Este exemplar inclui as modificações sugeridas pelo banca examinadora.

26/6/90

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELETRICA

ALGORITMO DE PONTOS INTERIORES EM UM PROCESSO DE "BRANCH AND BOUND" APLICADO A UM SISTEMA DE MECANIZAÇAO AGRICOLA

Dissertação apresentada para a obtenção do título de mestre em Engenharia Elétrica.

Autor : Paulo José Fogaça Martins

Orientador : Christiano Lyra Filho

Campinas 1990

Dedico este trabalho ao mestre Allan Kardec, que graças ao seu imortal trabalho, me deu a fé necessária para chegar até aqui...

Este trabalho foi financiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES

Paulo José Fagaça Martins

Algoritmo.

AGRADECIMENTOS...

... ao amigo, Prof. Dr. Christiano Lyra Filho pelo estímulo dado para tornar isto possível,

... ao amigo, Eng. Angelo Domingos Banchi pela grande ajuda, principalmente nos assuntos agrícolas,

... ao amigo, Aurélio R. Leite Oliveira pelas valiosas achegas,

... à Cooperativa de Produtores de Cana, Açucar e Álcool do Estado de São Paulo - COPERSUCAR , por fornecer dados precisos.

1. INTRODUÇÃO

O objetivo deste trabalho é estudar o desempenho de um algoritmo de ponto interior num processo de "branch and bound" para a otimização de um sistema de planejamento da mecanização agrícola.

Nestes últimos anos muitos trabalhos ([1] - [12]) tem sido escritos sobre algoritmos de ponto interior e sua aplicação à programação linear. Uma das razões dessas pesquisas é a grande importância dos problemas de programação linear. Outra razão é a grande vantagem que seria obtida através da redução do tempo gasto na otimização de alguns problemas reais de muito grande porte.

Esse assunto vem despertando interesse a partir de fevereiro de 1979 quando o matemático soviético L.G. Khachiyan anunciou na Doklady Akademii Nauk URSS um novo algoritmo para resolver problemas de programação linear. O fato chegou a ser publicado na imprensa não científica como The New York Times e The Manchester Guardian. Tratava-se do método dos elipsóides, que trabalha com pontos interiores no espaço das restrições, ao qual se apregoava ser mais rápido do que o simplex. Posteriormente foi observado que o método dos elipsóides possuia uma complexidade melhor do que o simplex porém esbarrava em problemas na implementação computacional relacionados à perda de precisão numérica.

A pesquisa porém prosseguiu intensa e muitos trabalhos surgiram no cenário. Em 1984 Karmarkar [2] publicou um artigo onde descrevia um novo algoritmo de ponto interior que possuia complexidade polinomial e, segundo o autor , tinha desempenho computacional melhor do que o simplex. Tal publicação teve grande influência na motivação de novas

pesquisas.

Trabalhos recentes [1, 3] mostram que um algoritmo de ponto interior pode ser muito rápido na resolução de problemas grandes e esparsos. Esse fato, e algumas propriedades que veremos adiante, sugerem uma averiguação da viabilidade de um algoritmo de ponto interior como agente ativo na resolução de subproblemas numa árvore de um processo "branch and bound". No presente trabalho estuda-se a utilização de um algoritmo branch and bound, acoplado ao método dual afim de ponto interior, na solução de um problema de otimização de um sistema de mecanização agrícola.

Em empresas agrícolas do setor de cana de açucar existem uma grande quantidade de elementos envolvidos nos trabalhos de uma safra. Alguns desses elementos são : planejamento de operações agrícolas, elaboração do escalonamento do uso e manutenção de máquinas e implementos, execução de cronogramas, etc.

Tendo como intenção uma visão global desses elementos formulou-se um modelo de programação linear mista, ou seja, com variáveis reais e inteiras, onde a meta é minimizar as perdas globais de mecanização. As variáveis inteiras serão tratadas através de um algoritmo do tipo "branch and bound".

2. FORMULAÇÃO DO PROBLEMA

Tratores e implementos são as principais máquinas envolvidas nos trabalhos agrícolas de empresas de cana de açucar. Essas máquinas devem realizar operações, observando cronogramas estabelecidos para a safra.

A partir desses elementos formulou-se um modelo de programação linear mista cuja finalidade é encontrar o número de horas a serem trabalhadas por conjuntos agrícolas nas diversas operações, de acordo com o cronograma. Além disso, o modelo encontra a quantidade ótima de tratores e implementos a serem utilizados numa safra.

Essas últimas variáveis, inteiras (pois não teria sentido falar em frações de tratores), são elementos importantes nas decisões de compras, vendas ou trocas de máquinas e implementos, os quais são de elevado valor financeiro.

O modelo vem sendo desenvolvidos há vários anos, sobretudo por Banchi e outros [15-17], e passou por muitas alterações. Atualmente é utilizado com as variáveis inteiras consideradas como reais, devido principalmente ao grande porte dos problemas práticos, o que dificulta a consideração explícita de variáveis inteiras.

