```
C
C AMOSTRA programa de chamada PARA RELAXAR-IV
C
C ------------------------------------------------ -------------
C
C Objetivo - Este programa lê NA DATA DE CUSTO LINEAR
C PROBLEMA ORDINÁRIA DO FLUXO rede pelo arquivo `RELAX4.INP ',
C chama o INIDAT ROTINA PARA CONSTRUIR lista encadeada pelo problema,
C e chama o RELAX4 rotina para resolver o problema.
C
C ------------------------------------------------ -------------
C
      Programa principal
      Implícita INTEGER (AZ)
C
C MAXNN = dimensão do ARRAYS NÓ de comprimento
C MAXNA = dimensão do ARRAYS ARC-length
C
      PARÂMETROS (MAXNN = 10000, MAXNA = 70000)
C
C parâmetros de entrada
C
CN = número de nós
C NA = número de arcos
C = A GRANDE INTEIRO MUITO GRANDE PARA REPRESENTAR INFINITY
C STARTN (J) = COMEÇAR nó para ARC J, J = 1, ..., NA
C ENDN (J) = TÉRMINO nó para ARC J, J = 1, ..., NA
CC (J) = CUSTO DE ARC J, J = 1, ..., NA
C
      INTEGER STARTN (MAXNA), ENDN (MAXNA), C (MAXNA)
      COMUM / INPUT / N, NA, GRANDE
      COMUNS / matrizes / STARTN / ARRAYE / ENDN / ARRAYC / C
C
PARÂMETROS C ATUALIZADO
C
CU (J) = CAPACIDADE DE ARC J na entrada E CAPACIDADE RESIDUAL
C Na saída, J = 1, ..., NA
CB (I) = demanda no nó i na entrada E zero na saída,
CI = 1, ..., N
C
      INTEGER U (MAXNA), B (MAXNN)
      COMUM / ARRAYU / U / arrayb / B
C
C parâmetros de saída
C
CX (J) = Flow no ARC J, J = 1, ..., NA
C RC (J) = custo reduzido de ARC J, J = 1, ..., NA
C NMULTINODE = número de iterações Multinode relaxamento em RELAX4
C ITER = NÚMERO DE RELAXAMENTO iterações na RELAX4
C NUM_AUGM = número de passos FLUXO EM aumento RELAX4
C NUM_ASCNT = NÚMERO DE PASSOS Multinode subida no RELAX4
C NSP = número de leilão / mais curto ITERATIONS PATH
C TCOST = CUSTO DE FLUXO
C
      INTEGER X (MAXNA), RC (MAXNA)
      Real * 8 TCOST
      COMUM / ARRAYX / X / ARRAYRC / RC
      COMUM / OUTPUT / NMULTINODE, ITER, NUM_AUGM, NUM_ASCNT, NSP
C
PARÂMETROS DE TRABALHO C
C
```

```
      INTEGER I1 (MAXNN), I2 (MAXNN), I3 (MAXNN), I4 (MAXNA)
      INTEGER I5 (MAXNN), I6 (MAXNA), I7 (MAXNA)
      INTEGER TFSTOU (MAXNN), TNXTOU (MAXNA)
      INTEGER TFSTIN (MAXNN), TNXTIN (MAXNA)
      INTEGER I14 (MAXNN), I15 (MAXNN), I16 (MAXNN), I17 (MAXNN)
      LÓGICO * 1 SCAN (MAXNN), Mark (MAXNN)
      LÓGICO * 1 REPEAT
      COMUM / BLK1 / I1 / BLK2 / I2 / BLK3 / I3 / BLK4 / I4
      COMUM / BLK5 / I5 / BLK6 / I6 / BLK7 / I7
      COMUM / BLK8 / SCAN / BLK9 / MARK
      COMUM / BLK10 / TFSTOU / BLK11 / TNXTOU / BLK12 / TFSTIN / BLK13
/ TNXTIN
      COMUM / BLK14 / I14 / BLK15 / I15 / BLK16 / I16 / BLK17 / I17
      COMUM / CR / CRASH
      COMUM / BLKR / REPEAT
C
C OPCIONAL DE TRABALHO parâmetros (por ANÁLISE DE SENSIBILIDADE ONLY)
C
      INTEGER CAP (MAXNA)
      COMUM / BLKCAP / CAP
C
C declarar variáveis calendário para sistema UNIX
C
      REAL * 4. time0, TIME1, TIME2
      COMUM / T / time0, TIME1
C
C ---------------------------------------------- --------------
C
C LEIA PROBLEMA DE DADOS DE ARQUIVO RELAX4.INP
C
      PRINT *, 'ler dados problema RELAX4.INP'
      ABERTO (13, FILE = "RELAX4.INP ', STATUS =' velho ')
      Rewind (13)
C
C LEIA número de nós e arcos
C
      LER (13,1000) N, NA
C
C LEIA INÍCIO NODE, nó final, custo e capacidade de cada ARC
C
      FAZER 20 I = 1, NA
        READ (13,1000) STARTN (I), ENDN (I), C (I), L (I)
20 CONTINUAR
C
C LEIA SUPPLY de cada nó; Convertê-lo para EXIGIR
C
      FAZER 30 I = 1, N
        READ (13,1000) B (I)
        B (I) = - B (I)
30 CONTINUAR
C
1000 FORMATO (4I8)
      Rewind (13)
      CLOSE (13)
C
      PRINT *, 'FIM DE LEITURA'
      PRINT *, 'número de nós =', N ', número de arcos =', NA
C
C SET grande para um inteiro para a sua máquina
C
      GRANDE = 500000000
```

```
      DANGER_THRESH = GRANDE / 10
C
C verificação de dados está dentro do alcance MACHINE
C
      FLAG1 = 0
      Flag2 = 0
      Flag3 = 0
      DO 40 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
40 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 *, "alguns custos são perigosamente grande '
         PRINT *, "programa pode não funcionar corretamente '
      END IF
C
C ------------------------------------------------ --------------
C
C CONSTRUCT listas ligadas PARA O PROBLEMA
C
      PRINT *, 'construir listas ligadas para o problema de'

      CHAMADA INIDAT
C
C ------------------------------------------------ --------------
C
C REINICIAR PREÇOS DUAL
C (padrão: Todos os preços DUAL = 0, custam tão reduzida é fixada
C igual ao custo)
C NO CURSO do algoritmo de custo reduzido da AN
C ARC é ajustado TO
C RD (ARC) = C (ARC) + de preços implícito do ENDNODE (ARC) -
C de preços implícito do StartNode (ARC)
C O SEGUINTE COMPLEMENTAR CONDIÇÃO frouxidão
C é aplicada por RELAX em sua fase de inicialização e
C é mantida durante todo o seu curso
C IF RD (ARC) .GT.0 então o fluxo de arco deve ser igual a 0
C IF RD (ARC) .LT.0 então o fluxo de arco SEJA IGUAL ARC CAPACIDADE
C
      FAZER 60 I = 1, NA
60 RC (I) = C (I)
C
C Nota Referente reoptimization:
C a repetição variável lógica ESTÁ DEFINIDO PARA .FALSE. Se o problema
C é resolvido PELA PRIMEIRA VEZ
```

```
C
C repeat está definido como .TRUE. Se o problema for reoptimized
C de um começo HOT. Neste caso, o usuário deve assegurar que
FLUXO DE C do Arco são redefinidos para que
C Complementar frouxidão CONDIÇÃO (veja acima) é
C satisfeito após CHAMANDO RELAXE E
C DO DFCT ARRAY CORRETAMENTE CORRESPONDENTE AO REINICIAR ARC FLUXO
C
C HERE WE especificar que estamos resolvendo O PROBLEMA DO RISCO
C
      REPEAT = .FALSE.
C
C STORE CAPACIDADE DE arcos no CAP
C (opcional se ANÁLISE DE SENSIBILIDADE não será ativado)
C
      FAZER 70 I = 1, NA
70 PAC (I) = U (I)
C
C ------------------------------------------------ --------------
C
C SET CRASH igual a 1 para ativar um leilão / MENOR PARA SUBROUTINE
PATH
C OBTENÇÃO DA INICIAL PREÇO-FLOW PAIR. Isto é recomendado para DIFÍCIL
C PROBLEMAS ONDE OS RENDIMENTOS inicialização default
C LONGO SOLUÇÃO TIMES.
C
      PRINT *, "ENTER a inicialização desejada"
      PRINT *, '<0> para a inicialização do padrão'
      PRINT *, '<1> para o leilão INITIALIZATION'
      LER *, CRASH
C
C CHAMADA RELAX4 para resolver o problema
C
75 CONTINUAR
      PRINT *, "CHAMANDO RELAX4 para resolver o problema"
      PRINT *, '**********************************'
C
C inicializar o sistema de TIMER
C
C = time0 LONGO (362) /60.0
      Time0 SECNDS = (0,0)
C
      RELAX4 CHAMADA
C
C chamada de sistema Timer para exibir tempo de execução para RELAX4
C
C = TIME2 LONGO (362) /60.0 - time0
      TIME2 = SECNDS (time0)

      PRINT *, "solução TEMPO TOTAL = ', TIME2,' SECS. '
      PRINT *, 'TIME NA INICIALIZAÇÃO =', TIME1, 'SECS.'
C
C ------------------------------------------------ --------------
C
C Verifica a exactidão da parâmetros de saída
C
      FAZER 80 NODE = 1, N
        IF (B (Nó) .NE.0) THEN
          PRINT *, "excedente diferente de zero AT NODE ', NODE
        END IF
80 CONTINUAR
```

```
      FAZER 90 ARC = 1, NA
        IF (X (ARC) .GT.0) THEN
          IF (RC (ARC) .GT.0) THEN
            PRINT *, 'frouxidão COMPLEMENTAR VIOLADOS AT ARC ", ARC
          ENDIF
        ENDIF
        IF (U (ARC) .GT.0) THEN
          IF (RC (ARC) .LT.0) THEN
            PRINT *, 'frouxidão COMPLEMENTAR VIOLADOS AT ARC ", ARC
          ENDIF
        ENDIF
90 CONTINUAR
C
C calcular e exibir do custo de fluxos (em precisão dupla)
C
      TCOST = FLOAT (0)
      FAZER 100 I = 1, NA
        TCOST = TCOST DFLOAT + (X (I) C * (I))
100 CONTINUAR
      PRINT *, "custo ótimo = ', TCOST
C
C exibir estatísticas RELAX4
C
      IF (CRASH.EQ.1) THEN
        PRINT *, "número de leilão / Shortest Path = ITERATIONS ', NSP
      END IF
      PRINT *, 'número de iterações =', ITER
      PRINT *, 'NÚMERO DE Multinode ITERATIONS =', NMULTINODE
      PRINT *, 'número de passos Multinode ASCENT =', NUM_ASCNT
      PRINT *, "número de aumentos regulares = ', NUM_AUGM
      PRINT *, '**********************************'
C
C PARA ATIVAR ANÁLISE DE SENSIBILIDADE, insira o seguinte
C TRÊS LINHAS aqui.
C
      CHAMADA SENSTV
      REPEAT = .TRUE.
      IR PARA 75
C
      FIM
C
C
      SUBROUTINE INIDAT
      Implícita INTEGER (AZ)
C
C ------------------------------------------------ --------------
C
C FINALIDADE - Esta rotina constrói dois listas ligadas PARA
C DA REDE TOPOLOGY: uma lista (dado pelo FOU, NXTOU) PARA
C Os arcos OUTGOING de nós e uma única lista (dado pelo FIN,
C NXTIN) PARA A ARCS de entrada de nós. Essas duas listas
C são obrigados por RELAX4.
C
C ------------------------------------------------ --------------
C
C MAXNN = dimensão do ARRAYS NÓ de comprimento
C MAXNA = dimensão do ARRAYS ARC-length
C
      PARÂMETROS (MAXNN = 10000, MAXNA = 70000)
C
C parâmetros de entrada
```

```
C
CN = número de nós
C NA = número de arcos
C STARTN (J) = COMEÇAR nó para ARC J, J = 1, ..., NA
C ENDN (J) = TÉRMINO nó para ARC J, J = 1, ..., NA
C
      INTEGER STARTN (MAXNA), ENDN (MAXNA)
      COMUNS / matrizes / STARTN / ARRAYE / ENDN
      COMUM / INPUT / N, NA, GRANDE
C
C parâmetros de saída
C
C FOU (I) = Primeiro ARC OUT do nó i, i = 1, ..., N
C NXTOU (J) = PRÓXIMO ARC FORA DO nó inicial ARC J,
CJ = 1, ..., NA
C FIN (I) = Primeiro ARC NÓ EM I, I = 1, ..., N
C NXTIN (J) = PRÓXIMO ARC INTO o nó FINAL DE ARC J,
CJ = 1, ..., NA
C
      INTEGER FOU (MAXNN), NXTOU (MAXNA), FIN (MAXNN), NXTIN (MAXNA)
      COMUM / BLK3 / FOU / BLK4 / NXTOU / BLK5 / FIN / BLK6 / NXTIN
C
PARÂMETROS DE TRABALHO C
C
      INTEGER TEMPIn (MAXNN), TEMPOU (MAXNN)
      COMUM / BLK1 / TEMPIn / BLK2 / TEMPOU
C
C ------------------------------------------------ --------------
C
      FAZER 10 I = 1, N
        FIN (I) = 0
        Fou (I) = 0
        TEMPIn (I) = 0
10 TEMPOU (I) = 0
      FAZER 20 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
20 TEMPIn (I2) = I
      RETURN
      FIM
C
C
      RELAX4 SUBROUTINE
      Implícita INTEGER (AZ)
C
C ------------------------------------------------ --------------
C
RELAX C-IV (VERSÃO DE Outubro de 1994)
```

```
C
C LIBERAÇÃO NOTA - Esta versão de relaxamento código tem OPÇÃO PARA
CA PROCEDIMENTO ESPECIAL PARA CRASH
C o par de preço inicial-FLOW. Isto é recomendado para DIFÍCIL
C problemas onde a inicialização PADRÃO
C resulta em tempos de execução longa.
CRASH C = 1 corresponde a um leilão / MENOR método do caminho
C
C ESTES inicializações são recomendados na ausência de qualquer
C Informações prévias sobre A iniciais favoráveis FLOW-PREÇO VETOR
PAIR
C que satisfaça frouxidão COMPLEMENTAR
C
C RELAXAMENTO DA PARCELA DO CÓDIGO diferente do código relaxt-III
C e outros códigos anteriores relaxamento em que o mantém
C o conjunto de nós com déficit diferente de zero EM UMA FILA FIFO.
C como seu antecessor relaxt-III, este código mantém uma lista ligada
C DE EQUILÍBRIO (IE, DE ZERO custo reduzido) ARCS SO PARA REDUZIR
C DO TRABALHO na rotulagem e na digitalização.
C AO CONTRÁRIO relaxt-III, ele não usa SELETIVAMENTE
C caminho mais curto iterações para inicialização.
