATENÇÃO: NÓ W / PREÇO GRANDE é terminal'
X PAUSE
X PARAR
X END IF

      IF (EXTARC.EQ.0) THEN

X TIMER = longa (362)

C construir a lista de ARCS W / QUARTO PARA EMPURRAR FLUXO

        FPUSHF (TERM) = 0
        ARC = FOUT (TERM)
3510 IF (ARC.GT.0) THEN
          IF (U (ARC) .GT.0) THEN
            IF (FPUSHF (TERM) .EQ.0) THEN
              FPUSHF (TERM) = ARC
              NXTPUSHF (ARC) = 0
              LAST = ARC
            MAIS
              NXTPUSHF (LAST) = ARC
              NXTPUSHF (ARC) = 0
              LAST = ARC
            END IF
          END IF
          ARC = NXTOU (ARC)
          IR PARA 3510
        END IF

        FPUSHB (TERM) = 0
        ARC = FIN (TERM)
3520 IF (ARC.GT.0) THEN
          IF (F (ARC) .GT.0) THEN
            IF (FPUSHB (TERM) .EQ.0) THEN
              FPUSHB (TERM) = ARC
              NXTPUSHB (ARC) = 0
              LAST = ARC
            MAIS
              NXTPUSHB (LAST) = ARC
              NXTPUSHB (ARC) = 0
              LAST = ARC
            END IF
          END IF
          ARC = NXTIN (ARC)
          IR PARA 3520
        END IF

X = PL_TIME PL_TIME + FLOAT (LONG (362) - TIMER) / 60
        IR PARA 3600
      END IF

C PATH ESPECULATIVO EXTENSÃO TENTATIVA

X = EXT_ATT EXT_ATT + 1

      IF (EXTARC.GT.0) THEN
        IF (U (EXTARC) .EQ.0) THEN
```

```
           SECLEVEL = SB_LEVEL (TERM)
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
            IR PARA 3580
         END IF
         END = ENDN (EXTARC)
         BSTLEVEL = P (END)
         IF (PTERM.GE.BSTLEVEL) THEN
X = EXT_SUCC EXT_SUCC + 1
X = NUM_EXT NUM_EXT + 1
           TERM = END
           PRDARC (TERM) = EXTARC

C Se o caminho é longo, FAZER UM AUMENTO

           IF (ARC_COUNT.GE.F_LIMIT) THEN
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
             GOTO 5000
           END IF

C Se a fonte for encontrada, FAZER UM AUMENTO

           IF (TERM.EQ.SOURCE) THEN
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
             GOTO 5000
           END IF

C voltar para outra ITERATION

           ARC_COUNT = + 1 ARC_COUNT
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
            IR PARA 3500
         END IF
       MAIS
         EXTARC = -EXTARC
         IF (F (EXTARC) .EQ.0) THEN
            SECLEVEL = SB_LEVEL (TERM)
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
            IR PARA 3580
         END IF
         START = STARTN (EXTARC)
         BSTLEVEL = P (START)
         IF (PTERM.GE.BSTLEVEL) THEN
X = EXT_SUCC EXT_SUCC + 1
X = NUM_EXT NUM_EXT + 1
           TERM = INÍCIO
           PRDARC (TERM) = - EXTARC

C Se o caminho é longo, FAZER UM AUMENTO

           IF (ARC_COUNT.GE.F_LIMIT) THEN
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
             GOTO 5000
           END IF

C Se a fonte for encontrada, FAZER UM AUMENTO

           IF (TERM.EQ.SOURCE) THEN
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
             GOTO 5000
           END IF
```

```
C voltar para outra ITERATION

          ARC_COUNT = + 1 ARC_COUNT
X = EXT_TIME EXT_TIME + FLOAT (LONG (362) - TIMER) / 60
           IR PARA 3500
         END IF
       END IF


C segundo melhor TEST lógica aplicada para salvar um SCAN nó completo
C Se o velho melhor nível continua a ser melhor ir para outra
contração

3550 CONTINUAR

X TIMER = longa (362)

          SECLEVEL = SB_LEVEL (TERM)
          IF (BSTLEVEL.LE.SECLEVEL) THEN
X = SB_SUCC SB_SUCC + 1
X = SB_TIME SB_TIME + FLOAT (LONG (362) - TIMER) / 60
            GOTO 3800
          END IF

C IF segundo melhor PODE SER USADO, siga um uma contração
C ou começar de novo com uma extensão ESPECULATIVO