Matematicamente, o problema pode ser estabelecido da forma a seguir, onde (2.1) é a função objetivo representando o gasto completo do planejamento para o período em questão, (2.2) indica qual a área a ser trabalhada pelos conjuntos, (2.3) indica as relações de precedência a serem respeitadas, (2.4) indica a disponibilidade de cada tipo de trator (observe que essa disponibilidade é uma variável inteira) (2.5) é uma restrição do tipo (2.4) aplicada aos implementos,

(2.6) indica a não negatividade das quantidades de horas operadas por cada conjunto, (2.7) e (2.8) indicam que as quantidades de tratores e implementos, respectivamente, são quantidades inteiras não negativas e finalmente (2.9) indica a não negatividade das folgas. Assim tem-se :

Minimizar

$$\sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{o=1}^{O}\sum_{p=1}^{P} k(t,i,o,p).c_{tiop}.h_{tiop} + \sum_{t=1}^{T} a_t x_t + \sum_{i=1}^{I} b_i y_i \qquad (2.1)$$

sujeito à :

$$\sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{p=1}^{P} k(t,i,o,p).r_{tiop}.h_{tiop} = s(o) \quad ; \quad o = 1 \,..\, O \qquad (2.2)$$

$$\sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{p=1}^{\bar{P}} k(t,i,o,p).r_{tiop}.h_{tiop} + f(o).z_{o\bar{p}} = \qquad (2.3)$$

$$\sum_{i=1}^{T}\sum_{i=1}^{I}\sum_{a=1}^{g(o)}\sum_{p=1}^{\bar{P}} k(t,i,q[o,a],p).\lambda(o).r_{ti[q(o,a)]p}.h_{ti[q(o,a)]p}$$

$$o = 1 \,..\, O$$

$$\bar{p} = 1 \,..\, P$$

$$g(o) \neq 0$$

$$\sum_{i=1}^{I}\sum_{o=1}^{O} k(t,i,o,p).h_{tiop} \leq NH(p).x_t \quad ; t = 1 \,..\, T \qquad (2.4)$$

$$; p = 1 \,..\, P$$

$$\sum_{t=1}^{T} \sum_{o=1}^{O} k(t,i,o,p).h_{tiop} \leq NH(p).y_i \quad ; i = 1 .. I \quad (2.5)$$
$$; p = 1 .. P$$

$$k(t,i,o,p).h_{tiop} \geq 0 \quad ; t = 1 .. T \quad (2.6)$$
$$; i = 1 .. I$$
$$; o = 1 .. O$$
$$; p = 1 .. P$$

$$x_t \geq 0 \text{ e inteiro} \quad ; t = 1 .. T \quad (2.7)$$

$$y_i \geq 0 \text{ e inteiro} \quad ; i = 1 .. I \quad (2.8)$$

$$z_{ox} \geq 0 \quad ; o = 1 .. O \quad (2.9)$$
$$; g(o) \neq 0$$
$$; x = 1 .. P$$

Onde :

T = quantidade de modelos distintos de tratores

I = quantidade de modelos distintos de implementos

O = quantidade de operações agrícolas existentes

P = quantidade de períodos agrícolas existentes

Conjunto (t,i,o,p) = trator tipo(t) acoplado ao implemento tipo(i) e realizando a operação(o) no período(p)

c_{tiop} = custo horário do conjunto (t,i,o,p)

h_{tiop} = horas trabalhadas pelo conjunto (t,i,o,p)

a_t = custo unitário do trator do tipo(t)

x_t = quantidade de tratores do tipo(t)

b_i = custo unitário do implemento do tipo(i)

sugeridas pela banca examinadora.
26/6/90

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA

ALGORITMO DE PONTOS INTERIORES EM UM PROCESSO DE "BRANCH AND BOUND" APLICADO A UM SISTEMA DE MECANIZAÇÃO AGRICOLA

Dissertação apresentada para a obtenção do título de mestre em Engenharia Elétrica.

Autor : Paulo José Fogaça Martins

Orientador : Christiano Lyra Filho

Campinas 1990

Dedico este trabalho ao mestre Allan Kardec,
que graças ao seu imortal trabalho, me deu
a fé necessária para chegar até aqui...

Este trabalho foi financiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES

y_i = quantidade de implementos do tipo(i)

r_{tiop} = rendimento em ha/h do conjunto (t,i,o,p)

s(o) = área a ser trabalhada a operação (o)

$$f(o) = \begin{cases} 0 \text{ se a operação (o) tem precedencias concomitantes} \\ 1 \text{ caso contrário} \end{cases}$$

z_{ox} = variável de folga de restrição de precedência

g(o) = quantidade de operações precedentes de (o)

q(o,a) = a-ésima precedente de (o)

$$\lambda(o) = \frac{\text{Área da operação (o)}}{\sum_{a=1}^{g(o)} \text{área da operação } q(o,a)}$$

$$k(t,i,o,p) = \begin{cases} 1 \text{ se existe o conjunto (t,i,o,p)} \\ 0 \text{ caso contrário} \end{cases}$$

NH(p) = número de horas disponíveis para trabalho agrícola no período (p)

Os rendimentos dos conjuntos agrícolas são normalmente levantados através de medições feitas na plantação. Um procedimento típico feito por técnicos agrícolas é cronometrar o tempo gasto por um determinado conjunto para , por exemplo , efetuar uma aração. Esses dados são depois multiplicados por fatores de segurança empíricos fornecendo o correspondente coeficiente da matriz tecnológica.