C
C ------------------------------------------------ --------------
C
C FINALIDADE - Esta rotina implementa o método de relaxamento
C DE Bertsekas E TSENG (ver [1], [2]) PARA LINEAR
C CUSTO problemas de rede ORDINÁRIA fluxo.
C
C [1] Bertsekas, DP, "uma estrutura unificada para MÉTODOS primal-dual
..."
C de programação matemática, VOL. 32, 1985, PP. 125-145.
C [2] Bertsekas, PD e TSENG, P., "Métodos de relaxamento para
C custo mínimo ... "26 operações de pesquisa, VOL., 1988, PP. 93-114.
C
C O método de relaxamento TAMBÉM É descrito nos livros:
C
C [3] Bertsekas, DP, "REDE LINEAR OTIMIZAÇÃO: algoritmos e códigos"
C MIT Press, 1991.
C [4] Bertsekas, DP E TSITSIKLIS, JN ", paralela e distribuída
C CÁLCULO: Métodos Numéricos ", Prentice-HALL, 1989.
C [5] Bertsekas, DP, "NETWORK otimização:
C CONTÍNUA E modelos discretos ", ATHENA SCIENTIFIC de 1998.
C
C
C
C ------------------------------------------------ --------------
C
C SOURCE - Esse código foi escrito por DIMITRI P. Bertsekas
C E PAULO TSENG, com uma contribuição de Jonathan ECKSTEIN
C na inicialização FASE II. O LEILÃO DE ROTINA FOI ELABORADO
C por Dimitri P. Bertsekas e baseia-se o método descrito no
C o papel:
C
C [6] Bertsekas, DP, "LEILÃO / SEQÜENCIAL MENOR algoritmo de caminho
C para o mínimo CUSTO DE FLUXO DO PROBLEMA ", RELATÓRIO tampas P-2146,
MIT, novembro 1992.
C
C para consultas sobre o código, por favor contacte:
C
C DIMITRI P. Bertsekas
C laboratório para informação e decisão SYSTEMS
```

```
C MASSACHUSETTS INSTITUTE OF TECHNOLOGY
C Cambridge, MA 02139
C (617) 253-7267, DIMITRIB@MIT.EDU
C
C ------------------------------------------------ --------------
C
C orientações de utilização -
C
C Esta rotina é de domínio público a ser usado apenas para RESEARCH
C propósitos. NÃO PODE SER USADO COMO PARTE de um produto comercial,
OR
C a satisfazer todas as exigências PARTE entrega comercial TO
C governo ou da indústria, sem o acordo prévio dos autores.
C usuários devem reconhecer a autoria do código,
C E o método de relaxamento.
C
C NO modificação deve ser feita a este Código Outro
C DO QUE O MÍNIMO NECESSÁRIO
C compatibilizar-LO COM OS Fortran of specific
MÁQUINAS C.
C
C ------------------------------------------------ --------------
C
C MAXNN = dimensão do ARRAYS NÓ de comprimento
C MAXNA = dimensão do ARRAYS ARC-length
C
      PARÂMETROS (MAXNN = 10000, MAXNA = 70000)
C
PARÂMETROS DE ENTRADA C (ver notas 1, 2, 4)
C
CN = número de nós
C NA = número de arcos
C = A GRANDE INTEIRO MUITO GRANDE PARA REPRESENTAR INFINITY
C (ver nota 3)
C REPEAT = .TRUE. Se a inicialização É a ser ignorado
C (.FALSE. OUTRO)
CRASH C = 0 se PADRÃO inicialização é USADO
C 1 IF LEILÃO inicialização é USADO
C STARTN (J) = COMEÇAR nó para ARC J, J = 1, ..., NA
C ENDN (J) = TÉRMINO nó para ARC J, J = 1, ..., NA
C FOU (I) = Primeiro ARC OUT do nó i, i = 1, ..., N
C NXTOU (J) = PRÓXIMO ARC FORA DO nó inicial ARC J,
CJ = 1, ..., NA
C FIN (I) = Primeiro ARC NÓ EM I, I = 1, ..., N
C NXTIN (J) = PRÓXIMO ARC INTO o nó FINAL DE ARC J,
CJ = 1, ..., NA
C
      INTEGER STARTN (MAXNA), ENDN (MAXNA)
      INTEGER FOU (MAXNN), NXTOU (MAXNA), FIN (MAXNN), NXTIN (MAXNA)
      LÓGICO * 1 REPEAT
      COMUM / INPUT / N, NA, GRANDE
      COMUNS / matrizes / STARTN / ARRAYE / ENDN
      COMUM / BLK3 / FOU / BLK4 / NXTOU / BLK5 / FIN / BLK6 / NXTIN
      COMUM / BLKR / REPEAT
      COMUM / CR / CRASH
C
C parâmetros atualizados (ver notas 1, 3, 4)
C
C RC (J) = custo reduzido de ARC J, J = 1, ..., NA
CU (J) = CAPACIDADE DE ARC J na entrada
C E (CAPACIDADE DE ARC J) - X (J) sobre a produção,
```

```
CJ = 1, ..., NA
C DFCT (I) = demanda no nó i na entrada
C E zero na saída, I = 1, ..., N
C
      INTEGER RC (MAXNA), U (MAXNA), DFCT (MAXNN)
      COMUM / ARRAYRC / RC / ARRAYU / U / arrayb / DFCT
C
PARÂMETROS DE SAÍDA C (ver notas 1, 3, 4)
C
CX (J) = Flow no ARC J, J = 1, ..., NA
C NMULTINODE = número de iterações Multinode relaxamento em RELAX4
C ITER = NÚMERO DE RELAXAMENTO iterações na RELAX4
C NUM_AUGM = número de passos FLUXO EM aumento RELAX4
C NUM_ASCNT = NÚMERO DE PASSOS Multinode subida no RELAX4
C NSP = número de leilão / mais curto ITERATIONS PATH
C
      INTEGER X (MAXNA)
      COMUM / ARRAYX / X
      COMUM / OUTPUT / NMULTINODE, ITER, NUM_AUGM, NUM_ASCNT, NSP
C
C parâmetros de trabalho (ver notas 1, 4, 5)
C
      INTEGER LABEL (MAXNN), PRDCSR (MAXNN), SAVE (MAXNA)
      INTEGER TFSTOU (MAXNN), TNXTOU (MAXNA), TFSTIN (MAXNN), TNXTIN
(MAXNA)
      INTEGER DDPOS (MAXNN), DDNEG (MAXNN), NXTQUEUE (MAXNN)
      INTEGER I15 (MAXNN), I16 (MAXNN), I17 (MAXNN)
      LÓGICO * 1 SCAN (MAXNN), Mark (MAXNN)
      LÓGICO * 1 FEASBL, parado, SWITCH, POSIT, pChange
      COMUM / BLK1 / LABEL / BLK2 / PRDCSR / BLK7 / SAVE
      COMUM / BLK10 / TFSTOU / BLK11 / TNXTOU / BLK12 / TFSTIN / BLK13
/ TNXTIN
      COMUM / BLK14 / NXTQUEUE / BLK15 / I15 / BLK16 / I16 / BLK17 /
I17
      COMUM / BLK8 / SCAN / BLK9 / MARK
      EQUIVALENCE (DDPOS, TFSTOU), (DDNEG, TFSTIN)
C
C parâmetros de tempo
C
      REAL * 4. time0, TIME1
      COMUM / T / time0, TIME1
C
C NOTA 1 -
C para ser executado em LIMITED MEMORY SYSTEMS, DECLARO as matrizes
C STARTN, ENDN, NXTIN, NXTOU, FIN, FOU, LABEL,
C PRDCSR, SAVE, TFSTOU, TNXTOU, TFSTIN, TNXTIN,
C DDPOS, DDNEG, NXTQUEUE AS INTEGER * 2 em vez disso.
C
C NOTA 2 -
C Esta rotina não faz nenhum esforço para inicializar COM UM FAVORÁVEL
X
C por entre aqueles VETORES fluxo que satisfazer frouxidão
COMPLEMENTAR
C com a inicial custo reduzido Vetor RC.
C se um FAVORÁVEL X é conhecida, então ele pode ser passado, JUNTO
C com os arranjos CORRESPONDENTES U e DFCT, a esta rotina
C directamente. Isso, no entanto, exige que CAPACIDADE
C PARCELA APERTO ea porção FLUXO DE INICIAÇÃO
C desta rotina (UP TO linha denominada 90) ser ignorada.
C
C NOTA 3 -
```

```
C ALL problema de dados deve ser menor do que as grandes em magnitude,
C E GRANDES deve ser menor do que, digamos, 1/4 o maior inteiro * 4
C da máquina usada. ESTE guardará PRINCIPALMENTE CONTRA
C Transbordo em PROBLEMAS incapacitado sempre que as capacidades ARC
C SÃO TOMADAS FINITE MAS MUITO GRANDE. Note, no entanto, QUE AS IN
C TODOS OS CÓDIGOS DE OPERAÇÃO com inteiros, excesso pode ocorrer se
alguns
C DA DATA DO PROBLEMA TOMA valores muito grandes.
C
C NOTA 4 -
C Cada bloco comum contém APENAS UM matriz para os vetores no RELAX4
C pode ser dimensionado para 1 e tomar a sua DIMENSÃO DA
C Chamada principal rotina. Com este truque, RELAX4 não precisam ser
recompilados
C Se as dimensões da matriz no CHAMANDO MUDANÇA DE ROTINA.
C SE SEU Fortran compilador não suporta esse recurso,
C Alterar A DIMENSÃO DE todas as matrizes de ser o mesmo como os
C declarada no programa de chamada principal.
C
C NOTA 5 -
C DDPOS E DDNEG são matrizes que dão a DERIVADOS direcional para
C ALL positivos e negativos variações de preços de nó único. Estes são
usados
C, somente na Fase II do procedimento de inicialização, antes que o
C lista ligada de ARCS BALANCEADOS vem para jogar. Portanto, para
reduzir
C STORAGE, eles são equivalência ao TFSTOU E TFSTIN,
C QUE SÃO DO TAMANHO SAME (número de nós) e são utilizados
C só depois que a árvore entra em uso.
C
C ------------------------------------------------ --------------
C
C INITIALIZATION FASE I
C
C nesta fase, nós reduzimos o CAPACIDADES ARC por tanto como
C POSSÍVEL sem mudar do problema;
C ENTÃO vamos definir o INICIAL FLUXO ARRAY X, JUNTAMENTE COM
C CORRESPONDENTE ARRAYS U E DFCT.
C
C Esta fase e fase II (DAQUI ATÉ linha denominada 90)
C pode ser ignorada (definindo REPEAT TO .TRUE). Se o programa de
chamada
LOCAIS C em valores escolhidos-comuns do usuário para o Arco fluxos,
sendo ARC RESIDUAL
CAPACIDADES C e os déficits nodal. Quando isso é feito,
C é fundamental que o fluxo ea custo reduzido para cada arco
C SATISFY frouxidão COMPLEMENTAR
C ea matriz DFCT CORRETAMENTE corresponder à ARC INICIAL / fluxos.
C
      IF (REPEAT) Vá para 90
C
      Faça 10 NODE = 1, N

        NODE_DEF = DFCT (nó)
        DDPOS (nó) = NODE_DEF
        DDNEG (nó) = - NODE_DEF
        MaxCap = 0
        SCAPOU = 0
        ARC = FOU (nó)
11 IF (ARC.GT.0) THEN
          IF (SCAPOU.LE.LARGE-U (ARC)) THEN
```

```
            SCAPOU = SCAPOU + U (ARC)
          MAIS
            IR PARA 10
          END IF
          ARC = NXTOU (ARC)
          IR PARA 11
        END IF
        IF (SCAPOU.LE.LARGE-NODE_DEF) THEN
          CAPOUT = SCAPOU + NODE_DEF
        MAIS
          IR PARA 10
        END IF
        IF (CAPOUT.LT.0) THEN
C
C PROBLEMA é inviável - SAIR
C
          PRINT *, 'Sair durante o ajustamento da capacidade'
          PRINT *, "fluxo exógeno em NODE ', nó
     $ 'Excede OUT CAPACIDADE'
          CHAMADA PRINTFLOWS (nó)
          IR PARA 4400
        END IF
C
        SCAPIN = 0
        ARC = FIN (nó)
12 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 10
          END IF
          ARC = NXTIN (ARC)
          IR PARA 12
        END IF
        IF (SCAPIN.LE.LARGE + NODE_DEF) THEN
          Capin = SCAPIN-NODE_DEF
        MAIS
          IR PARA 10
        END IF
        IF (CAPIN.LT.0) THEN
C
C PROBLEMA é inviável - SAIR
C
          PRINT *, 'Sair durante o ajustamento da capacidade'
          PRINT *, "exógenas fluir de NODE ', nó
     $ 'Excede na capacidade de'
          CHAMADA PRINTFLOWS (nó)
          IR PARA 4400
        END IF
C
        ARC = FOU (nó)
15 IF (ARC.GT.0) THEN
          U (ARC) = MIN0 (U (ARC), Capin)
          ARC = NXTOU (ARC)
          IR PARA 15
        END IF
10 CONTINUAR
C
C ---------------------------------------------- --------------
```

```
C
C INITIALIZATION FASE II
C
C nesta fase, WE inicializar os preços e flui SEJA CALLING
C DO LEILÃO DE ROTINA OU atuando somente um único nó (coordenada)
ITERATIONS C relaxamento.
C
      IF (CRASH.EQ.1) THEN
        NSP = 0
        LEILÃO DE CHAMADA
        IR PARA 70
      END IF
C
C inicializar o ARC FLUXO PARA SATISFAZER frouxidão complementares,
com
PREÇOS C. U (ARC) é a capacidade residual de arco, X (ARC) é o fluxo.
C Estes dois sempre somam capacidade total para ARC.
C também calcular a DERIVADOS direcional para cada coordenada
C e calcule a défices.
C
      FAZER 20 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
20 CONTINUAR
C
C Certifique-2 OU 3 passa por todos os nós, atuando somente
C único nó ITERATIONS relaxamento. O NÚMERO DE
C passa depende da densidade DA REDE
C
      IF (NA.GT.N * 10) THEN
        NumPasses = 2
      MAIS
        NumPasses = 3
      END IF
C
      Fazer 30 passes = 1, NumPasses

      DO 40 NODE = 1, N

        IF (DFCT (nó) .EQ. 0) Ir para 40

        IF (DDPOS (nó) .LE. 0) THEN
C
C COMPUTE DELPRC, O stepSize AO PRÓXIMO BREAKPOINT
C NO CUSTO DUAL como o preço do nó é aumentado.
C [desde a custo reduzido de ALL OUTGOING (RESP.,
C entrada) ARCS vai diminuir (RESP., AUMENTO) AS
```

```
C O PREÇO DO NÓ é aumentada, o ponto de interrupção Em seguida é
C o mínimo do custo reduzido POSITIVO NO OUTGOING
C ARCS E DO custo reduzido negativo sobre seus arcos de entrada.]