3580 IF (SECLEVEL.GT.-GRANDE) THEN
           EXTARC = SB_ARC (TERM)
           IF (EXTARC.GT.0) THEN
              IF (U (EXTARC) .EQ.0) THEN
X = SB_TIME SB_TIME + FLOAT (LONG (362) - TIMER) / 60
                GOTO 3600
              END IF
              BSTLEVEL = P (ENDN (EXTARC))
           MAIS
              IF (F (-EXTARC) .EQ.0) THEN
X = SB_TIME SB_TIME + FLOAT (LONG (362) - TIMER) / 60
                GOTO 600
              END IF
              BSTLEVEL = P (STARTN (-EXTARC))
           END IF
           IF (BSTLEVEL.EQ.SECLEVEL) THEN
X = SB_SUCC SB_SUCC + 1
             SB_LEVEL (TERM) = - GRANDE
             EXTEND_ARC (TERM) = EXTARC
X = SB_TIME SB_TIME + FLOAT (LONG (362) - TIMER) / 60
             GOTO 3800
           END IF
        END IF

C
C TENTATIVA extensão foi vencida, SO SCAN nodo terminal
C

3600 CONTINUAR

X TIMER = longa (362)


        BSTLEVEL = GRANDE
```

```
       SECLEVEL = GRANDE


        ARC = FPUSHF (TERM)
3700 IF (ARC.GT.0) THEN
         NEW_LEVEL = P (ENDN (ARC))
         IF (NEW_LEVEL.LT.SECLEVEL) THEN
           IF (NEW_LEVEL.LT.BSTLEVEL) THEN
             SECLEVEL = BSTLEVEL
             BSTLEVEL = NEW_LEVEL
             SECARC = EXTARC
             EXTARC = ARC
           MAIS
             SECLEVEL = NEW_LEVEL
             SECARC = ARC
           END IF
         END IF
         ARC = NXTPUSHF (ARC)
         GOTO 3700
       END IF

       ARC = FPUSHB (TERM)
3710 IF (ARC.GT.0) THEN
        NEW_LEVEL = P (STARTN (ARC))
        IF (NEW_LEVEL.LT.SECLEVEL) THEN
          IF (NEW_LEVEL.LT.BSTLEVEL) THEN
              SECLEVEL = BSTLEVEL
              BSTLEVEL = NEW_LEVEL
              SECARC = EXTARC
              EXTARC = -ARC
            MAIS
              SECLEVEL = NEW_LEVEL
              SECARC = -ARC
          END IF
        END IF
        ARC = NXTPUSHB (ARC)
        GOTO 3710
       END IF

       SB_LEVEL (TERM) = SECLEVEL
       SB_ARC (TERM) = SECARC
       EXTEND_ARC (TERM) = EXTARC

X = IT_TIME IT_TIME + FLOAT (LONG (362) - TIMER) / 60


C ************* FIM DO NÓ SCAN *************

C Se o nó terminal é o ROOT, ajuste o seu preço e CHANGE ROOT

3800 CONTINUAR


           NEW_LEVEL = + 1 BSTLEVEL

           P (TERM) = NEW_LEVEL

C VERIFICAÇÃO DE PREÇO CHANGE

           IF (PTERM.LT.NEW_LEVEL) THEN
```

```
              Niter = Niter + 1

X TIMER = longa (362)

C IF termo tem POSITIVO SURPPLUS AJUSTE O POS. BUCKETS EXCEDENTES

             IF (excedente (TERM) .GT.0) THEN

               ISUCC = PSUCC (TERM)
               IPRED = PPRED (TERM)
               PSUCC (IPRED) = ISUCC
               PPRED (ISUCC) = IPRED
                IF (IPRED.EQ.0) PFIRST (PTERM) = ISUCC

               IF (PLEVEL.LT.NEW_LEVEL) PLEVEL = NEW_LEVEL

               NÓ = PFIRST (NEW_LEVEL)
               PFIRST (NEW_LEVEL) = TERM
               PPRED (nó) = TERM
               PSUCC (TERM) = NÓ
               PPRED (TERM) = 0
             END IF
           END IF

C ********** INÍCIO DE CONTRAÇÃO / extensão LOGIC **********

X TIMER = longa (362)

C FAZER uma contração na raiz IF TERM = ROOT


            IF (TERM.EQ.ROOT) THEN
X = CEL_TIME CEL_TIME + FLOAT (LONG (362) - TIMER) / 60
             IR PARA 3200
           END IF