O número de horas disponíveis para trabalho agrícola num dado período é obtido através de dados pluviométricos, os quais fornecem o número de dias adequados para a operação de máquinas agrícolas. Esse número de dias é multiplicado pela jornada de trabalho em horas, fornecendo o coeficiente da matriz tecnológica.

Os custos horários de cada conjunto agrícola são calculados de maneira a incluir gastos com combustível, óleo e peças. Também são considerados gastos com mão de obra e desvalorização das máquinas e implementos.

O modelo que vem sendo utilizado [15] não considera as quantidades de tratores e implementos como variáveis, sendo utilizadas as quantidades disponíveis pela empresa. Uma das motivações do nosso trabalho é justamente tentar descobrir, através de uma análise global do planejanento, estas quantidades e permitir que as empresas mantenham um parque de equipamentos mais realista.

Um aspecto importante do modelo são as relações de precedência , representadas pelas equações do tipo (2.3). Em empresas agrícolas do setor de cana de açucar há um grande número de operações realizadas com ligação direta ou indireta com a área cultivada. Na realização

dessas operações existem seqüências de atividades, ou seja, o término de uma (ou mais) permite o início de uma outra (ou outras). Faremos aqui uma descrição dos vários tipos de relações de precedência entre as operações agrícolas. A classificação que segue foi elaborada em [17].

Antes de iniciar a descrição das relações de precedência é necessário definir o conceito de período agrícola.

Período Agrícola é um intervalo de tempo onde o número de operações agrícolas realizadas não muda, ou seja , durante esse intervalo não ocorre a execução de uma operação diferente das que iniciaram com o intervalo, assim como não ocorre o término de alguma operação antes do término do intervalo.

Na discussão que segue consideramos A, B e C tres operações agrícolas (arbitrárias) e $\bar{P}$ ($\bar{P} \leq P$) um dado período agrícola.

3.1 PRECEDÊNCIA SIMPLES

Sejam A e B as operações sujeitas a essa relação. Isso significará que as áreas totais de ambas serão iguais e a operação A nos diversos períodos deve ser feita antes de B, não havendo nenhuma outra imposição.

Exemplo :

A = cobertura de sulco(650 ha)

B = limpeza de terreno(650 ha)

esquematicamente : A ⇒ B (significa que A é realizada antes de B)

Sejam $\bar{p}$ ($\bar{p} \leq P$) um período agrícola, T a quantidade de modelos distintos de tratores e I a quantidade de modelos distintos de implementos. A restrição de precedência simples terá a seguinte forma :

$$\sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{p=1}^{\bar{p}} k(t,i,B,p) \cdot r_{tiBp} \cdot h_{tiBp} \leq \sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{p=1}^{\bar{p}} k(t,i,A,p) \cdot r_{tiAp} \cdot h_{tiAp}$$

onde lembramos que :

$$k(i,j,k,l) = \begin{cases} 1 \text{ se existe o conjunto } (i,j,k,l) \\ 0 \text{ caso contrário} \end{cases}$$

r_{ijkl} = rendimento em ha/h do trator i acoplado ao implemento j executando a operação agrícola k no período l

h_{ijkl} = número de horas trabalhadas pelo trator i acoplado ao implemento j executando a operação agrícola k no período l

Poderíamos visualizar a relação de precedência simples com o uso da seguinte tabela (fig 3.1), representando algumas operações agrícolas (linhas) e respectivos períodos de execução(colunas) :

	1	2	3	4
operação A	A1	A2	A3	
operação B	B1	B2	B3	B4

figura 3.1 : tabela de períodos e operações

suponha que temos : T = 2 $\qquad$ $k(i,j,k,l) = 1$ ($\forall$ i, j, k, l)

I = 2

A = operação 1

B = operação 2

Neste caso teremos :

para o período 1 : área(B_1) ≤ área(A_1)

$$r_{1121}\cdot h_{1121} + r_{1221}\cdot h_{1221} + r_{2121}\cdot h_{2121} + r_{2221}\cdot h_{2221} \leq$$
$$r_{1111}\cdot h_{1111} + r_{1211}\cdot h_{1211} + r_{2111}\cdot h_{2111} + r_{2211}\cdot h_{2211}$$

para o período 2 : área(B_1) + área(B_2) ≤ área(A_1) + área(A_2)

$$r_{1121}\cdot h_{1121} + r_{1221}\cdot h_{1221} + r_{2121}\cdot h_{2121} + r_{2221}\cdot h_{2221} +$$
$$r_{1122}\cdot h_{1122} + r_{1222}\cdot h_{1222} + r_{2122}\cdot h_{2122} + r_{2222}\cdot h_{2222} \leq$$
$$r_{1111}\cdot h_{1111} + r_{1211}\cdot h_{1211} + r_{2111}\cdot h_{2111} + r_{2211}\cdot h_{2211} +$$
$$r_{1112}\cdot h_{1112} + r_{1212}\cdot h_{1212} + r_{2112}\cdot h_{2112} + r_{2212}\cdot h_{2212}$$