C
          DELPRC = GRANDE
          ARC = FOU (nó)
51 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 51
          END IF
          ARC = FIN (nó)
52 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 52
          END IF
C
C SE NO BREAKPOINT é de esquerda e ASCENT DUAL IS STILL
C POSSÍVEL, o problema é inviável.
C
          IF (DELPRC.GE.LARGE) THEN
            IF ((DDPOS nó) .EQ.0) GOTO 40
            GOTO 4400
          END IF
C
C DELPRC É A stepSize TO próximo ponto de interrupção. AUMENTAR
C PREÇO DO nó DELPRC e calcule a TO stepSize
C ponto de interrupção seguinte NO CUSTO DUAL.
C
53 NXTBRK = GRANDE
C
C olhar para todos ARCS OUT OF nó.
C
          ARC = FOU (nó)
54 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
C diminuem o custo reduzido EM TODOS OS arcos de saída.
C
             TRC = TRC - DELPRC
```

```
            IF ((TRC.GT.0) .E. (TRC.LT.NXTBRK)) THEN
              NXTBRK = TRC
            Else if (TRC.EQ.0) THEN
C
C ARC GOES de inativo para EQUILIBRADO. Atualizar o
C taxa de subida DUAL AT nó e AT seu vizinho.
C
              DDPOS (nó) = DDPOS (nó) + U (ARC)
              DDNEG (ENDN (ARC)) = DDNEG (ENDN (ARC)) + U (ARC)
            END IF
            RC (ARC) = TRC
            ARC = NXTOU (ARC)
            GOTO 54
          END IF
C
C olhar para todos ARCS EM nó.
C
          ARC = FIN (nó)
55 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
C de aumento o custo reduzido ON todos os arcos de entrada.
C
            TRC = TRC + DELPRC
            IF ((TRC.LT.0) .E. (TRC.GT.-NXTBRK)) THEN
              NXTBRK = -TRC
            Else if (TRC.EQ.0) THEN
C
C ARC GOES de ativo para EQUILIBRADO. Atualizar o
C taxa de subida DUAL AT nó e AT seu vizinho.
C
              DDNEG (STARTN (ARC)) = DDNEG (STARTN (ARC)) + X (ARC)
              DDPOS (nó) = DDPOS (nó) + X (ARC)
            END IF
            RC (ARC) = TRC
            ARC = NXTIN (ARC)
            GOTO 55
          END IF
C
C IF PREÇO DO nó pode ser aumentada sem diminuir
C a dupla COST (mesmo que o custo DUAL NÃO AUMENTA),
C RETURN para aumentar o preço ainda mais.
C
          IF ((DDPOS (nó) .LE.0) .E. (NXTBRK.LT.LARGE)) THEN
            DELPRC = NXTBRK
            GOTO 53
          END IF
```

```
        Else if (DDNEG (nó) .LE.0) THEN
C
C COMPUTE DELPRC, O stepSize AO PRÓXIMO BREAKPOINT
C NO CUSTO DUAL como o preço do nó é diminuída.
C [desde a custo reduzido de ALL OUTGOING (RESP.,
C entrada) ARCS AUMENTARÁ (RESP., REDUÇÃO) AS
C O PREÇO DO NÓ é diminuída, o ponto de interrupção Em seguida é
C o mínimo do custo reduzido negativo sobre OUTGOING
C ARCS E DO custo reduzido POSITIVO NO ARCS de entrada.]
C
          DELPRC = GRANDE
          ARC = FOU (nó)
61 IF (ARC.GT.0) THEN
             TRC = RC (ARC)
             IF ((TRC.LT.0) .E. (TRC.GT.-DELPRC)) THEN
                DELPRC = -TRC
             ENDIF
             ARC = NXTOU (ARC)
             GOTO 61
          END IF
          ARC = FIN (nó)
62 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 62
          END IF
C
C SE NO BREAKPOINT é de esquerda e ASCENT DUAL IS STILL
C POSSÍVEL, o problema é inviável.
C
          IF (DELPRC.EQ.LARGE) THEN
            IF (DDNEG (nó) .EQ.0) GOTO 40
            GOTO 4400
          END IF
C
C DELPRC É A stepSize TO próximo ponto de interrupção. DIMINUIÇÃO
C PREÇO DO nó DELPRC e calcule a TO stepSize
C ponto de interrupção seguinte NO CUSTO DUAL.
C
63 NXTBRK = GRANDE
C
C olhar para todos ARCS OUT OF nó.
C
          ARC = FOU (nó)
64 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
C de aumento o custo reduzido EM TODOS OS arcos de saída.
C
            TRC = TRC + DELPRC
            IF ((TRC.LT.0) .E. (TRC.GT.-NXTBRK)) THEN
              NXTBRK = -TRC
            Else if (TRC.EQ.0) THEN
C
C ARC GOES de ativo para EQUILIBRADO. Atualizar o
C taxa de subida DUAL AT nó e AT seu vizinho.
C
              DDNEG (nó) = DDNEG (nó) + X (ARC)
              DDPOS (ENDN (ARC)) = DDPOS (ENDN (ARC)) + X (ARC)
            END IF
            RC (ARC) = TRC
            ARC = NXTOU (ARC)
            GOTO 64
          END IF
C
C olhar para todos ARCS EM nó.
C
          ARC = FIN (nó)
65 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
C
C diminuem o custo reduzido ON todos os arcos de entrada.
C
            TRC = TRC - DELPRC
            IF ((TRC.GT.0) .E. (TRC.LT.NXTBRK)) THEN
              NXTBRK = TRC
            Else if (TRC.EQ.0) THEN
C
C ARC GOES de inativo para EQUILIBRADO. Atualizar o
C taxa de subida DUAL AT nó e AT seu vizinho.
C
              DDPOS (STARTN (ARC)) = DDPOS (STARTN (ARC)) + U (ARC)
              DDNEG (nó) = DDNEG (nó) + U (ARC)
            END IF
            RC (ARC) = TRC
            ARC = NXTIN (ARC)
            GOTO 65
          END IF
C
C IF PREÇO DO nó pode ser diminuiu ainda mais, sem diminuir
```

```
C a dupla COST (mesmo que o custo DUAL NÃO AUMENTA),
C RETURN para diminuir o preço ainda mais.
C
          IF ((DDNEG (nó) .LE.0) .E. (NXTBRK.LT.LARGE)) THEN
            DELPRC = NXTBRK
            GOTO 63
          END IF

        END IF

40 CONTINUAR

30 CONTINUAR
C
C
70 CONTINUAR
C
C ler o tempo para inicialização
C
C = TIME1 LONGO (362) /60.0 - time0
      TIME1 = SECNDS (time0)
C
C ---------------------------------------------- --------------
C
ESTRUTURA C INITIALIZE árvore de dados.

      FAZER 80 I = 1, N
        TFSTOU (I) = 0
        TFSTIN (I) = 0
80 CONTINUAR
      FAZER 81 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
81 CONTINUAR

C
C REINICIAR outras variáveis.
C
90 FEASBL = .TRUE.
      ITER = 0
      NMULTINODE = 0
      NUM_AUGM = 0
      NUM_ASCNT = 0
      NUM_PASSES = 0
      NUMNZ = N
      NUMNZ_NEW = 0
      CHAVE = .FALSE.
      FAZER 91 I = 1, N
        MARK (I) =. FALSE.
        SCAN (I) =. FALSE.
91 CONTINUAR
      NLABEL = 0
C
C RELAX4 utiliza uma estratégia adaptativa para decidir se
```

```
C continuam sendo o processo de digitalização depois de uma mudança
Multinode PREÇO.
C DO LIMIAR TP parâmetro eo TS que o controle
C desta estratégia são fixados nas próximas duas linhas.
C
      TP = 10
      TS = INT (n / 15)
C
C inicializar o FILA DE NÓS COM DÉFICIT diferente de zero
C
      FAZER 92 NODE = 1, N-1
        NXTQUEUE (nó) = NODE + 1
92 CONTINUAR
      NXTQUEUE (N) = 1
      NÓ = N
      LASTQUEUE = N
C
C ---------------------------------------------- --------------
C
C INICIAR O ALGORITHM relaxamento.
C
100 CONTINUAR
C
C CODE para o avanço da FILA DE NÓS DE DÉFICIT diferente de zero
C
      PrevNode = NÓ
      NÓ = NXTQUEUE (nó)
      DEFCIT = DFCT (nó)
      IF (NODE.EQ.LASTQUEUE) THEN
        NUMNZ = NUMNZ_NEW
        NUMNZ_NEW = 0
        LASTQUEUE = PrevNode
        NUM_PASSES + 1 = NUM_PASSES
      END IF
C
C CODE para excluir um nó DA FILA
C
      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)
      END IF
C
      ITER = ITER + 1
      NUMNZ_NEW = + 1 NUMNZ_NEW
C
      IF (POSIT) THEN
C
C tentar uma ITERATION único nó do nó com déficit POSITIVO
C
      PChange = .FALSE.
      INDEF = DEFCIT
      DELX = 0
```

```
      NB = 0
C
C VERIFICAÇÃO DE SAÍDA (provavelmente) ARCS equilibradas de nó.
C
      ARC = TFSTOU (nó)
4500 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 4500
      END IF
C
C CONFIRA ARCS de entrada.
C
      ARC = TFSTIN (nó)
4501 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 4501
      END IF
C
C FIM DA INICIAL SCAN nó.
C
4018 CONTINUAR
C
C SE NO PREÇO A mudança é possível, EXIT.
C
      IF (DELX.GT.DEFCIT) THEN
        SAIR = (DEFCIT .lt. INDEF)
        IR PARA 4016
      END IF
C
C RELAX4 PESQUISADOS ao longo da direção SUBIDA PARA O
C MELHOR PREÇO POR VERIFICAR A inclinação da COST DUAL
C em pontos de paragem sucessivas. Em primeiro lugar,
C calcular a distância para o ponto de quebra NEXT.
C
      DELPRC = GRANDE
      ARC = FOU (nó)
4502 IF (ARC .gt. 0) THEN
        RDCOST = RC (ARC)
        IF ((RDCOST.LT.0) .E. (RDCOST.GT.-DELPRC)) THEN
          DELPRC = -RDCOST
        END IF
        ARC = NXTOU (ARC)
        GOTO 4502
      END IF
      ARC = FIN (nó)
4503 IF (ARC .gt. 0) THEN
        RDCOST = RC (ARC)
        IF ((RDCOST.GT.0) .E. (RDCOST.LT.DELPRC)) THEN
          DELPRC = RDCOST
        END IF
        ARC = NXTIN (ARC)
```

```
        GOTO 4503
      END IF
C
C Verifique se PROBLEMA é inviável.
C
      IF ((DELX.LT.DEFCIT) .E. (DELPRC.EQ.LARGE)) THEN
C
C O CUSTO DUAL pode ser diminuída sem limites.
C
        IR PARA 4400
      END IF
C
C SKIP FLUXO ADJUSTEMT IF não há fluxo de modificar.
C
      IF (DELX.EQ.0) IR PARA 4014
C
C ajustar o fluxo sobre o incidente ARCS equilibrada para nó para
C MANTER frouxidão complementar, após a mudança de preço.
C
      FAZER 4013 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
4013 CONTINUAR
      DEFCIT = DEFCIT-DELX
4014 IF (DELPRC.EQ.LARGE) THEN
        SAIR = .TRUE.
        IR PARA 4019
      END IF
C
C nó corresponde a A DIREÇÃO DE SUBIDA DUAL. DIMINUIÇÃO
C O PREÇO DO nó DELPRC e calcule a stepSize AO
C BREAKPOINT PRÓXIMO NO CUSTO DUAL.
C
      NB = 0
      PChange = .TRUE.
      DP = DELPRC
      DELPRC = GRANDE
      DELX = 0
```

```
      ARC = FOU (nó)
4504 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 4504
      END IF
      ARC = FIN (nó)
4505 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 4505
      END IF
C
C Voltar para verificar se um outro preço a mudança é possível.
C
      IR PARA 4018
C
C Faça FLUXO DE AUMENTO NO nó.
C
4016 DO 4011 J = 1, NB
        ARC = SAVE (J)
        IF (ARC.GT.0) THEN
C
C ARC é um arco de saída do nó.
C
          NODE2 = ENDN (ARC)
          T1 = DFCT (NODE2)
          IF (T1.LT.0) THEN
C
C diminuem um défice total, diminuindo FLUXO DE ARC.
C
            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) IR PARA 4019
          END IF
        MAIS
C
```

```
C -ARC é um arco de entrada para nó.
C
          NARC = -ARC
          NODE2 = STARTN (NARC)
          T1 = DFCT (NODE2)
          IF (T1.LT.0) THEN
C
C diminuem um défice total, aumentando o fluxo DE -ARC.
C
            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) IR PARA 4019
          END IF
        END IF
4011 CONTINUAR
4019 DFCT (nó) = DEFCIT
C
C reconstruir a lista ligada de equilíbrio ARCS incidente para este
nó.
C para cada nó adjacente, WE adicionar qualquer ARCS RECENTEMENTE
blanced
C para a lista, mas não se preocupe RETIRAR AQUELES ANTERIORMENTE
EQUILIBRADO
C (eles serão removidos próximo horário de cada nó adjacente é
digitalizada).
C
      IF (pChange) THEN
        ARC = TFSTOU (nó)
        TFSTOU (nó) = 0
4506 IF (ARC .gt. 0) THEN
          NXTARC = TNXTOU (ARC)
          TNXTOU (ARC) = -1
          ARC = NXTARC
          GOTO 4506
        END IF
        ARC = TFSTIN (nó)
        TFSTIN (nó) = 0
4507 IF (ARC .gt. 0) THEN
          NXTARC = TNXTIN (ARC)
          TNXTIN (ARC) = -1
          ARC = NXTARC
          GOTO 4507
          END IF
C
C Agora adicione a ARCS ATUALMENTE equilibrada para a lista para este
nó
C (que agora está vazia) e as adjacentes apropriado.
C
          FAZER 4508 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
4508 CONTINUAR

        END IF
C
C FIM DE ITERATION NÓ ÚNICO DE POSITIVO NODE déficit.
C
      MAIS
C
C tentar uma ITERATION único nó do nó com déficit NEGATIVO
C
      PChange = .FALSE.
      DEFCIT = -DEFCIT
      INDEF = DEFCIT
      DELX = 0
      NB = 0
C
      ARC = TFSTIN (nó)
4509 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
        END IF
        ARC = TNXTIN (ARC)
        GOTO 4509
      END IF
      ARC = TFSTOU (nó)
4510 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
        END IF
        ARC = TNXTOU (ARC)
        GOTO 4510
      END IF
C
4028 CONTINUAR
      IF (DELX.GE.DEFCIT) THEN
        SAIR = (DEFCIT .lt. INDEF)
        IR PARA 4026
      END IF
C
C COMPUTE Distância até ao próximo ponto de interrupção.