C Verifique se a extensão ou contração

           PRD = PRDARC (TERM)

           IF (PRD.GT.0) THEN
              PR_TERM = STARTN (PRD)
           MAIS
              PR_TERM = ENDN (-PRD)
           END IF
           PREVLEVEL = P (PR_TERM)

           IF (PREVLEVEL.GT.BSTLEVEL) THEN

C EXTENSÃO PATH


X = NUM_EXT NUM_EXT + 1
             IF (EXTARC.GT.0) THEN
               END = ENDN (EXTARC)
               TERM = END
             MAIS
               START = STARTN (-EXTARC)
               TERM = INÍCIO
             END IF
             PRDARC (TERM) = EXTARC
```

```
C Se a fonte for encontrada, FAZER UM AUMENTO

              IF (TERM.EQ.SOURCE) THEN
X = CEL_TIME CEL_TIME + FLOAT (LONG (362) - TIMER) / 60
                GOTO 5000
              END IF

              ARC_COUNT = + 1 ARC_COUNT

C voltar para outra ITERATION

X = CEL_TIME CEL_TIME + FLOAT (LONG (362) - TIMER) / 60
              IR PARA 3500
           MAIS

C CONTRAÇÃO PATH

              TERM = PR_TERM
              ARC_COUNT = ARC_COUNT-1
              PTERM = P (TERM)
              EXTARC = PRD
              BSTLEVEL = + 1 BSTLEVEL

C DO SEGUNDO A melhor teste e se isso falhar, fazer uma varredura nó
completo

X = CEL_TIME CEL_TIME + FLOAT (LONG (362) - TIMER) / 60
              GOTO 3550

            END IF


C ********************************************** **********
C DO AUMENTO DE RAIZ, CORRETO BUCKETS com resultados positivos

5000 CONTINUAR

X TIMER = longa (362)

C inicializar o AUGMENTATION

         INCR = 0
        NÓ = ROOT

5100 EXTARC = EXTEND_ARC (nó)
        Naug = Naug + 1
        IF (EXTARC.GT.0) THEN
          EXCESSO = excedente (nó)
          IF (EXCESS.GT.U (EXTARC)) THEN
            NEWINCR = L (EXTARC)
          MAIS
            NEWINCR = excesso
          END IF

C INSERT / REMOVER nó no BUCKETS com resultados positivos

          IF ((NEWINCR.LT.INCR) .E. (INCR.EQ.EXCESS)) THEN

C NODE INSERIR no balde de seu preço
```

```
              PREÇO = P (nó)
              IFIRST = PFIRST (preço)
              PFIRST (preço) = NÓ
              PPRED (nó) = 0
              PSUCC (nó) = IFIRST
              PPRED (IFIRST) = NÓ
              IF (PLEVEL.LT.PRICE) PLEVEL = PREÇO
            MAIS
              IF ((NEWINCR.EQ.EXCESS) .E. (INCR.LT.EXCESS)) THEN

C Retire nó a partir do POS CURRENT. BALDE EXCEDENTE

                ISUCC = PSUCC (nó)
                IPRED = PPRED (nó)
                PSUCC (IPRED) = ISUCC
                PPRED (ISUCC) = IPRED

                IF (IPRED.EQ.0) THEN
                  PFIRST (P (nó)) = ISUCC
                  IF (ISUCC.EQ.0) THEN

C lacuna no POS. BUCKETS EXCEDENTES foi obtida
C encontrar o novo POS. EXCEDENTE Nível de Preços

                    IF (PLEVEL.EQ.P (nó)) THEN
5170 PLEVEL = PLEVEL-1
                      IF (PFIRST (PLEVEL) .EQ.0) GOTO 5170
                    END IF
                  END IF
                END IF
              END IF
            END IF
            INCR = NEWINCR
            END = ENDN (EXTARC)

C Adicionar ARC ao gráfico REDUZIDA

            IF (F (EXTARC) .EQ.0) THEN
              NXTPUSHB (EXTARC) = FPUSHB (END)
              FPUSHB (END) = EXTARC
              NEW_LEVEL = P (nó)
              IF (SB_LEVEL (END) .GT.NEW_LEVEL) THEN
                SB_LEVEL (END) = NEW_LEVEL
                SB_ARC (END) = - EXTARC
              END IF
            END IF