para o período 3 : área(B_1) + área(B_2) + área(B_3) ≤
área(A_1) + área(A_2) + área(A_3)

$$r_{1121}\cdot h_{1121} + r_{1221}\cdot h_{1221} + r_{2121}\cdot h_{2121} + r_{2221}\cdot h_{2221} +$$
$$r_{1122}\cdot h_{1122} + r_{1222}\cdot h_{1222} + r_{2122}\cdot h_{2122} + r_{2222}\cdot h_{2222} +$$
$$r_{1123}\cdot h_{1123} + r_{1223}\cdot h_{1223} + r_{2123}\cdot h_{2123} + r_{2223}\cdot h_{2223} \leq$$
$$r_{1111}\cdot h_{1111} + r_{1211}\cdot h_{1211} + r_{2111}\cdot h_{2111} + r_{2211}\cdot h_{2211} +$$
$$r_{1112}\cdot h_{1112} + r_{1212}\cdot h_{1212} + r_{2112}\cdot h_{2112} + r_{2212}\cdot h_{2212} +$$
$$r_{1113}\cdot h_{1113} + r_{1213}\cdot h_{1213} + r_{2113}\cdot h_{2113} + r_{2213}\cdot h_{2213}$$

A figura 3.2, a seguir, ilustra uma possibilidade para as áreas no período 3, o quadro representa uma área onde são realizadas as operações A e B.

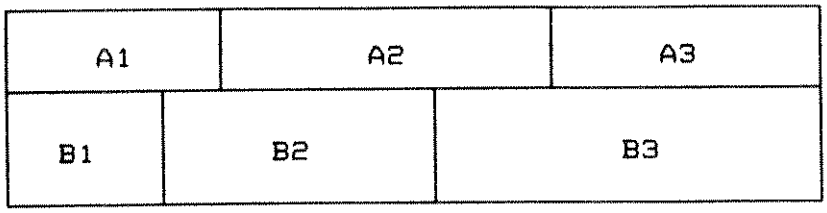


figura 3.2 : áreas trabalhadas

3.2 PRECEDÊNCIA SIMPLES CONCOMITANTE

As áreas de A e B ainda são iguais, porém ambas devem ser executadas num mesmo periodo agrícola, ou seja :

$$\sum_{t=1}^{T}\sum_{i=1}^{I} k(t,i,A,\bar{p}).r_{tiA\bar{p}}.h_{tiA\bar{p}} = \sum_{t=1}^{T}\sum_{i=1}^{I} k(t,i,B,\bar{p}).r_{tiB\bar{p}}.h_{tiB\bar{p}}$$

Exemplo :

A = abertura de sulco

B = distribuição das mudas

C = cobertura de sulco

Por motivos técnicos, não deve haver muito tempo entre a abertura do sulco e a distribuição das mudas , o mesmo ocorrendo entre distribuição das mudas e a cobertura do sulco.

3.3 PRECEDÊNCIA PROPORCIONAL

Temos agora discrepância de área entre as operações relacionadas, porém sem exigencia quanto aos periodos.

Sejam A e B duas operações agrícolas que sofrem essa restrição. Se

utilizássemos a modelagem da precedência simples (relação 1) teríamos no último período :

Área total de A $\geq$ Área total de B

(em todos os períodos) (em todos os períodos)

No entanto, há operações em que a área de A pode ser menor do que a área de B. Assim o problema se tornaria infactível se usássemos a relação de precedência simples, devido à exigência de se trabalhar toda a área de B.

A solução adotada para representar esta situação foi definir a relação de precedência proporcional, que consiste em multiplicar os rendimentos dos conjuntos de A pela razão entre as áreas de B e A.

Exemplo :

A = subsolagem(10 ha)

B = gradeação média(100 ha)

temos : $\lambda_{BA} = 100 / 10 = 10$

$$\sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{p=1}^{\bar{P}} k(t,i,B,p).r_{tiBp}.h_{tiBp} \leq$$

$$\sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{p=1}^{\bar{P}} k(t,i,A,p).\lambda_{BA}.r_{tiAp}.h_{tiAp}$$

O diagrama(fig 3.3) a seguir ilustra o conceito de precedência proporcional.

figura 3.3 : áreas desiguais

3.4 PRECEDÊNCIA PROPORCIONAL CONCOMITANTE

Esta relação é semelhante à anterior , porém as operações relacionadas devem ser executadas concomitantemente(no mesmo período agrícola). Isso leva imediatamente à :

$$\sum_{t=1}^{T}\sum_{i=1}^{I} k(t,i,B,\bar{p}).r_{tiB\bar{p}}.h_{tiB\bar{p}} = \sum_{t=1}^{T}\sum_{i=1}^{I} k(t,i,A,\bar{p}).\lambda_{BA}.r_{tiA\bar{p}}.h_{tiA\bar{p}}$$

Exemplo :

A = carregamento de cana muda

B = sulcação e adubação

3.5 PRECEDÊNCIA COMPOSTA

A palavra composta aqui significa que duas operações liberam área para a realização de uma terceira.