C
      DELPRC = GRANDE
      ARC = FIN (nó)
4511 IF (ARC .gt. 0) THEN
        RDCOST = RC (ARC)
        IF ((RDCOST .lt. 0) .E. (RDCOST.GT.-DELPRC)) THEN
          DELPRC = -RDCOST
        END IF
        ARC = NXTIN (ARC)
```

```
        GOTO 4511
      END IF
      ARC = FOU (nó)
4512 IF (ARC .gt. 0) THEN
        RDCOST = RC (ARC)
        IF ((RDCOST.GT.0) .E. (RDCOST.LT.DELPRC)) THEN
          DELPRC = RDCOST
        END IF
        ARC = NXTOU (ARC)
        GOTO 4512
      END IF
C
C Verifique se PROBLEMA é inviável.
C
      IF ((DELX.LT.DEFCIT) .E. (DELPRC.EQ.LARGE)) THEN
        IR PARA 4400
      END IF
      IF (DELX.EQ.0) IR PARA 4024
C
C FLUXO AUGMENTATION é possível.
C
      FAZER 4023 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
4023 CONTINUAR
      DEFCIT = DEFCIT-DELX
4024 IF (DELPRC.EQ.LARGE) THEN
        SAIR = .TRUE.
        IR PARA 4029
      END IF
C
C aumento de preços AT NODE é possível.
C
      NB = 0
      PChange = .TRUE.
      DP = DELPRC
      DELPRC = GRANDE
      DELX = 0
```

```
      ARC = FIN (nó)
4513 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 4513
      END IF
      ARC = FOU (nó)
4514 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 4514
      END IF
      IR PARA 4028
C
C Faça FLUXO DE AUMENTO NO nó.
C
4026 DO 4021 J = 1, NB
        ARC = SAVE (J)
        IF (ARC.GT.0) THEN
C
C ARC é um arco de entrada para nó.
C
          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) IR PARA 4029
          END IF
        MAIS
C
C -ARC é um arco de saída do nó.
C
          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) IR PARA 4029
          END IF
        END IF
4021 CONTINUAR
4029 DFCT (nó) = - DEFCIT
C
C reconstruir a LISTA DE INCIDENTE EQUILIBRADA ARCS para o nó.
C
      IF (pChange) THEN

        ARC = TFSTOU (nó)
        TFSTOU (nó) = 0
4515 IF (ARC .gt. 0) THEN
          NXTARC = TNXTOU (ARC)
          TNXTOU (ARC) = -1
          ARC = NXTARC
          GOTO 4515
        END IF
        ARC = TFSTIN (nó)
        TFSTIN (nó) = 0
4516 IF (ARC .gt. 0) THEN
          NXTARC = TNXTIN (ARC)
          TNXTIN (ARC) = -1
          ARC = NXTARC
          GOTO 4516
        END IF
C
C Agora adicione a ARCS ATUALMENTE equilibrada para a lista para este
nó
C (que agora está vazia) e as adjacentes apropriado.
C
        FAZER 4517 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
4517 CONTINUAR

      END IF
C
C FIM DE única iteração nó para um nó DEFICIT NEGATIVO.
C
      END IF
```

```
C
      IF (QUIT.OR. (NUM_PASSES.LE.3)) ir para 100
C
C FAZER A ITERATION Multinode do nó.
C
      NMULTINODE = + 1 NMULTINODE
C
C IF número de nós DÉFICIT diferente de zero é pequeno, CONTINUAR
C de rotulagem até um aumento do fluxo é FEITO.
C
      CHAVE = (NUMNZ.LT.TP)
C
C UNMARK NODES LABELED anteriores.
C
      FAZER 4090 J = 1, NLABEL
        NODE2 = LABEL (J)
        MARK (NODE2) =. FALSE.
        SCAN (NODE2) =. FALSE.
4090 CONTINUAR
C
C REINICIAR rotulagem.
C
      NLABEL = 1
      LABEL (1) = NÓ
      MARK (nó) =. TRUE.
      PRDCSR (nó) = 0
C
C SCAN COMEÇAR nó.
C
      SCAN (nó) =. TRUE.
      NSCAN = 1
      DM = DFCT (nó)
      DELX = 0
      FAZER 4095 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
4095 CONTINUAR
C
C começar a digitalizar um nó definido mas não digitalizadas.
C
4120 NSCAN = NSCAN + 1
C
C Verifique se switch precisar ser definido como true SO TO
C continuar a digitalização mesmo após uma alteração de preço.
C
      CHAVE = CHAVE .OR.
     $ ((NSCAN .gt. TS) .E. (NUMNZ.LT.TS))
C
C DIGITALIZAÇÃO continuará até uma sobreavaliação dos RESIDUAL
C capacidade através dos CUT que corresponde ao conjunto digitalizadas
de nós (CHAMADO
C DELX) excede o valor absoluto de um défice total de digitalizada
C nós (DM chamado), OU ENTÃO UM caminho de aumento foi encontrado.
ARCS QUE SEJAM
C na árvore, mas não são equilibrados são removidas como parte de
digitalização
PROCESSO C.
C
      I = LABEL (NSCAN)
      SCAN (I) =. TRUE.
      NAUGNOD = 0
      IF (POSIT) THEN
C
C DIGITALIZAÇÃO nó i EM CASO DE DEFICIT positivo.
C
        PRVARC = 0
        ARC = TFSTOU (I)

4518 IF (ARC.GT.0) THEN
C
C ARC é um arco de saída do nó.
C
          IF (RC (ARC) .EQ. 0) THEN
            IF (X (ARC) .gt. 0) THEN
              NODE2 = ENDN (ARC)
              IF (.NOT. MARK (NODE2)) THEN
C
C NODE2 não é rotulado, então adicione NODE2 ao conjunto rotulado.
C
                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 4518
        END IF

        PRVARC = 0
        ARC = TFSTIN (I)
4519 IF (ARC.GT.0) THEN
C
C ARC é um arco que chega para dentro de nós.
C
          IF (RC (ARC) .EQ. 0) THEN
            IF (U (ARC) .gt. 0) THEN
              NODE2 = STARTN (ARC)
              IF (.NOT. MARK (NODE2)) THEN
C
C NODE2 não é rotulado, então adicione NODE2 ao conjunto rotulado.
C
                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 4519
        END IF
C
Correta C a capacidade residual da CUT NODE VARRIDAS.
C
        ARC = PRDCSR (I)
        IF (ARC.GT.0) THEN
           DELX = DELX-X (ARC)
        MAIS
           DELX = DELX-U (-ARC)
        END IF
C
C final da digitalização do nó i PARA POSITIVO CASE déficit.
C
      MAIS
```

```
C
C DIGITALIZAÇÃO NODE I FOR NEGATIVO CASE déficit.
C
        PRVARC = 0
        ARC = TFSTIN (I)
4520 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 4520
      END IF
C
      PRVARC = 0
      ARC = TFSTOU (I)
4521 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 4521
         END IF
C
         ARC = PRDCSR (I)
         IF (ARC.GT.0) THEN
            DELX = DELX-X (ARC)
         MAIS
            DELX = DELX-U (-ARC)
         END IF
       END IF
C
C Adicionar DEFICIT DE NÓ digitalizada para DM.
C
       DM = DM + DFCT (I)
C
C Verifique se o conjunto de nós digitalizada CORRESPONDE
C a uma direção ASCENT DUAL; SE SIM, REALIZE UM
C ajuste de preço STEP, caso contrário, continue rotulagem.
C
       IF (NSCAN.LT.NLABEL) THEN
         IF (Switch) IR PARA 4210
         IF ((DELX.GE.DM) .E. (DELX.GE.-DM)) Vá para 4210
       END IF
C
C tentar uma mudança de preço.
C [notar que desde DELX-ABS (DM) é uma superestimativa de ascensão
SLOPE, WE
C podem, ocasionalmente, tentar uma direção que NÃO É UMA DIREÇÃO
subida.
C NESTE CASO, as rotinas ASCNT voltar com SAIR = .FALSE.,
C Então, nós continuamos NODES rotulagem.
C
       IF (POSIT) THEN
         ASCNT1 CALL (DM, DELX, NLABEL, FEASBL,
      $ SWITCH, NSCAN, NODE, PrevNode)
         NUM_ASCNT = + 1 NUM_ASCNT
       MAIS
         ASCNT2 CALL (DM, DELX, NLABEL, FEASBL,
      $ SWITCH, NSCAN, NODE, PrevNode)
         NUM_ASCNT = + 1 NUM_ASCNT
       END IF
       IF (.NOT.FEASBL) IR PARA 4400
       IF (.NOT.SWITCH) ir para 100
C
C armazenar esses nós recém-rotulados para que fluem AUGMENTATION é
possível.
C
       NAUGNOD = 0
       FAZER 530 J = NSCAN + 1, NLABEL
         NODE2 = LABEL (J)
         IF (POSIT.AND. (DFCT (NODE2) .LT.0)) THEN
           NAUGNOD = + 1 NAUGNOD
           SAVE (NAUGNOD) = NODE2
         Else if ((.NOT.POSIT) .E. (DFCT (NODE2) .GT.0)) THEN
           NAUGNOD = + 1 NAUGNOD
           SAVE (NAUGNOD) = NODE2
```

```
        END IF
530 CONTINUAR
C
C Verifique se FLUXO AUGMENTATION é possível.
C NÃO SE, volte para digitalizar outro nó.
C
4210 CONTINUAR
C
      IF (NAUGNOD.EQ.0) IR PARA 4120
C
      FAZER 4096 J = 1, NAUGNOD
        NUM_AUGM = + 1 NUM_AUGM
        AUGNOD = SAVE (J)
        IF (POSIT) THEN
C
C FAZER o aumento do nó com déficit positivo.
C
          DX = -DFCT (AUGNOD)
          IB = AUGNOD
1500 IF (IB.NE.NODE) 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 1500
          END IF
          DX = MIN0 (DX, DFCT (nó))
          IF (DX .gt. 0) THEN
C
C Aumento (redução) de fluxo de todas FORWARD (para trás)
ARCS C no caminho do fluxo aumentado. AJUSTE NODE DEFICIT
conformidade.
C
            IF (NXTQUEUE (AUGNOD) .EQ.0) THEN
              NXTQUEUE (PrevNode) = AUGNOD
              NXTQUEUE (AUGNOD) = NÓ
              PrevNode = AUGNOD
            END IF
            DFCT (AUGNOD) = DFCT (AUGNOD) + DX
            DFCT (nó) = DFCT (nó) -DX
            IB = AUGNOD
1501 IF (IB.NE.NODE) 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 1501
            END IF
          END IF
        MAIS
```

```
C
C FAZER o aumento do nó com déficit NEGATIVO.
C
          DX = DFCT (AUGNOD)
          IB = AUGNOD
1502 IF (IB.NE.NODE) 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 1502
          END IF
          DX = MIN0 (DX, -DFCT (nó))
          IF (DX .gt. 0) THEN
C
C ATUALIZAÇÃO o fluxo e défices.
C
            IF (NXTQUEUE (AUGNOD) .EQ.0) THEN
              NXTQUEUE (PrevNode) = AUGNOD
              NXTQUEUE (AUGNOD) = NÓ
              PrevNode = AUGNOD
            END IF
            DFCT (AUGNOD) = DFCT (AUGNOD) -DX
            DFCT (nó) = DFCT (nó) + DX
            IB = AUGNOD
1503 IF (IB.NE.NODE) 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 1503
            END IF
          END IF
        END IF
        IF (DFCT (nó) .EQ.0) ir para 100
        IF (DFCT (AUGNOD) .NE.0) CHAVE = .FALSE.
4096 CONTINUAR
C
C IF nó ainda TEM DÉFICIT diferente de zero e todos os recém-
C NODES rotulado TEM SINAL MESMO PARA AS seu défice
C NODE, podemos continuar de rotulagem. NESTE CASO, CONTINUAR
C rotulagem, apenas quando o fluxo AUGMENTATION É FEITO
C relativamente frequente.
C
      IF (SWITCH.AND. (ITER.GT.8 * NUM_AUGM)) Vá para 4120
C
C RETURN para fazer outro RELAXAMENTO iteração.
C
      Ir para 100
C
```

```
C PROBLEMA é encontrado para ser inviável
C
4400 PRINT *, "o problema é encontrado para ser inviável."
      PRINT *, 'programa terminou; PRESS <CR> PARA SAIR '
      PAUSA
      PARE
C
      FIM
C
C
      LEILÃO SUBROUTINE
      Implícita INTEGER (AZ)
C
C ------------------------------------------------ --------------
C
C FINALIDADE - Esta sub-rotina usa uma versão do leilão
C ALGORITHM PARA MIN REDE COST FLUXO para calcular um
C fluxo inicial BOM e preços para o problema.
C
C ------------------------------------------------ --------------
C
C MAXNN = dimensão do ARRAYS NÓ de comprimento
C MAXNA = dimensão do ARRAYS ARC-length
C
      PARÂMETROS (MAXNN = 10000, MAXNA = 70000)
C
C parâmetros de entrada
C
CN = número de nós
C NA = número de arcos
C = A GRANDE INTEIRO MUITO GRANDE PARA REPRESENTAR INFINITY
C (ver nota 3)
C STARTN (I) = COMEÇAR nó para o ARC I-TH, I = 1, ..., NA
C ENDN (I) = TÉRMINO nó para o ARC I-TH, I = 1, ..., NA
C FOU (I) = Primeiro ARC DEIXANDO I-TH NODE, I = 1, ..., N
C NXTOU (I) = PRÓXIMO ARC saem do nó INICIAL DE J-TH ARC,
CI = 1, ..., NA
C FIN (I) = Primeiro ARC ENTRANDO I-TH NODE, I = 1, ..., N
C NXTIN (I) = PRÓXIMO ARC ENTRANDO o nó FINAL DE J-TH ARC,
CI = 1, ..., NA
C
      INTEGER STARTN (MAXNA), ENDN (MAXNA)
      INTEGER FOU (MAXNN), NXTOU (MAXNA), FIN (MAXNN), NXTIN (MAXNA)
      COMUM / INPUT / N, NA, GRANDE
      COMUNS / matrizes / STARTN / ARRAYE / ENDN
      COMUM / BLK3 / FOU / BLK4 / NXTOU / BLK5 / FIN / BLK6 / NXTIN
      COMUM / CR / CRASH
C
PARÂMETROS C ATUALIZADO
C
C RC (J) = custo reduzido de ARC J, J = 1, ..., NA
CU (J) = capacidade residual de arco J,
C                                                    J = 1, ..., NA
CX (J) = Flow no ARC J, J = 1, ..., NA
C DFCT (I) = DEFICIT no nó i, i = 1, ..., N
C
      INTEGER RC (MAXNA), U (MAXNA), X (MAXNA), DFCT (MAXNN)
      COMUM / ARRAYRC / RC / ARRAYU / U / ARRAYX / X / arrayb / DFCT
C
C parâmetros de saída
C
```

```
      COMUM / OUTPUT / NMULTINODE, ITER, NUM_AUGM, NUM_ASCNT, NSP
C
PARÂMETROS DE TRABALHO C
C
      INTEGER P (MAXNN), PRDCSR (MAXNN), SAVE (MAXNA)
      INTEGER FPUSHF (MAXNN), NXTPUSHF (MAXNA)
      INTEGER FPUSHB (MAXNN), NXTPUSHB (MAXNA)
      INTEGER NXTQUEUE (MAXNN), EXTEND_ARC (MAXNN)
      INTEGER SB_LEVEL (MAXNN), SB_ARC (MAXNN)
      LÓGICO * 1 PATH_ID (MAXNN)
      COMUM / BLK1 / P / BLK2 / PRDCSR / BLK7 / SAVE
      COMUM / BLK10 / FPUSHF / BLK11 / NXTPUSHF / BLK12 / FPUSHB /
BLK13 / NXTPUSHB
      COMUM / BLK14 / NXTQUEUE / BLK15 / EXTEND_ARC
      COMUM / BLK16 / SB_LEVEL / BLK17 / SB_ARC
      COMUM / BLK9 / PATH_ID
C
C Início de inicialização usando LEILÃO
C
      Naug = 0
      PASS = 0
      THRESH_DFCT = 0
C
C fator que pode determinar por quanto EPSILON é reduzido em EACH
MINIMIZATION
C
      FACTOR = 3
C
C NUM_PASSES determina quantas LEILÃO FASES dimensionamento SEJAM
REALIZADOS
C
      NUM_PASSES = 1

C SET ARC FLUXO PARA SATISFAZER CS e calcular MAXCOST E MINCOST

      MAXCOST = -Grande
      MINCOST = GRANDE
      FAZER 49 ARC = 1, NA
        START = STARTN (ARC)
        END = ENDN (ARC)
        RDCOST = RC (ARC)
        IF (MAXCOST.LT.RDCOST) MAXCOST = RDCOST
        IF (MINCOST.GT.RDCOST) MINCOST = RDCOST
        IF (RDCOST.LT.0) THEN
          DFCT (START) = DFCT (START) + U (ARC)
          DFCT (END) = DFCT (END) -U (ARC)
          X (ARC) = U (ARC)
          U (ARC) = 0
        MAIS
          X (ARC) = 0
        END IF
49 CONTINUAR
C
C SET EPSILON INICIAL
C
      IF ((MAXCOST-MINCOST) .GE.8) THEN
        EPS = INT ((MAXCOST-MINCOST) / 8)
      MAIS
        EPS = 1
      END IF
C
```

```
C fixar preços iniciais para ZERO
C
      FAZER 48 NODE = 1, N
        P (nó) = 0
48 CONTINUAR
C
C de inicialização usando leilão / caminhos mais curtos.