            F (EXTARC) = F (EXTARC) + INCR
            L (EXTARC) = L (EXTARC) -INCR

            EXCEDENTE (END) = excedente (END) + INCR
            EXCEDENTE (nó) = excedente (nó) -INCR

C Retire ARC a partir do gráfico REDUZIDA

            IF (U (EXTARC) .EQ.0) THEN
              ARC = FPUSHF (nó)
              IF (ARC.EQ.EXTARC) THEN
                FPUSHF (nó) = NXTPUSHF (ARC)
              MAIS
```

```
              PREVARC = ARC
              ARC = NXTPUSHF (ARC)
5200 IF (ARC.GT.0) THEN
               IF (ARC.EQ.EXTARC) THEN
                 NXTPUSHF (PREVARC) = NXTPUSHF (ARC)
                 IR PARA 5250
               END IF
               PREVARC = ARC
               ARC = NXTPUSHF (ARC)
               GOTO 5200
             END IF
           END IF
X IF (SB_LEVEL (nó) .gt (-. GRANDE)) THEN
X IF (EXTARC.EQ.SB_ARC (nó)) THEN
X SB_LEVEL (nó) = - GRANDE
X PRINT *, '** ATENÇÃO ** SECARC = EXTARC no aumento'
X END IF
X END IF
         END IF

5250 NODE = END


        MAIS
          EXTARC = -EXTARC
          EXCESSO = excedente (nó)
          IF (EXCESS.GT.F (EXTARC)) THEN
            NEWINCR = F (EXTARC)
          MAIS
            NEWINCR = excesso
          END IF

C INSERT / REMOVE nó na baldes com resultados positivos

          IF ((NEWINCR.LT.INCR) .E. (INCR.EQ.EXCESS)) THEN

C NODE INSERIR no balde de seu preço

             PREÇO = P (nó)
             IFIRST = PFIRST (preço)
             PFIRST (preço) = NÓ
             PPRED (nó) = 0
             PSUCC (nó) = IFIRST
             PPRED (IFIRST) = NÓ
             IF (PLEVEL.LT.PRICE) PLEVEL = PREÇO
           MAIS
              IF ((NEWINCR.EQ.EXCESS) .E. (INCR.LT.EXCESS)) THEN

C Retire nó a partir do POS CURRENT. BALDE EXCEDENTE

               ISUCC = PSUCC (nó)
               IPRED = PPRED (nó)
               PSUCC (IPRED) = ISUCC
               PPRED (ISUCC) = IPRED

               IF (IPRED.EQ.0) THEN
                 PFIRST (P (nó)) = ISUCC
                 IF (ISUCC.EQ.0) THEN

C lacuna no POS. BUCKETS EXCEDENTES foi obtida
C encontrar o novo POS. EXCEDENTE Nível de Preços
```

```
                    IF (PLEVEL.EQ.P (nó)) THEN
5270 PLEVEL = PLEVEL-1
                      IF (PFIRST (PLEVEL) .EQ.0) GOTO 5270
                    END IF
                  END IF
                END IF
              END IF
            END IF
            INCR = NEWINCR
            START = STARTN (EXTARC)

C Adicionar ARC ao gráfico REDUZIDA

            IF (U (EXTARC) .EQ.0) THEN
              NXTPUSHF (EXTARC) = FPUSHF (START)
              FPUSHF (START) = EXTARC
              NEW_LEVEL = P (nó)
              IF (SB_LEVEL (START) .GT.NEW_LEVEL) THEN
                SB_LEVEL (START) = NEW_LEVEL
                SB_ARC (START) = EXTARC
              END IF
            END IF


            L (EXTARC) = L (EXTARC) + œINCR
            F (EXTARC) = F (EXTARC) -INCR

            EXCEDENTE (START) = excedente (START) + INCR
            EXCEDENTE (nó) = excedente (nó) -INCR

C Retire ARC a partir do gráfico REDUZIDA

            IF (F (EXTARC) .EQ.0) THEN
              ARC = FPUSHB (nó)
              IF (ARC.EQ.EXTARC) THEN
                FPUSHB (nó) = NXTPUSHB (ARC)
              MAIS
                PREVARC = ARC
                ARC = NXTPUSHB (ARC)
5300 IF (ARC.GT.0) THEN
                  IF (ARC.EQ.EXTARC) THEN
                    NXTPUSHB (PREVARC) = NXTPUSHB (ARC)
                    IR PARA 5350
                  END IF
                  PREVARC = ARC
                  ARC = NXTPUSHB (ARC)
                  GOTO 5300
                END IF
              END IF
X IF (SB_LEVEL (nó) .gt (-. GRANDE)) THEN
X IF ((-EXTARC) .EQ.SB_ARC (nó)) THEN
X SB_LEVEL (nó) = - GRANDE
X PRINT *, '** ATENÇÃO ** SECARC = EXTARC no aumento'
X END IF
X END IF
            END IF