Esquematicamente teríamos :

A

→ C (significa : A ou B devem ser feitas antes de C)

B

Então a restrição será da forma :

$$\sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{p=1}^{\bar{P}} k(t,i,C,p).r_{tiCp}.h_{tiCp} \leq$$

$$\sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{p=1}^{\bar{P}} k(t,i,A,p).r_{tiAp}.h_{tiAp} + \sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{p=1}^{\bar{P}} k(t,i,B,p).r_{tiBp}.h_{tiBp}$$

Exemplo :

A = aração

B = eliminação mecânica de soqueira

C = gradagem pesada

D = gradagem média

Esquematicamente :

A

B → D (significa que A ou B ou C devem vir antes de D)

C

3.6 PRECEDÊNCIA COMPOSTA PROPORCIONAL

Temos agora discrepâncias de áreas entre as operações relacionadas. Uma solução possível é multiplicar os rendimentos das operações precedentes pelas razões da operação seguinte em relação à

soma das áreas das operações precedentes, ou seja :

A

$\Rightarrow$ C $\qquad \lambda_{CA} = \lambda_{CB} = \text{área(C)}/(\text{área(A)}+\text{área(B)})$

B

Então a restrição será da forma :

$$\sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{p=1}^{\bar{P}} k(t,i,C,p).r_{tiCp}.h_{tiCp} \leq$$

$$\sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{p=1}^{\bar{P}} k(t,i,A,p).\lambda_{CA}.r_{tiAp}.h_{tiAp} +$$

$$\sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{p=1}^{\bar{P}} k(t,i,B,p).\lambda_{CB}.r_{tiBp}.h_{tiBp}$$

Exemplo :

A = subsolagem de bordadura com haste curva

B = subsolagem de bordadura com haste reta

C = sulcação e adubação

4. OBTENÇÃO DE SOLUÇÕES ÓTIMAS

Neste capítulo estudaremos a metodologia utilizada para obtermos soluções ótimas para o problema de planejamento agrícola. Na discussão a seguir assumiremos que estamos tratando um problema de *minimização*.

4.1 O MÉTODO "BRANCH AND BOUND"

Recordando a idéia de enumeração note que se $\underline{x}$ é um n-vetor solução cujas componentes estão restritas a valores inteiros e também limitadas a um n-vetor $\underline{u}$, ou seja , $\underline{x} \leq \underline{u}$, temos $\prod_{i=1}^{n} (u_i+1)$ soluções distintas.

Uma idéia simplista para se resolver um problema com variáveis mistas seria testar cada uma das possibilidades de valores inteiros e escolher a que for factível e ao mesmo tempo forneça o menor custo. Isso seria feito atribuindo-se valores inteiros às variáveis inteiras e resolvendo-se o problema linear resultante. Certamente esbarraríamos no problema de tempo pois normalmente teríamos que testar um número muito grande de combinações. O método "branch and bound" é um procedimento de enumeração que testa *implicitamente* um grande número de combinações de valores das variáveis inteiras permitindo tempos de processamento menores. Essa enumeração implícita é conseguida com o uso de certas propriedades, conhecidas como testes de sondagem, como veremos adiante.

Mencionaremos apenas as idéias básicas do funcionamento de um algoritmo "branch and bound". Para maiores detalhes sugere-se

consultar Salkin (referência [25]).

4.1.1 PRINCIPAIS COMPONENTES

a) Lista Mestra :

É uma lista onde cada elemento está associado a um problema de programação linear. Contém limites inferiores e limites superiores para as variáveis, um limitante inferior para o valor da função objetivo deste problema (corresponde ao valor da funcão objetivo na solução ótima de um problema relaxado), bem como uma solução inicial factível. Os demais dados do PL (matriz A, vetor b e vetor c) são comuns à todos os problemas analizados no processo de "branch and bound".

Esses problemas são criados durante a execução do método e são retirados da lista à medida que são resolvidos. Cada problema pode provocar a derivação de outros dois que diferem apenas nos limites inferiores e superiores de uma variável. O processo é ilustrado na fig. 4.1 :

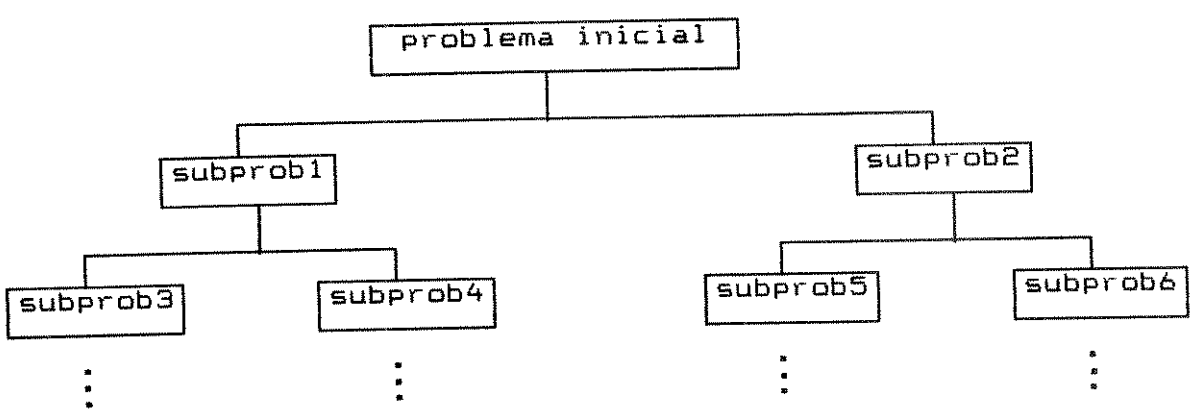


Figura 4.1 : Árvore binária do processo de "branch and bound"