C início da fase PRIMEIRO DE ESCALA.
C
100 CONTINUAR

      PASS = PASS + 1
      IF ((PASS.EQ.NUM_PASSES) .OR. (EPS.EQ.1)) CRASH = 0
      NOLIST = 0
C
C CONSTRUCT lista de nós com resultados positivos e uma fila de
excedentes NEGATIVO
NÓS DE C
C
      FAZER 110 NÓ = 1, N
        PRDCSR (nó) = 0
        PATH_ID (nó) =. FALSE.
        EXTEND_ARC (nó) = 0
        SB_LEVEL (nó) = - GRANDE
        NXTQUEUE (nó) = NODE + 1
        IF (DFCT (nó) .GT.0) THEN
          NOLIST = NOLIST + 1
          SAVE (NOLIST) = NÓ
        END IF
110 CONTINUAR
C
      NXTQUEUE (N) = 1
      ROOT = 1
      PrevNode = N
      LASTQUEUE = N
C
C INITIALIZATION COM BAIXO ITERATIONS Para nós de EXCEDENTES NEGATIVOS
C
      FAZER 150 I = 1, NOLIST
        NÓ = SAVE (I)
        NSP = NSP + 1
C
C construir a lista de ARCS W / QUARTO PARA EMPURRAR FLUXO
C E ENCONTRAR preço adequado para a iteração BAIXO
C
        BSTLEVEL = -Grande
        FPUSHF (nó) = 0
        ARC = FOU (nó)
152 IF (ARC.GT.0) THEN
          IF (U (ARC) .GT.0) THEN
            IF (FPUSHF (nó) .EQ.0) THEN
              FPUSHF (nó) = ARC
              NXTPUSHF (ARC) = 0
              LAST = ARC
            MAIS
              NXTPUSHF (LAST) = ARC
              NXTPUSHF (ARC) = 0
              LAST = ARC
            END IF
          END IF
          IF (X (ARC) .GT.0) THEN
```

```
            NEW_LEVEL = P (ENDN (ARC)) + RC (ARC)
            IF (NEW_LEVEL.GT.BSTLEVEL) THEN
              BSTLEVEL = NEW_LEVEL
              EXTARC = ARC
            END IF
          END IF
          ARC = NXTOU (ARC)
          IR PARA 152
        END IF
C
        FPUSHB (nó) = 0
        ARC = FIN (nó)
154 IF (ARC.GT.0) THEN
          IF (X (ARC) .GT.0) THEN
            IF (FPUSHB (nó) .EQ.0) THEN
              FPUSHB (nó) = ARC
              NXTPUSHB (ARC) = 0
              LAST = ARC
            MAIS
              NXTPUSHB (LAST) = ARC
              NXTPUSHB (ARC) = 0
              LAST = ARC
            END IF
          END IF
          IF (U (ARC) .GT.0) THEN
            NEW_LEVEL = P (STARTN (ARC)) - RC (ARC)
            IF (NEW_LEVEL.GT.BSTLEVEL) THEN
              BSTLEVEL = NEW_LEVEL
              EXTARC = -ARC
            END IF
          END IF
          ARC = NXTIN (ARC)
          IR PARA 154
        END IF
        EXTEND_ARC (nó) = EXTARC
        P (nó) = BSTLEVEL-EPS

150 CONTINUAR
C
C iniciam ciclos de aumento da nova fase de dimensionamento.
C
200 CONTINUAR

      IF (DFCT (ROOT) .GE.THRESH_DFCT) GOTO 3000

      TERM = ROOT
      PATH_ID (ROOT) =. TRUE.
C
C MAIN algoritmo Forward com a raiz como a origem.
C
500 CONTINUAR
C INÍCIO DE UMA NOVA FRENTE ITERATION
C
      PTERM = P (TERM)
      EXTARC = EXTEND_ARC (TERM)

      IF (EXTARC.EQ.0) THEN
C
C construir a lista de ARCS W / QUARTO PARA EMPURRAR FLUXO
C
        FPUSHF (TERM) = 0
```

```
        ARC = FOU (TERM)
502 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 502
        END IF
C
        FPUSHB (TERM) = 0
        ARC = FIN (TERM)
504 IF (ARC.GT.0) THEN
          IF (X (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 504
        END IF

        IR PARA 600
      END IF
C
C PATH ESPECULATIVO EXTENSÃO TENTATIVA
C NOTA: ARC> 0 significa que ARC é orientada a partir da raiz para os
destinos
C ARC <0 significa que ARC é orientada a partir de Destinos para a
raiz
C EXTARC = 0 ou PRDARC = 0, maneira, o arco extensão ou o antecessor
ARC,
C, respectivamente, não foi estabelecida
C
510 CONTINUAR

      IF (EXTARC.GT.0) THEN

        IF (U (EXTARC) .EQ.0) THEN
          SECLEVEL = SB_LEVEL (TERM)
          IR PARA 580
        END IF
        END = ENDN (EXTARC)
        BSTLEVEL = P (END) + RC (EXTARC)
        IF (PTERM.GE.BSTLEVEL) THEN
          IF (PATH_ID (END)) GOTO 1200
          TERM = END
          PRDCSR (TERM) = EXTARC
```

```
          PATH_ID (TERM) =. TRUE.
C
C Se for negativo NODE excedente é encontrado, FAZER UM AUMENTO
C
          IF (DFCT (TERM) .GT.0) GOTO 2000
C
C voltar para outra ITERATION
C
          IR PARA 500
        END IF
      MAIS
        EXTARC = -EXTARC
        IF (X (EXTARC) .EQ.0) THEN
          SECLEVEL = SB_LEVEL (TERM)
          IR PARA 580
        END IF
        START = STARTN (EXTARC)
        BSTLEVEL = P (START) -RC (EXTARC)
        IF (PTERM.GE.BSTLEVEL) THEN
          IF (PATH_ID (START)) GOTO 1200
          TERM = INÍCIO
          PRDCSR (TERM) = - EXTARC
          PATH_ID (TERM) =. TRUE.
C
C Se for negativo NODE excedente é encontrado, FAZER UM AUMENTO
C
          IF (DFCT (TERM) .GT.0) GOTO 2000
C
C voltar para outra ITERATION
C
          IR PARA 500
        END IF
      END IF
C
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
C
550 SECLEVEL = SB_LEVEL (TERM)
      IF (BSTLEVEL.LE.SECLEVEL) GOTO 800
C
C IF segundo melhor PODE SER USADO fazer qualquer uma contração
C ou começar de novo com uma extensão ESPECULATIVO
C
580 IF (SECLEVEL.GT.-GRANDE) THEN
        EXTARC = SB_ARC (TERM)
        IF (EXTARC.GT.0) THEN
          IF (U (EXTARC) .EQ.0) GOTO 600
          BSTLEVEL = P (ENDN (EXTARC)) + RC (EXTARC)
        MAIS
          IF (X (-EXTARC) .EQ.0) GOTO 600
          BSTLEVEL = P (STARTN (-EXTARC)) - RC (-EXTARC)
        END IF
        IF (BSTLEVEL.EQ.SECLEVEL) THEN
          SB_LEVEL (TERM) = - GRANDE
          EXTEND_ARC (TERM) = EXTARC
          GOTO 800
        END IF
      END IF
C
C TENTATIVA DE EXTENSÃO / contração foi vencida, SO SCAN nodo terminal
```

```
C
600 CONTINUAR
      NSP = NSP + 1

      BSTLEVEL = GRANDE
      SECLEVEL = GRANDE

      ARC = FPUSHF (TERM)
700 IF (ARC.GT.0) THEN
        NEW_LEVEL = P (ENDN (ARC)) + RC (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)) - RC (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
C
C FIM DO NÓ SCAN.
C Se o nó terminal é o ROOT, ajuste o seu preço e CHANGE ROOT
C
800 IF (TERM.EQ.ROOT) THEN
        P (TERM) = BSTLEVEL + EPS
        IF (PTERM.GE.LARGE) THEN
          PRINT *, 'NO caminho para o destino'
          PRINT *, "o problema é encontrado para ser inviável."
          PRINT *, 'programa terminou; PRESS <CR> PARA SAIR '
          PAUSA
          PARE
        END IF
        PATH_ID (ROOT) =. FALSE.
        PrevNode = ROOT
```

```
        ROOT = NXTQUEUE (ROOT)
        IR PARA 200
      END IF
C
C Verifique se a extensão ou contração
C
      PRD = PRDCSR (TERM)
      IF (PRD.GT.0) THEN
        PR_TERM = STARTN (PRD)
        PREVLEVEL = P (PR_TERM) -RC (PRD)
      MAIS
        PR_TERM = ENDN (-PRD)
        PREVLEVEL = P (PR_TERM) + RC (-PRD)
      END IF
C
      IF (PREVLEVEL.GT.BSTLEVEL) THEN
C
C EXTENSÃO PATH
C
        IF (PREVLEVEL.GE.BSTLEVEL + EPS) THEN
          P (TERM) = BSTLEVEL + EPS
        MAIS
          P (TERM) = PREVLEVEL
        END IF
        IF (EXTARC.GT.0) THEN
          END = ENDN (EXTARC)
          IF (PATH_ID (END)) GOTO 1200
          TERM = END
        MAIS
          START = STARTN (-EXTARC)
          IF (PATH_ID (START)) GOTO 1200
          TERM = INÍCIO
        END IF
        PRDCSR (TERM) = EXTARC
        PATH_ID (TERM) =. TRUE.
C
C Se for negativo NODE excedente é encontrado, FAZER UM AUMENTO
C
      IF (DFCT (TERM) .GT.0) GOTO 2000
C
C voltar para outra ITERATION
C
        IR PARA 500
      MAIS
C
C CONTRAÇÃO PATH.
C
        P (TERM) = BSTLEVEL + EPS
        PATH_ID (TERM) =. FALSE.
        TERM = PR_TERM
        IF (PR_TERM.NE.ROOT) THEN
          IF (BSTLEVEL.LE.PTERM + EPS) THEN
          GOTO 2000
        END IF
      END IF
      PTERM = P (TERM)
      EXTARC = PRD
      IF (PRD.GT.0) THEN
        BSTLEVEL = BSTLEVEL + EPS + RC (PRD)
      MAIS
        BSTLEVEL = BSTLEVEL + EPS-RC (-PRD)
```

```
      END IF
C
C DO SEGUNDO A melhor teste e se isso falhar, fazer uma varredura nó
completo
C
        GOTO 550
      END IF
C
CA CICLO ESTÁ PRESTES A forma; FAZER UMA SEQUÊNCIA Retiro.
C
1200 CONTINUAR
C
      NÓ = TERM
1600 IF (NODE.NE.ROOT) THEN
        PATH_ID (nó) =. FALSE.
        PRD = PRDCSR (nó)
        IF (PRD.GT.0) THEN
          PR_TERM = STARTN (PRD)
          IF (P (PR_TERM) .EQ.P (nó) + RC (PRD) + EPS) THEN
            NÓ = PR_TERM
            GOTO 1600
          END IF
        MAIS
          PR_TERM = ENDN (-PRD)
          IF (P (PR_TERM) .EQ.P (nó) -RC (-PRD) + EPS) THEN
            NÓ = PR_TERM
            GOTO 1600
          END IF
        END IF
C
C fazer uma varredura COMPLETO E PREÇO RISE AT PR_TERM
C
        NSP = NSP + 1
        BSTLEVEL = GRANDE
        SECLEVEL = GRANDE
        ARC = FPUSHF (PR_TERM)
1700 IF (ARC.GT.0) THEN
          NEW_LEVEL = P (ENDN (ARC)) + RC (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 1700
        END IF
C
        ARC = FPUSHB (PR_TERM)
1710 IF (ARC.GT.0) THEN
          NEW_LEVEL = P (STARTN (ARC)) - RC (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 1710
        END IF

        SB_LEVEL (PR_TERM) = SECLEVEL
        SB_ARC (PR_TERM) = SECARC
        EXTEND_ARC (PR_TERM) = EXTARC

        P (PR_TERM) = BSTLEVEL + EPS
        IF (PR_TERM.EQ.ROOT) THEN
          PrevNode = ROOT
          PATH_ID (ROOT) =. FALSE.
          ROOT = NXTQUEUE (ROOT)
          GOTO 200
        END IF

        PATH_ID (PR_TERM) =. FALSE.
        PRD = PRDCSR (PR_TERM)
        IF (PRD.GT.0) THEN
          TERM = STARTN (PRD)
        MAIS
          TERM = ENDN (-PRD)
        END IF

        IF (TERM.EQ.ROOT) THEN
          PrevNode = ROOT
          PATH_ID (ROOT) =. FALSE.