5350 NODE = INÍCIO

          END IF
```

```
        IF (NODE.NE.TERM) GOTO 5100

X IF (INCR.LE.0) THEN
X PRINT *, "aumento inválido '
X PAUSE
X PARAR
X END IF


C Verifique se TERMINAL NODE MUDOU DE ZERO AO POS. EXCEDENTE

        IF (EXCEDENTE (nó) .EQ.INCR) THEN

C NODE INSERIR no balde de seu preço

          PREÇO = P (nó)
          IFIRST = PFIRST (preço)
          PFIRST (preço) = NÓ
          PPRED (nó) = 0
          PSUCC (nó) = IFIRST
          PPRED (IFIRST) = NÓ
          IF (PLEVEL.LT.PRICE) PLEVEL = PREÇO
        END IF

X = AUG_TIME AUG_TIME + FLOAT (LONG (362) - TIMER) / 60

C *************** FIM DE CICLO DE AUMENTO **************

C VERIFICAÇÃO DE ENCERRAMENTO

        IF (PLEVEL.EQ.0) RETURN

C voltar a começar um outro caminho


        GOTO 3200

      FIM




C ********************************************** ***
      SUBROUTINE INIDAT
C Esta sub-rotina USAM O STARTN matrizes de dados E ENDN
C PARA CONSTRUIR AUXILIAR DE DADOS ARRAYS FOUT, NXTOU, FIN, E
C NXTIN que são exigidos por AUCT_MF. Nisto SUBROUTINE
C arbitrariamente ENCOMENDAR AS ARCS DEIXANDO cada nó e STORE
C esta informação em FOUT E NXTOU. Da mesma forma, arbitragem
C Rily ORDER os arcos que entram em cada nó e armazenar esta
C INFORMAÇÃO EM FIN E NXTIN. NA CONCLUSÃO DO
C CONSTRUÇÃO, TEMOS
C
C FOUT (I) = Primeiro ARC DEIXANDO NODE I.
C NXTOU (J) = PRÓXIMO ARC saem do nó CHEFE DE ARC J.
C FIN (I) = Primeiro ARC Entrando no passo I.
C NXTIN (J) = PRÓXIMO ARC ENTRANDO NO NÓ DE CAUDA ARC J.

      PARÂMETROS (maxNodes = 20001, MAXARCS = 120000)
```

```
      Implícita INTEGER (AZ)
      COMUM / escalares / N, NA, GRANDE, SOURCE, pia
      COMUM / BLK1 / STARTN
      COMUM / BLK2 / ENDN
      COMUM / BLK4 / FIN
      COMUM / BLK5 / FOUT
      COMUM / BLK6 / NXTIN
      COMUM / BLK7 / NXTOU
      INTEGER STARTN (MAXARCS)
      INTEGER ENDN (MAXARCS)
      INTEGER FIN (maxNodes), FOUT (maxNodes)
      INTEGER NXTIN (MAXARCS), NXTOU (MAXARCS)
      INTEGER FINALIN (maxNodes), FINALOU (maxNodes)
C
      FAZER 20 NODE = 1, N
        FIN (nó) = 0
        FOUT (nó) = 0
        FINALIN (nó) = 0
        FINALOU (nó) = 0
20 CONTINUAR
C
      Fazer 30 ARC = 1, NA
        START = STARTN (ARC)
        END = ENDN (ARC)
        IF (FOUT (START) .NE.0) THEN
          NXTOU (FINALOU (START)) = ARC
        MAIS
          FOUT (START) = ARC
        END IF
        IF (FIN (END) .NE.0) THEN
          NXTIN (FINALIN (END)) = ARC
        MAIS
          FIN (END) = ARC
        END IF
        FINALOU (START) = ARC
        FINALIN (END) = ARC
        NXTIN (ARC) = 0
        NXTOU (ARC) = 0
30 CONTINUAR
C
      RETURN
      FIM
```