Note que o valor ótimo da solução de um problema relaxado é um limitante inferior para o valor ótimo da solução de qualquer problema derivado deste através de modificações nos limites das variáveis.

b) Solução incumbente :

Consiste na melhor solução encontrada até o momento, satisfazendo todas as restrições do problema (inclusive integralidade). É formada pelos valores das variáveis e o valor da função objetivo.

4.1.2 FUNCIONAMENTO DO ALGORITMO "BRANCH AND BOUND"

i) inicialização :

Obtenha um limitante superior Z^* para o valor ótimo da função objetivo - se nenhuma solução factível é disponível atribuímos um valor muito grande à Z^*.

Obtenha uma solução inicial (relaxada) para o problema a ser resolvido. Esse ponto juntamente com os limites inferiores e superiores das variáveis formam o primeiro problema colocado na lista mestra.

ii) iteração principal :

Agora surgem os mencionados testes de sondagem :

- se a lista mestra está vazia , pare , a solução incumbente é ótima (pode não existir incumbente nessa ocasião).
- se a lista mestra não está vazia , retire um problema da mesma e resolva-o. Isso equivale a alterar um limite de uma variável e resolver o problema resultante desta alteração.

- se o problema resolvido é infactível consideramos o mesmo sondado, volte a (ii).

- se o problema resolvido é factível, seja Z_o o valor da função objetivo para o problema. Então ,

  - se $Z_o \geq Z^*$ também consideramos o problema sondado pois nenhuma solução inteira originada desse problema poderá ser melhor que a atual incumbente, volte a (ii).

  - se $Z_o < Z^*$ e as variáveis inteiras já assumem valores inteiros esta solução passa a ser a incumbente, volte a (ii).

  - se $Z_o < Z^*$ e algumas variáveis inteiras assumem valores não inteiros escolhemos uma dessas variáveis inteiras. Seja i essa variável e $\bar{x}_i$ o seu valor ,temos :

$$k < \bar{x}_i < k + 1, \; k \text{ inteiro}$$

  crie (coloque na lista mestra) dois problemas , que diferem do corrente nas seguintes restrições :

$$k + 1 \leq x_i \leq u_i$$

$$l_i \leq x_i \leq k$$

  onde l_i e u_i são os limitantes de x_i no problema corrente volte a (ii)

Observações :

1. Posteriormente (ítem 5.3) modificaremos a estrutura da lista mestra, eliminando a necessidade de incluir na mesma uma solução inicial factível para o problema associado.

2. Existem táticas para escolher um problema da lista ([13] e

[14]), visando reduzir o número de problemas resolvidos. No nosso caso, escolhemos o problema que possui o menor limitante inferior do valor da função objetivo.

3. Também existem algumas táticas para escolher a variável visando reduzir a quantidade de subproblemas resolvidos ([13] e [14]). No nosso caso, escolhemos a variável que assume o valor mais fracionário, ou seja, a variável x_i onde i é determinado por :

i tal que $f_i = \max \{ f_k : f_k =$ parte fracionária de $x_k \}$

4.2 O MÉTODO DUAL AFIM

Nesta secção focalizaremos a atenção numa versão de algoritmo de pontos interiores conhecida como Dual Afim. A razão desse nome reside no fato desta versão trabalhar com o problema dual do problema a ser resolvido e de se aplicar uma transformação afim às variáveis de folga.

A característica que torna o dual afim extremamente atraente para um processo de "branch and bound" é expressa no teorema a seguir :

> "Seja o problema (P), minimize $c^t.x$ sujeito à $A.x = b$ $x \geq 0$, e seu dual (D), maximize $b^t.w$ sujeito à $A^t.w \leq c$. Então, se w é interior a (D), continua interior se alterarmos algum componente de b."

A demonstração é imediata pois o interior da região definida por (D) não depende de b.

Faremos inicialmente uma análise da versão de algoritmo de ponto interior utilizado. A partir deste algoritmo, faremos as

justificativas do processo de funcionamento do "branch and bound". Finalmente passaremos à aplicação ao modelo de planejamento de mecanização agrícola e aos resultados dos testes e comparações feitos.

Outras variantes de métodos de ponto interior e maiores informações são encontradas no trabalho de Oliveira [6].

4.2.1 ALGORITMO DUAL AFIM PARA VARIAVEIS LIVRES

4.2.1.1 DESCRIÇÃO BASICA

O algoritmo a seguir foi proposto por Adler e outros[1]. Após a discussão do algoritmo básico, apresentaremos no próximo item as adaptações necessárias para trabalharmos adequadamente com variáveis canalizadas.