          ROOT = NXTQUEUE (ROOT)
          GOTO 200
        MAIS
          GOTO 2000
        END IF

      END IF
C
C FIM DO LEILÃO / MENOR DE ROTINA PATH.
C DO AUMENTO DE RAIZ e corrigir o LISTAS DE PRESSÃO
C
2000 CONTINUAR

      INCR = -DFCT (ROOT)
      NÓ = ROOT
2050 EXTARC = EXTEND_ARC (nó)
      PATH_ID (nó) =. FALSE.
      IF (EXTARC.GT.0) THEN
        NÓ = ENDN (EXTARC)
        IF (INCR.GT.U (EXTARC)) INCR = U (EXTARC)
      MAIS
        NÓ = STARTN (-EXTARC)
        IF (INCR.GT.X (-EXTARC)) incr = X (-EXTARC)
      END IF
      IF (NODE.NE.TERM) GOTO 2050
      PATH_ID (TERM) =. FALSE.
      IF (DFCT (TERM) .GT.0) THEN
        IF (INCR.GT.DFCT (TERM)) INCR = DFCT (TERM)
```

```
      END IF
C
      NÓ = ROOT
2100 EXTARC = EXTEND_ARC (nó)
      IF (EXTARC.GT.0) THEN
        END = ENDN (EXTARC)
C
C Adicionar ARC ao gráfico REDUZIDA
C
        IF (X (EXTARC) .EQ.0) THEN
          NXTPUSHB (EXTARC) = FPUSHB (END)
          FPUSHB (END) = EXTARC
          NEW_LEVEL = P (nó) -RC (EXTARC)
          IF (SB_LEVEL (END) .GT.NEW_LEVEL) THEN
            SB_LEVEL (END) = NEW_LEVEL
            SB_ARC (END) = - EXTARC
          END IF
        END IF
        X (EXTARC) = X (EXTARC) + INCR
        L (EXTARC) = L (EXTARC) -INCR
C
C Retire ARC a partir do gráfico REDUZIDA
C
        IF (U (EXTARC) .EQ.0) THEN
          NAS = NAS + 1
          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
        END IF

2250 NODE = END

      MAIS
        EXTARC = -EXTARC
        START = STARTN (EXTARC)
C
C Adicionar ARC ao gráfico REDUZIDA
C
        IF (U (EXTARC) .EQ.0) THEN
          NXTPUSHF (EXTARC) = FPUSHF (START)
          FPUSHF (START) = EXTARC
          NEW_LEVEL = P (nó) + RC (EXTARC)
          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
```

```
        X (EXTARC) = X (EXTARC) -INCR
C
C Retire ARC a partir do gráfico REDUZIDA
C
        IF (X (EXTARC) .EQ.0) THEN
          NAS = NAS + 1
          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
        END IF

2350 NODE = INÍCIO

      END IF

      IF (NODE.NE.TERM) GOTO 2100
      DFCT (TERM) = DFCT (TERM) -INCR
      DFCT (ROOT) = DFCT (ROOT) + INCR
C
C INSERIR PRAZO NA FILA Se ele tem um excedente grande o suficiente
C
      IF (DFCT (TERM) .LT.THRESH_DFCT) THEN
        IF (NXTQUEUE (TERM) .EQ.0) THEN
          NXTNODE = NXTQUEUE (ROOT)
          IF ((P (TERM) .GE.P (NXTNODE)). E. (ROOT.NE.NXTNODE)) THEN
            NXTQUEUE (ROOT) = TERM
            NXTQUEUE (TERM) = NXTNODE
          MAIS
            NXTQUEUE (PrevNode) = TERM
            NXTQUEUE (TERM) = ROOT
            PrevNode = TERM
          END IF
        END IF
      END IF
C
C Se a raiz tem um excedente grande o suficiente, MANTENHA-
C na fila e voltar para outra ITERATION
C
      IF (DFCT (ROOT) .LT.THRESH_DFCT) THEN
        PrevNode = ROOT
        ROOT = NXTQUEUE (ROOT)
        IR PARA 200
      END IF
C
C FIM DO AUMENTO DO CICLO
C
3000 CONTINUAR
C
```

```
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.
C
      NXTNODE = NXTQUEUE (ROOT)
      IF (ROOT.NE.NXTNODE) THEN
        NXTQUEUE (ROOT) = 0
        NXTQUEUE (PrevNode) = NXTNODE
        ROOT = NXTNODE
        IR PARA 200
      END IF
C
C FIM DE subproblema (escalonamento FASE).
C
3600 CONTINUAR
C
C REDUZIR EPSILON.
C
      EPS = INT (EPS / FACTOR)
      IF (EPS.LT.1) EPS = 1
      THRESH_DFCT = INT (THRESH_DFCT / FACTOR)
      IF (EPS.EQ.1) THRESH_DFCT = 0
C
C Se outro LEILÃO DE ESCALA FASE PERMANECE, redefinir o FLUXO & LISTAS
DE PRESSÃO
C MAIS REAJUSTE ARC FLUXO PARA SATISFAZER CS e calcular custos
reduzidos de
C
      IF (CRASH.EQ.1) THEN

        FAZER 3800 ARC = 1, NA
          START = STARTN (ARC)
          END = ENDN (ARC)
          Pinicial = P (START)
          PEND = P (END)
          IF (PSTART.GT.PEND + EPS + RC (ARC)) THEN
            RESID = U (ARC)
            IF (RESID.GT.0) THEN
              DFCT (START) = DFCT (START) + RESID
              DFCT (END) = DFCT (END) -RESID
              X (ARC) = X (ARC) + RESID
              U (ARC) = 0
            END IF
          MAIS
            IF (PSTART.LT.PEND-EPS + RC (ARC)) THEN
              FLUXO = X (ARC)
              IF (FLOW.GT.0) THEN
                DFCT (START) = DFCT (START) -FLOW
                DFCT (END) = DFCT (END) + FLUXO
                X (ARC) = 0
                U (ARC) = U (ARC) + FLUXO
              END IF
            END IF
          END IF
3800 CONTINUAR
C
C voltar para outra FASE
C
3850 CONTINUAR
        GOTO 100

      MAIS
```

```
        CRASH = 1
        FAZER 3900 ARC = 1, NA
          START = STARTN (ARC)
          END = ENDN (ARC)
          RED_COST = RC (ARC) + P (END) -P (START)
          IF (RED_COST.LT.0) THEN
            RESID = U (ARC)
            IF (RESID.GT.0) THEN
              DFCT (START) = DFCT (START) + RESID
              DFCT (END) = DFCT (END) -RESID
              X (ARC) = X (ARC) + RESID
              U (ARC) = 0
            END IF
          MAIS
            IF (RED_COST.GT.0) THEN
              FLUXO = X (ARC)
              IF (FLOW.GT.0) THEN
                DFCT (START) = DFCT (START) -FLOW
                DFCT (END) = DFCT (END) + FLUXO
                X (ARC) = 0
                U (ARC) = U (ARC) + FLUXO
              END IF
            END IF
          END IF
          RC (ARC) = RED_COST
3900 CONTINUAR

      END IF

      RETURN
      FIM
C
C
      PRINTFLOWS sub-rotina (nó)
      Implícita INTEGER (AZ)
C
C ---------------------------------------------- --------------
C
C FINALIDADE - Isso imprime ROTINA DO DEFICIT e os fluxos
C de arcos incidente ao nó. É UTILIZADO PARA DIAGNÓSTICO
C EFEITOS EM CASO DE UM PROBLEMA inviável aqui.
C ele também pode ser usado PARA MAIS DE DIAGNÓSTICO GERAL
C propósitos.
C
C ---------------------------------------------- --------------
C
C MAXNN = dimensão do ARRAYS NÓ de comprimento
C MAXNA = dimensão do ARRAYS ARC-length
C
      PARÂMETROS (MAXNN = 10000, MAXNA = 70000)
C
      COMUNS / matrizes / STARTN / ARRAYE / ENDN / ARRAYU / U / ARRAYX
/ X
     $ / Arrayb / DFCT / BLK3 / FOU / BLK4 / NXTOU / BLK5 / FIN / BLK6
/ NXTIN
C
      INTEGER STARTN (MAXNA), ENDN (MAXNA), L (MAXNA), X (MAXNA), DFCT
(MAXNN)
      INTEGER FOU (MAXNN), NXTOU (MAXNA)
      INTEGER FIN (MAXNN), NXTIN (MAXNA)
```

```
C
C ------------------------------------------------ --------------
C
      PRINT *, «défice (IE, rede de fluxo OUT) de um nó = ', DFCT (nó)
      PRINT *, "fluxos e capacidades dos INCIDENTE arcos de nó ', NODE
C
C Verifique todos os arcos SAEM NÓ
C
      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
C
C Verifique todos os arcos de entrada para nó
C
      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
C
      RETURN
      FIM
C
C
      SUBROUTINE ASCNT1 (DM, DELX, NLABEL, FEASBL, SWITCH,
     $ NSCAN, CURNODE, PrevNode)
      Implícita INTEGER (AZ)
C
C ------------------------------------------------ --------------
C
C FINALIDADE - Esta sub-rotina executa as PREÇO vários nós
C passo de ajuste para o caso em que os nós VARRIDAS
C têm DEFICIT positivo. Ele primeiro verifica IF DIMINUINDO
C DO PREÇO dos nós digitalizada aumenta o custo DUAL.
C Se sim, então ele diminui o preço de todos os nós digitalizada.
C há duas possibilidades de preços Redução:
C Se o interruptor = .TRUE., Então o conjunto de nós digitalizada
C corresponde a uma DIREÇÃO elementar de MÁXIMA
C taxa de subida, CASO EM QUE O PREÇO DO ALL VARRIDAS
NÓS DE C são diminuídos até o próximo ponto de interrupção no
C CUSTO DUAL é encontrado. AT Neste ponto, alguns ARC
C TORNA NODE equilibrada e mais (S) são adicionados à
C rotulado definido eo SUBROUTINE é encerrado.
C Se o interruptor = .FALSE., Então o preço de todos os nós
digitalizada
```

```
C são reduzidos até que a taxa de ascensão TORNA-SE
C NEGATIVO (Isto corresponde ao ajuste de preço
C passo em que tanto a linha de pesquisa ea DEGENERATE
C ITERATION ASCENT são implementadas).
C
C ------------------------------------------------ --------------
C
C MAXNN = dimensão do ARRAYS NÓ de comprimento
C MAXNA = dimensão do ARRAYS ARC-length
C
      PARÂMETROS (MAXNN = 10000, MAXNA = 70000)
C
C parâmetros de entrada
C
C DM = DEFICIT total de nós digitalizada
C = CHAVE .TRUE. Se a rotulagem é continuar depois do preço CHANGE
C NSCAN = número de nós digitalizada
C = CURNODE nó mais recém-digitalizada
CN = número de nós
C NA = número de arcos
C = A GRANDE INTEIRO MUITO GRANDE PARA REPRESENTAR INFINITY
C (ver nota 3)
C STARTN (I) = COMEÇAR nó para o ARC I-TH, I = 1, ..., NA
C ENDN (I) = TÉRMINO nó para o ARC I-TH, I = 1, ..., NA
C FOU (I) = Primeiro ARC DEIXANDO I-TH NODE, I = 1, ..., N
C NXTOU (I) = PRÓXIMO ARC saem do nó INICIAL DE J-TH ARC,
CI = 1, ..., NA
C FIN (I) = Primeiro ARC ENTRANDO I-TH NODE, I = 1, ..., N
C NXTIN (I) = PRÓXIMO ARC ENTRANDO o nó FINAL DE J-TH ARC,
CI = 1, ..., NA
C
      INTEGER STARTN (MAXNA), ENDN (MAXNA)
      INTEGER FOU (MAXNN), NXTOU (MAXNA), FIN (MAXNN), NXTIN (MAXNA)
      COMUM / INPUT / N, NA, GRANDE
      COMUNS / matrizes / STARTN / ARRAYE / ENDN
      COMUM / BLK3 / FOU / BLK4 / NXTOU / BLK5 / FIN / BLK6 / NXTIN
C
PARÂMETROS C ATUALIZADO
C
C = DELX A estimativa mais baixa do fluxo total ON ARCS EQUILIBRADO
C NA CUT-nodes digitalizada
C NLABEL = número de nós rotulados
C = FEASBL .FALSE. Se o problema for considerada inviável
C PrevNode = O nó antes CURNODE na fila
C RC (J) = custo reduzido de ARC J, J = 1, ..., NA
CU (J) = capacidade residual de arco J,
CJ = 1, ..., NA
CX (J) = Flow no ARC J, J = 1, ..., NA
C DFCT (I) = DEFICIT no nó i, i = 1, ..., N
C LABEL (K) = NODE LABELED K-TH, K = 1, NLABEL
C PRDCSR (I) = antecessor do nó i IN árvore de nós rotulados
C (O IF I IS UNLABELED), I = 1, ..., N
C TFSTOU (I) = primeiro equilibrada ARC OUT do nó i, i = 1, ..., N
C TNXTOU (J) = PRÓXIMO EQUILIBRADA ARC FORA DO nó inicial ARC J,
CJ = 1, ..., NA
C TFSTIN (I) = ARC primeiro equilibrada INTO nó i, i = 1, ..., N
C TNXTIN (J) = ARC EQUILIBRADA PRÓXIMO NO NÓ FINAL DE ARC J,
CJ = 1, ..., NA
C NXTQUEUE (I) = NODE seguinte nó I NA FILA FIFO
C (0 se nó não está na fila), I = 1, ..., N
C SCAN (I) = .TRUE. Se o nó I é digitalizado, I = 1, ..., N
```

```
C MARK (I) = .TRUE. Se o nó é rotulado I, I = 1, ..., N
C
      INTEGER RC (MAXNA), U (MAXNA), X (MAXNA), DFCT (MAXNN)
      INTEGER LABEL (MAXNN), PRDCSR (MAXNN)
      INTEGER TFSTOU (MAXNN), TNXTOU (MAXNA), TFSTIN (MAXNN), TNXTIN
(MAXNA)
      INTEGER NXTQUEUE (MAXNN)
      LÓGICO * 1 SCAN (MAXNN), Mark (MAXNN)
      COMUM / ARRAYRC / RC / ARRAYU / U / ARRAYX / X / arrayb / DFCT
      COMUM / BLK1 / LABEL / BLK2 / PRDCSR
      COMUM / BLK10 / TFSTOU / BLK11 / TNXTOU / BLK12 / TFSTIN / BLK13
/ TNXTIN
      COMUM / BLK14 / NXTQUEUE
      COMUM / BLK8 / SCAN / BLK9 / MARK
C
PARÂMETROS DE TRABALHO C
C
      INTEGER SAVE (MAXNA)
      LÓGICO * 1 SWITCH, FEASBL
      COMUM / BLK7 / SAVE
C
C armazenar os arcos entre o conjunto de nós digitalizado e
C seu complemento no SAVE e calcular DELPRC, O stepSize
C PARA A PRÓXIMA BREAKPOINT NO CUSTO DUAL NA DIREÇÃO
C da diminuição dos preços dos nós digitalizada.