Sejam os pares de problemas :

$$(P) \quad \min \; c^t.x$$
$$\text{s.a. } A.x = b$$
$$x \geq 0$$

$$(D) \quad \max \; b^t.v$$
$$\text{s.a. } A^t.v \leq c$$
$$v \text{ irrestrito}$$

Onde, b e v são vetores de dimensão m, c e x são vetores de dimensão n e A uma matriz de rank completo e dimensões mxn com $m \leq n$.

O algoritmo dual afim para variáveis livres pode ser resumido na seqüência de passos a seguir :

início

Sejam v^0 e γ dados tais que $A^t.v^0 < c$ e $0 < \gamma < 1$.

Faça k = 0

enquanto "critério de parada não satisfeito" faça

início

$$y^k = c - A^t.v^k \qquad \text{(DA.1)}$$

$$D^k = \text{diag}(1/y_1^k , \ldots , 1/y_n^k) \qquad \text{(DA.2)}$$

$$d_v = (A.(D^k)^2.A^t)^{-1}.b \qquad \text{(DA.3)}$$

$$d_y = -A^t.d_v \qquad \text{(DA.4)}$$

$$x^k = -(D^k)^2.d_y \qquad \text{; x variáveis primais} \qquad \text{(DA.5)}$$

$$\alpha = \gamma \cdot \min_{i = 1, \ldots, n} \left\{ -y_i^k / (d_y)_i : (d_y)_i < 0 \right\} \qquad \text{(DA.6)}$$

obs.: se $d_y \geq 0$, o dual é ilimitado

$$v^{k+1} = v^k + \alpha.d_v \qquad \text{(DA.7)}$$

faça k = k + 1 (DA.8)

fim

4.2.1.2 SOLUÇÃO DO PROBLEMA PRIMAL (ETAPA DA.5)

A fórmula da etapa (DA.5) do método dual afim para variáveis livres fornece a solução do problema primal e sua obtenção pode ser verificada através das idéias a seguir.

Sejam os problemas (P) e (D) do ítem anterior. Sabemos que se $\bar{x}$ e $\bar{v}$ são soluções ótimas de (P) e (D), respectivamente, então valem as seguintes condições de otimalidade :

Condições de Otimalidade :

i) $A.\bar{x} = b \ , \ \bar{x} \geq 0$

ii) $A^t.\bar{v} \leq c$

iii) $\bar{x}.(c - A^t.\bar{v}) = 0$ ou $X.(c - A^t.\bar{v}) = 0$ onde

$X = diag(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$

Dado um ponto v que satisfaz (ii) determinemos o correspondente ponto x que satisfaz (i) de modo que x e v procurem satisfazer (iii). Isso é traduzido no seguinte problema quadrático, se $c - A^t.v = y$:

$$\min_x \ (X.y)^t.(X.y) \quad \text{s.a. } A.x = b \qquad (PQ1)$$

note que $(X.y)^t.(X.y) = y^t.X^2.y = \sum_{i=1}^{n} (y_i.x_i)^2 = x^t.D^{-2}x$

onde $D = diag(1/y_1, \ \ldots \ , 1/y_n)$, portanto (PQ1) se torna :

$$\min \quad x^t.D^{-2}.x \quad \text{s.a.} \quad A.x = b \qquad (PQ2)$$

A solução de (PQ2) é então obtida via multiplicadores de Lagrange :

$$L(x,\lambda) = x^t.D^{-2}.x + \lambda^t.(A.x - b)$$

$$\frac{\partial L(x,\lambda)}{\partial x} = -2.D^{-2}.x + A^t.\lambda = 0 \qquad (4.1)$$

$$\frac{\partial L(x,\lambda)}{\partial \lambda} = A.x - b = 0 \qquad (4.2)$$

de (4.1) temos :

$$D^{-2}.x = (1/2).A^t.\lambda \quad \Rightarrow \quad x = (1/2).D^2.A^t.\lambda \qquad (4.3)$$

substituindo em (4.2) temos :

$$(1/2).A.D^2.A^t.\lambda - b = 0 \quad \Rightarrow \quad A.D^2.A^t.\lambda = 2.b$$

$$\lambda = 2.(A.D^2.A^t)^{-1}.b$$

substituindo em (4.3) temos $x = D^2.A^t.(A.D^2.A^t)^{-1}.b$ (4.4)

Se o valor da função objetivo de (PQ2) é suficientemente pequeno o ponto x resolve (P) e o ponto v resolve (D), caso contrário moveremos v numa direção d_v tal que $v + \alpha.d_v$ (α escalar) continue

interior à (D) e ainda aumenta função objetivo de (D). Esse movimento de v acarreta um movimento $\alpha.d_y$ do ponto y pois,

$$y + \alpha.d_y = c - A^t.(v + \alpha.d_v)$$

$$y + \alpha.d_y = c - A^t.v - A^t.\alpha.d_v$$

$$d_y = -A^t.d_v \qquad (DA.4)$$

O valor de α deve ser escolhido de modo que $y + \alpha.d_y > 0$ (DA.6).