C [os arcos são armazenados na SALVAR, olhando para a ARCS
C INCIDENTE, quer ao conjunto de nós digitalizada ou ITS
C COMPLEMENTO, dependendo se NSCAN> N / 2 ou não.
C ESTE melhora a eficiência do armazenamento.]
C
      DELPRC = GRANDE
      DLX = 0
      NSalvar = 0
      IF (NSCAN.LE.N / 2) THEN
        DO 1 I = 1, NSCAN
          NÓ = LABEL (I)
          ARC = FOU (nó)
500 IF (ARC.GT.0) THEN
C
C ARC pontos em NODE digitalizada para um nó não digitalizadas.
C
            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
C ARC pontos em NODE unscanned ao nó VARRIDAS.
C
            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
C Verifique se o SET DE digitalizada NÓS VERDADEIRAMENTE CORRESPONDE
C a uma direção ASCENT DUAL. [AQUI DELX + DLX é exatamente
C soma dos fluxos ON ARCS do conjunto digitalizado para o
C unscanned SET PLUS THE (CAPACIDADE - FLUXO) ON ARCS DA
C DO unscanned conjunto para o de VARRIDAS.]
C Se não fosse este o caso, interruptor na .TRUE.
C E SAÍDA SUBROUTINE.
C
      IF ((.NOT.SWITCH) .E. (DELX + DLX.GE.DM)) THEN
        CHAVE = .TRUE.
```

```
        RETURN
      END IF
      DELX = DELX + DLX
C
C controle que o problema é viável.
C
4 IF (DELPRC.EQ.LARGE) THEN
C
WE C pode aumentar o custo DUAL sem limites, SO
C THE PRIMAL PROBLEMA é inviável.
C
        FEASBL = .FALSE.
        RETURN
      END IF
C
C reduzir os preços dos nós digitalizada, adicione mais
C nós ao SET rotulados e verificar se um nó recém-LABELED
C TEM DÉFICIT NEGATIVO.
C
      IF (CHAVE) THEN
        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
              IF (.NOT.MARK (NODE2)) THEN
                PRDCSR (NODE2) = ARC
                NLABEL = + 1 NLABEL
                LABEL (NLABEL) = NODE2
                MARK (NODE2) =. TRUE.
              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
              IF (.NOT.MARK (NODE2)) THEN
                PRDCSR (NODE2) = - ARC
                NLABEL = + 1 NLABEL
                LABEL (NLABEL) = NODE2
                MARK (NODE2) =. TRUE.
              END IF
            END IF
```

```
          END IF
7 CONTINUAR
        RETURN

      MAIS
C
C reduzir os preços dos nós examinadas pelo DELPRC.
C Ajuste FLUXO DE MANTER frouxidão complementares com
C os preços.
C
      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) = CURNODE
              PrevNode = T3
            END IF
            T3 = ENDN (ARC)
            DFCT (T3) = DFCT (T3) + T2
            IF (NXTQUEUE (T3) .EQ.0) THEN
              NXTQUEUE (PrevNode) = T3
              NXTQUEUE (T3) = CURNODE
              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) = CURNODE
                PrevNode = T3
              END IF
              T3 = ENDN (ARC)
              DFCT (T3) = DFCT (T3) -T2
              IF (NXTQUEUE (T3) .EQ.0) THEN
                NXTQUEUE (PrevNode) = T3
                NXTQUEUE (T3) = CURNODE
                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
C
      IF (DELX.LE.DM) THEN
C
C DO JOGO DE NÓS AINDA digitalizado corresponde a um
C DUAL (possivelmente DEGENERATE) directon subida. COMPUTE
C DO stepSize DELPRC AO PRÓXIMO ponto de interrupção no
C CUSTO DUAL.
C
        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
C adicionar novos ARCS equilibrada para a SUPERSET de arcos
equilibrado.
C
      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
C
C
      SUBROUTINE ASCNT2 (DM, DELX, NLABEL, FEASBL, SWITCH,
     $ NSCAN, CURNODE, PrevNode)
      Implícita INTEGER (AZ)
C
C ------------------------------------------------ --------------
C
```

```
C FINALIDADE - Essa rotina é análogo ao ASCNT MAS PARA
C o caso em que os nós digitalizadas terem DEFICIT NEGATIVO.
C
C ------------------------------------------------ -------------
C
C MAXNN = dimensão do ARRAYS NÓ de comprimento
C MAXNA = dimensão do ARRAYS ARC-length
C
      PARÂMETROS (MAXNN = 10000, MAXNA = 70000)
C
      INTEGER STARTN (MAXNA), ENDN (MAXNA)
      INTEGER FOU (MAXNN), NXTOU (MAXNA), FIN (MAXNN), NXTIN (MAXNA)
      COMUM / INPUT / N, NA, GRANDE
      COMUNS / matrizes / STARTN / ARRAYE / ENDN
      COMUM / BLK3 / FOU / BLK4 / NXTOU / BLK5 / FIN / BLK6 / NXTIN
      INTEGER RC (MAXNA), U (MAXNA), X (MAXNA), DFCT (MAXNN)
      INTEGER LABEL (MAXNN), PRDCSR (MAXNN)
      INTEGER TFSTOU (MAXNN), TNXTOU (MAXNA), TFSTIN (MAXNN), TNXTIN
(MAXNA)
      INTEGER NXTQUEUE (MAXNN)
      LÓGICO * 1 SCAN (MAXNN), Mark (MAXNN)
      COMUM / ARRAYRC / RC / ARRAYU / U / ARRAYX / X / arrayb / DFCT
      COMUM / BLK1 / LABEL / BLK2 / PRDCSR
      COMUM / BLK10 / TFSTOU / BLK11 / TNXTOU / BLK12 / TFSTIN / BLK13
/ TNXTIN
      COMUM / BLK14 / NXTQUEUE
      COMUM / BLK8 / SCAN / BLK9 / MARK
      INTEGER SAVE (MAXNA)
      LÓGICO * 1 SWITCH, FEASBL
      COMUM / BLK7 / SAVE
C
C armazenar os arcos entre o conjunto de nós digitalizado e
C seu complemento no SAVE e calcular DELPRC, O stepSize
C PARA A PRÓXIMA BREAKPOINT NO CUSTO DUAL NA DIREÇÃO
C o aumento dos preços dos nós digitalizada.
C
      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
C
      IF ((.NOT.SWITCH) .E. (DELX + DLX.GE.-DM)) THEN
        CHAVE = .TRUE.
        RETURN
      END IF
      DELX = DELX + DLX
C
C controle que o problema é viável.
C
4 IF (DELPRC.EQ.LARGE) THEN
        FEASBL = .FALSE.
        RETURN
      END IF
C
```

```
C aumentar os preços dos os nós digitalizada, adicione mais
C nós ao SET rotulados e verificar se um nó recém-LABELED
C TEM DÉFICIT positivo.
C
      IF (CHAVE) THEN

        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
              IF (.NOT.MARK (NODE2)) THEN
                PRDCSR (NODE2) = ARC
                NLABEL = + 1 NLABEL
                LABEL (NLABEL) = NODE2
                MARK (NODE2) =. TRUE.
              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
              IF (.NOT.MARK (NODE2)) THEN
                PRDCSR (NODE2) = - ARC
                NLABEL = + 1 NLABEL
                LABEL (NLABEL) = NODE2
                MARK (NODE2) =. TRUE.
              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) = CURNODE
                PrevNode = T3
              END IF
              T3 = ENDN (ARC)
              DFCT (T3) = DFCT (T3) + T2
              IF (NXTQUEUE (T3) .EQ.0) THEN
                NXTQUEUE (PrevNode) = T3
                NXTQUEUE (T3) = CURNODE
                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) = CURNODE
                PrevNode = T3
              END IF
              T3 = ENDN (ARC)
              DFCT (T3) = DFCT (T3) -T2
              IF (NXTQUEUE (T3) .EQ.0) THEN
                NXTQUEUE (PrevNode) = T3
                NXTQUEUE (T3) = CURNODE
                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
C
      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
C adicionar novos ARCS equilibrada para a SUPERSET de arcos
equilibrado.
C
      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
C
C
      SENSTV SUBROUTINE
      Implícita INTEGER (AZ)
C
C ------------------------------------------------ --------------
C
C FINALIDADE - Esta sub-rotina permite ao usuário de forma interativa
C QUER MUDAR DE FORNECIMENTO nó ou alteração do fluxo limite superior
C de um arco existentes, ou Change custo de um ARC existente,
C ou excluir um ARC existente, ou adicionar um ARC.
C [NOTA: Se o sistema operacional redefine todas as variáveis LOCAIS
C PARA ALGUNS Valor padrão cada vez que este SUBROUTINE é chamado,
C seguida, o usuário deve fazer as seguintes ATUALMENTE LOCAL
C variáveis:
C Delarc, DARC, DU, ADDARC, AARC
C GLOBAL (POR QUALQUER colocá-los em um bloco comum OR
C passá-los através do parâmetro de chamada).]
C
C ------------------------------------------------ --------------
C
C MAXNN = dimensão do ARRAYS NÓ de comprimento
C MAXNA = dimensão do ARRAYS ARC-length
C
      PARÂMETROS (MAXNN = 10000, MAXNA = 70000)
C
C parâmetros de entrada
C
CN = número de nós
C NA = número de arcos
C = A GRANDE INTEIRO MUITO GRANDE PARA REPRESENTAR INFINITY
```

```
C STARTN (J) = COMEÇAR nó para ARC J, J = 1, ..., NA
C ENDN (J) = TÉRMINO nó para ARC J, J = 1, ..., NA
C FOU (I) = Primeiro ARC OUT do nó i, i = 1, ..., N
C NXTOU (J) = PRÓXIMO ARC FORA DO nó inicial ARC J,
CJ = 1, ..., NA
C FIN (I) = Primeiro ARC NÓ EM I, I = 1, ..., N
C NXTIN (J) = PRÓXIMO ARC INTO o nó FINAL DE ARC J,
CJ = 1, ..., NA
C REPEAT = .TRUE. IF CAP (J) = X (J) + U (J), J = 1, ..., NA, na
entrada
C (.FALSE. OUTRO)
C
      INTEGER STARTN (MAXNA), ENDN (MAXNA)
      INTEGER FOU (MAXNN), NXTOU (MAXNA), FIN (MAXNN), NXTIN (MAXNA)
      LÓGICO * 1 REPEAT
      COMUM / INPUT / N, NA, GRANDE
      COMUNS / matrizes / STARTN / ARRAYE / ENDN
      COMUM / BLK3 / FOU / BLK4 / NXTOU / BLK5 / FIN / BLK6 / NXTIN
      COMUM / BLKR / REPEAT
C
PARÂMETROS C ATUALIZADO
C
CC (J) = CUSTO DE ARC J, J = 1, ..., NA
C CAP (J) = CAPACIDADE DE ARC J, J = 1, ..., NA
C RC (J) = custo reduzido de ARC J, J = 1, ..., NA
CX (J) = Flow no ARC J, J = 1, ..., NA
CU (J) = CAP (J) - X (J) sobre a produção, J = 1, ..., NA
C DFCT (I) = demanda no nó i na entrada
C E zero na saída, I = 1, ..., N
C TFSTOU (I) = primeiro equilibrada ARC OUT do nó i, i = 1, ..., N
C TNXTOU (J) = PRÓXIMO EQUILIBRADA ARC FORA DO nó inicial ARC J,
CJ = 1, ..., NA
C TFSTIN (I) = ARC primeiro equilibrada INTO nó i, i = 1, ..., N
C TNXTIN (J) = ARC EQUILIBRADA PRÓXIMO NO NÓ FINAL DE ARC J,
CJ = 1, ..., NA
C
      INTEIRO C (MAXNA), CAP (MAXNA)
      INTEGER RC (MAXNA), X (MAXNA), U (MAXNA), DFCT (MAXNN)
      INTEGER TFSTOU (MAXNN), TNXTOU (MAXNA), TFSTIN (MAXNN), TNXTIN
(MAXNA)
      COMUM / ARRAYC / C / BLKCAP / CAP
      COMUM / ARRAYRC / RC / ARRAYX / X / ARRAYU / U / arrayb / DFCT
      COMUM / BLK10 / TFSTOU / BLK11 / TNXTOU / BLK12 / TFSTIN / BLK13
/ TNXTIN
C
PARÂMETROS DE TRABALHO C
C
      INTEGER LABEL (MAXNN), PREÇO (MAXNN), SAVE (MAXNA)
      LÓGICO * 1 ADDARC, Delarc, MARK (MAXNN)
      COMUM / BLK1 / LABEL / BLK2 / preço / BLK7 / SAVE
      COMUM / BLK9 / MARK
C
      IF (.NOT.REPEAT) THEN
C
C restabelecer a capacidade ARC à do problema original
C [recordar que, na solução do problema original, RELAX4
C pode ter diminuído as capacidades ARC DURANTE
C INITIALIZATION FASE I.] Em seguida, atualize fluxo ea
DEFICITS C nó para corresponder a esse "novo" capacidade.
C
        FAZER 10 I = 1, NA
```

```
          IF (RC (I) .LT.0) THEN
            DFCT (STARTN (I)) = DFCT (STARTN (I)) + CAP (I) -X (I)
            DFCT (ENDN (I)) = DFCT (ENDN (I)) - PAC (I) + X (I)
            X (I) = PAC (I)
          MAIS
            U (I) = PAC (I) -X (I)
          END IF
10 CONTINUAR
        REPEAT = .TRUE.
      END IF
20 ESCREVA (6,30)
        ESCREVA (6,40)
        ESCREVA (6,50)
        ESCREVA (6,60)
        ESCREVA (6,70)
        ESCREVA (6,80)
        IF (ADDARC) ESCREVA (6,90) AARC
        IF (Delarc) Write (6100) DARC
        Write (6105)
30 FORMATO ("Entrada 0 para resolver o problema modificado")
40 FORMATO ('1 para alterar NODE FLUXO DE FORNECIMENTO')
50 FORMATO ('2 para mudar ARC FLUXO limite superior')
60 FORMATO ('3 para alterar ARC COST')
70 FORMATO ('4 Para apagar um ARC')
80 FORMATO ('5 Para adicionar um ARC')
90 FORMATO ('6 para excluir o último ARC', I8 ", acrescentou ')
100 FORMATO ('7 para restaurar LAST ARC', I8, 'excluídos')
105 FORMATO ('8 PARA SAIR DO PROGRAMA')
        READ (5 *) SEL
        IF (SEL.EQ.8) THEN
          PARE
        Else if (SEL.EQ.0) THEN
          RETURN
        Else if (SEL.EQ.1) THEN
C
C CHANGE NÓ DE FLUXO DE ALIMENTAÇÃO.
C
110 Write (6120)
120 formato ('nó de entrada # onde o fluxo de oferta reforçada')
          READ (5 *), o nó
          IF ((NODE.LE.0) .OR. (NODE.GT.N)) Vá para 110
          Write (6130)
130 FORMATO ('ENTRADA valor do aumento (<0 valor permitido)')
          READ (5 *) DELSUP
          DFCT (nó) = DFCT (nó) -DELSUP
140 Write (6150)
150 FORMATO ("nó de entrada NO. Onde o fluxo SUPPLY é diminuída ')
          READ (5 *), o nó
          IF ((NODE.LE.0) .OR. (NODE.GT.N)) Vá para 140
          DFCT (nó) = DFCT (nó) + DELSUP
        Else if (SEL.EQ.2) THEN
C
C Alterar ARC FLUXO DE CAPACIDADE.
C [NOTA: U é a capacidade residual, IE, U = CAPACIDADE - X.]
C
160 Write (6170)
170 FORMATO ('ENTRADA NO ARC. E o aumento da cota superior')
          READ (5 *) ARC, DELUB
          IF ((ARC.LE.0) .OR. (ARC.GT.NA)) Vá para 160
          IF (RC (ARC) .LT.0) THEN
C
```

```
C ARC está ativo, SO manter o fluxo de AT (NOVO) de capacidade.
C
            DFCT (STARTN (ARC)) = DFCT (STARTN (ARC)) + DELUB
            DFCT (ENDN (ARC)) = DFCT (ENDN (ARC)) - DELUB
            X (ARC) = X (ARC) + DELUB
            IF (X (ARC) .LT.0) Write (6180)
          Else if (RC (ARC) .EQ.0) THEN
            IF (U (ARC) .GE.-DELUB) THEN
              U (ARC) = U (ARC) + DELUB
            MAIS
C
C NEW capacidade for inferior fluxo de corrente, SO DIMINUIÇÃO
C fluxo para nova capacidade.
C
              DEL = -DELUB-U (ARC)
              DFCT (STARTN (ARC)) = DFCT (STARTN (ARC)) - DEL
              DFCT (ENDN (ARC)) = DFCT (ENDN (ARC)) + DEL
              X (ARC) = X (ARC) -Del
              IF (X (ARC) .LT.0) Write (6180)
              U (ARC) = 0
            END IF
          MAIS
            U (ARC) = U (ARC) + DELUB
            IF (U (ARC) .LT.0) Write (6180)
180 FORMATO ("fluxo limite superior É AGORA <0 ')
          END IF
        Else if (SEL.EQ.3) THEN
C
C CHANGE ARC COST.
C
190 Write (6200)
200 FORMATO ('ENTRADA NO ARC. & Aumento no custo')
          READ (5 *) ARC, DELc
          IF ((ARC.LE.0) .OR. (ARC.GT.NA)) Vá para 190
          IF ((RC (ARC) .GE.0) .E. (RC (ARC) + DELC.LT.0)) THEN
C
C ARC torna-se ativa, SO aumentar o fluxo à capacidade.
C
            DFCT (STARTN (ARC)) = DFCT (STARTN (ARC)) + U (ARC)
            DFCT (ENDN (ARC)) = DFCT (ENDN (ARC)) - U (ARC)
            X (ARC) = U (ARC) + X (ARC)
            U (ARC) = 0
          Else if ((RC (ARC) .LE.0) .E. (RC (ARC) + DELC.GT.0)) THEN
C
C ARC torna-se inativo, SO diminuir o fluxo de zero.
C
            DFCT (STARTN (ARC)) = DFCT (STARTN (ARC)) - X (ARC)
            DFCT (ENDN (ARC)) = DFCT (ENDN (ARC)) + X (ARC)
            U (ARC) = U (ARC) + X (ARC)
            X (ARC) = 0
          END IF
          RC (ARC) = RC (ARC) + DELc
          C (ARC) = C (ARC) + DELc
C
C SE TORNA ARC equilibrado, de seleção para adicionar ARC TO TFSTOU,
TFSTIN, ....
C
          IF ((RC (ARC) .EQ.0) .E. (DELC.NE.0)) THEN
            CHAME ADDTR (ARC)
          END IF
C
```

```
        Else if ((SEL.EQ.4) .OR. (SEL.EQ.6)) THEN
C
C Apagar um arco.
C
          IF (SEL.EQ.6) THEN
            IF (.NOT.ADDARC) Vá para 20
            ADDARC = .FALSE.
            ARC = AARC
          MAIS
210 Write (6220)
220 FORMATO ('ENTRADA ARC NO. PARA ELIMINAÇÃO ")
            READ (5 *) ARC
            IF ((ARC.LE.0) .OR. (ARC.GT.NA)) Vá para 210
            Delarc = .TRUE.
            DARC = ARC
            DU = U (ARC) + X (ARC)
          END IF
C
C Retire ARC a partir da matriz FIN, FOU, NXTIN, NXTOU.
C
          ARC1 = FOU (STARTN (ARC))
          IF (ARC1.EQ.ARC) THEN
            FOU (STARTN (ARC)) = NXTOU (ARC1)
          MAIS
230 ARC2 = NXTOU (ARC1)
            IF (ARC2.EQ.ARC) THEN
              NXTOU (ARC1) = NXTOU (ARC2)
              IR PARA 240
            END IF
            ARC1 = ARC2
            IF (NXTOU (ARC1) .GT.0) ir para 230
          END IF
240 ARC1 = FIN (ENDN (ARC))
          IF (ARC1.EQ.ARC) THEN
            FIN (ENDN (ARC)) = NXTIN (ARC1)
          MAIS
250 ARC2 = NXTIN (ARC1)
            IF (ARC2.EQ.ARC) THEN
              NXTIN (ARC1) = NXTIN (ARC2)
              IR PARA 260
            END IF
            ARC1 = ARC2
            IF (NXTIN (ARC1) .GT.0) ir para 250
          END IF
C
C Retire ARC a partir da matriz TFSTIN, TFSTOU, TNXTIN, TNXTOU.
C
260 ARC1 = TFSTOU (STARTN (ARC))
          IF (ARC1.EQ.0) ir para 262
          IF (ARC1.EQ.ARC) THEN
            TFSTOU (STARTN (ARC)) = TNXTOU (ARC1)
          MAIS
261 ARC2 = TNXTOU (ARC1)
            IF (ARC2.EQ.ARC) THEN
              TNXTOU (ARC1) = TNXTOU (ARC2)
              IR PARA 262
            END IF
            ARC1 = ARC2
            IF (TNXTOU (ARC1) .GT.0) ir para 261
          END IF
262 ARC1 = TFSTIN (ENDN (ARC))
```

```
          IF (ARC1.EQ.0) ir para 264
          IF (ARC1.EQ.ARC) THEN
            TFSTIN (ENDN (ARC)) = TNXTIN (ARC1)
          MAIS
263 ARC2 = TNXTIN (ARC1)
            IF (ARC2.EQ.ARC) THEN
              TNXTIN (ARC1) = TNXTIN (ARC2)
              IR PARA 264
            END IF
            ARC1 = ARC2
            IF (TNXTIN (ARC1) .GT.0) ir para 263
          END IF
264 TNXTOU (ARC) = -1
          TNXTIN (ARC) = -1
C
C REMOVE FLUXO DE ARC DE rede definindo seu fluxo
C e capacidade para 0.
C
          DFCT (STARTN (ARC)) = DFCT (STARTN (ARC)) - X (ARC)
          DFCT (ENDN (ARC)) = DFCT (ENDN (ARC)) + X (ARC)
          X (ARC) = 0
          U (ARC) = 0
        Else if ((SEL.EQ.5) .OR. (SEL.EQ.7)) THEN
          IF (SEL.EQ.7) THEN
            IF (.NOT.DELARC) Vá para 20
            IARC = DARC
            IH = STARTN (IARC)
            IT = ENDN (IARC)
            Delarc = .FALSE.
            IU = DU
          MAIS
270 Write (6280) NA + 1
280 FORMATO ('input Head & CAUDA nós de novo arco', I8)
          READ (5 *) IH, IT
          IF ((IH.LE.0) .OR. (IH.GT.N) .OR. (IT.LE.0) .OR. (IT.GT.N))
Vá para 270
290 Write (6300)
300 FORMATO ('ENTRADA COST & Flow UPPER BD')
          READ (5 *) IC, IU
          IF (IU.LT.0) ir para 290
          ADDARC = .TRUE.
          AARC = NA + 1
          NA NA + 1 =
          C (NA) = IC
          STARTN (NA) = IH
          ENDN (NA) = TI
          IARC = NA
        END IF
C
C determinam os preços duplos a IH E de TI:
C primeiro set o preço pelo IH a zero e então construir THE
C PREÇO em outros nós BY em largura PESQUISA E USANDO
C O FATO DE QUE
C RC (ARC) = C (ARC) - PREÇO (STARTN (ARC)) + PREÇO (ENDN (ARC)).
C DA REDE (dado pelo FOU, NXTOU, FIN, NXTIN) não precisa ser
conectado.
C
        NSCAN = 0
        NLABEL = 1
        LABEL (1) = IH
        PREÇO (IH) = 0
```

```
        FAZER 310 I = 1, N
310 MARK (I) =. FALSE.
        MARK (IH) =. TRUE.
320 IF (NLABEL.GT.NSCAN) THEN
          NSCAN = + 1 NSCAN
          NÓ = LABEL (NSCAN)
          ARC = FOU (nó)
330 IF (ARC.GT.0) THEN
            NODE2 = ENDN (ARC)
            IF (.NOT.MARK (NODE2)) THEN
              MARK (NODE2) =. TRUE.
              PREÇO (NODE2) = RC (ARC) -C (ARC) + PREÇO (nó)
              IF (NODE2.EQ.IT) ir para 370
              NLABEL = + 1 NLABEL
              LABEL (NLABEL) = NODE2
            END IF
            ARC = NXTOU (ARC)
            IR PARA 330
          END IF
          ARC = FIN (nó)
340 IF (ARC.GT.0) THEN
            NODE2 = STARTN (ARC)
            IF (.NOT.MARK (NODE2)) THEN
              MARK (NODE2) =. TRUE.
              PREÇO (NODE2) = C (ARC) -RC (ARC) + PREÇO (nó)
              IF (NODE2.EQ.IT) ir para 370
              NLABEL = + 1 NLABEL
              LABEL (NLABEL) = NODE2
            END IF
            ARC = NXTIN (ARC)
            IR PARA 340
          END IF
          IR PARA 320
        END IF
        PREÇO (IT) = - C (IARC)
C
C COMPUTE custo reduzido do arco NOVO E ATUALIZAÇÃO DE FLUXO e déficit
C, em conformidade.
C
370 RC (IARC) = C (IARC) + PREÇO (IT)
        IF (RC (IARC) .LT.0) THEN
          DFCT (IH) = DFCT (IH) + IU
          DFCT (TI) = DFCT (TI) -IU
          X (IARC) = IU
          U (IARC) = 0
        MAIS
          X (IARC) = 0
          U (IARC) = IU
        END IF
        NXTOU (IARC) = FOU (IH)
        FOU (IH) = IARC
        NXTIN (IARC) = FIN (IT)
        FIN (IT) = IARC
        IF (RC (IARC) .EQ.0) THEN
          TNXTOU (IARC) = TFSTOU (IH)
          TFSTOU (IH) = IARC
          TNXTIN (IARC) = TFSTIN (IT)
          TFSTIN (IT) = IARC
        END IF
      END IF
      IR PARA 20
```

```
      FIM
C
C
      SUBROUTINE ADDTR (ARC)
      Implícita INTEGER (AZ)
C
C ---------------------------------------------- --------------
C
Verifica essa sub-rotina IF Arc é nas matrizes TFSTOU, TNXTOU, - C
FINALIDADE
C TFSTIN, TNXTIN E, se não, acrescenta o ARC para eles.
C
C ---------------------------------------------- --------------
C
C MAXNN = dimensão do ARRAYS NÓ de comprimento
C MAXNA = dimensão do ARRAYS ARC-length
C
      PARÂMETROS (MAXNN = 10000, MAXNA = 70000)
C
      COMUNS / matrizes / STARTN / ARRAYE / ENDN
      COMUM / BLK10 / TFSTOU / BLK11 / TNXTOU / BLK12 / TFSTIN / BLK13
/ TNXTIN
      INTEGER STARTN (MAXNA), ENDN (MAXNA)
      INTEGER TFSTOU (MAXNN), TNXTOU (MAXNA), TFSTIN (MAXNN), TNXTIN
(MAXNA)
C
      NÓ = STARTN (ARC)
      ARC1 = TFSTOU (nó)
10 IF (ARC1.GT.0) THEN
        IF (ARC1.EQ.ARC) Vá para 20
        ARC1 = TNXTOU (ARC1)
        IR PARA 10
      END IF
      TNXTOU (ARC) = TFSTOU (nó)
      TFSTOU (nó) = ARC
20 NODE = ENDN (ARC)
      ARC1 = TFSTIN (nó)
30 IF (ARC1.GT.0) THEN
        IF (ARC1.EQ.ARC) RETURN
        ARC1 = TNXTIN (ARC1)
        IR PARA 30
      END IF
      TNXTIN (ARC) = TFSTIN (nó)
      TFSTIN (nó) = ARC
      RETURN
      FIM
C



CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C amostra de entrada ARQUIVO RELAX4.INP

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      20 80
       1 10 38 115
       1 11 17 120
       8 18 82 115
       8 1 56 197
```

 8 11 48 126
10 8 69 115
10 115 16 39
10 15 78 179
10 14 82 131
10 12 100 146
9 10 74 101
 2 15 27 166
 2 1 75 195
 2 16 57 129
 2 11 38 181
 2 8 67 185
 2 14 39 162
12 18 1 166
12 14 99 126
12 10 85 155
12 2 157 15
12 11 59 148
15 12 36 166
15 16 18 166
15 13 100 114
4 15 1 130
15 7 96 100
15 155 18 30
 3 6 71 100
 3 5 12 153
 6 7 68 100
 6 17 89 100
 6 12 13 163
 7 9 93 100
 7 13 26 176
 7 16 24 148
 7 20 45 169
 9 20 2 100
 9 16 58 100
 9 77 1 180
 9 19 13 145
 9 6 63 104
 4 13 81 100
 4 61 1 160
 4 12 4 133
 4 3 23 136
 4 14 77 188
13 19 95 100
13 100 17 43
13 11 82 104
 5 11 31 119
 5 103 8 18
 5 7 84 173
 5 20 95 119
 5 6 27 175
11 14 62 119
11 194 15 15
11 10 83 185
11 4 52 184
11 7 94 165
14 19 87 119
14 16 92 119
14 128 12 52
16 14 80 145
16 12 89 148

```
 16 17 83 173
 17 189 19 33
 17 4 20 132
 17 14 15 136
 18 17 92 114
 18 2 3 157
 19 16 77 146
 19 166 14 48
 20 9 69 160
 20 13 9 154
 20 18 32 157
 20 165 17 80
 20 16 76 185
 20 8 54 117
 20 7 3 131
115
166
  5
 95
119
  0
  0
  0
  0
  0
  0
  0
  0
  0
  0
-183
 -73
-161
 -80
  -3
```