Como o dual (D) é de maximização, $v + \alpha.d_v$ deve produzir um aumento no valor da função objetivo de (D), ou seja :

$$b^t.v + \alpha.b^t.d_v > b^t.v \rightarrow b^t.d_v > 0$$

De (4.4), a função objetivo de (PQ2) fica,

$$x^t.D^{-2}.x = b^t.[D^2.A^t.(A.D^2.A^t)^{-1}]^t.D^{-2}.D^2.A^t.(A.D^2.A^t)^{-1}.b$$
$$x^t.D^{-2}.x = b^t.(A.D^2.A^t)^{-1}.A.D^2.D^{-2}.D^2.A^t.(A.D^2.A^t)^{-1}.b$$
$$x^t.D^{-2}.x = b^t.(A.D^2.A^t)^{-1}.b > 0$$

Portanto escolhemos $d_v = (A.D^2.A^t)^{-1}.b$ e note que dai teremos de (4.4) $x = D^2.A^t.(A.D^2.A^t)^{-1}.b = D^2.A^t.d_v = -D^2.d_y$ (DA.5). Repare que a discussão acima descreve totalmente o método dual afim.

4.2.2 ALGORITMO DUAL AFIM COM VARIÁVEIS CANALIZADAS

Vamos agora considerar a especialização do algoritmo anterior para o caso de variáveis canalizadas. As manipulações que seguem assumem que todas as variáveis são canalizadas. Entretanto, é fácil

acomodar a formulação para o caso onde só algumas variáveis são canalizadas, evitando que sejam realizados cálculos com limites superiores grandes. Veremos adiante como isso é feito.

Seja o problema primal :

$$(P1) : \min \quad c^t x$$

$$\text{s.a.} \quad Ax = b$$

$$x \leq u$$

$$x \geq 0$$

onde :

A : matriz m x n

x : vetor n x 1

b : vetor m x 1

c : vetor n x 1

u : vetor n x 1

Obs.: se $x_i \geq l > 0$ fazemos uma transformação de variáveis para colocar o problema na forma anterior.

Introduzindo variáveis de folga, podemos reescrever (P1) como :

$$(P2) : \min \quad c^t x$$

$$\text{s.a.} \quad Ax = b$$

$$x + y = u$$

$$x , y \geq 0$$

o dual de (P2) é :

$$(D) : \max \quad b^t v + u^t w$$

$$\text{s.a. } A^t v + w \leq c$$

$$w \leq 0$$

$$v \text{ irrestrito}$$

Para aplicar a (D) o algoritmo dual afim para variáveis livres faremos a seguinte identificação :

$\mathcal{A} = \begin{bmatrix} A & 0 \\ I & I \end{bmatrix}$, $\mathcal{b} = \begin{bmatrix} b \\ u \end{bmatrix}$, $\mathcal{c} = \begin{bmatrix} c \\ 0 \end{bmatrix}$, $\mathcal{v} \quad \begin{bmatrix} v \\ w \end{bmatrix}$ e os passos do algoritmo podem ser descritos como ,

<u>Passo DA.1:</u>

$$y_1^k = c - A^t.v^k - w^k$$

$$y_2^k = \qquad\qquad - w^k$$

<u>Passo DA.2:</u>

$$D_{y1}^k = \text{diag}[\ 1/(y_1^k)_1, \ \ldots\ , \ 1/(y_1^k)_n]$$

$$D_{y2}^k = \text{diag}[\ 1/(y_2^k)_1, \ \ldots\ , \ 1/(y_2^k)_n]$$

<u>Passo DA.3:</u>

$$\begin{bmatrix} d_v \\ d_w \end{bmatrix} = \left(\begin{bmatrix} A & 0 \\ I & I \end{bmatrix} . \begin{bmatrix} (D_{y1}^k)^2 & \\ & (D_{y2}^k)^2 \end{bmatrix} . \begin{bmatrix} A^t & I \\ 0 & I \end{bmatrix} \right)^{-1} . \begin{bmatrix} b \\ u \end{bmatrix}$$

uma análise mais detalhada da expressão acima fornece :

$$\begin{bmatrix} d_v \\ d_w \end{bmatrix} = \left(\begin{bmatrix} A.(D_{y1}^k)^2 & 0 \\ (D_{y1}^k)^2 & (D_{y2}^k)^2 \end{bmatrix} . \begin{bmatrix} A^t & I \\ 0 & I \end{bmatrix} \right)^{-1} . \begin{bmatrix} b \\ u \end{bmatrix}$$

$$\begin{bmatrix} d_v \\ d_w \end{bmatrix} = \begin{bmatrix} A.(D_{y1}^k)^2.A^t & A.(D_{y1}^k)^2 \\ (D_{y1}^k)^2.A^t & (D_{y1}^k)^2 + (D_{y2}^k)^2 \end{bmatrix}^{-1} . \begin{bmatrix} b \\ u \end{bmatrix}$$

esta última expressão é traduzida no seguinte sistema de equações lineares :

$$A.(D_{y1}^k)^2.A^t . d_v + A.(D_{y1}^k)^2 . d_w = b \quad (4.5)$$

$$(D_{y1}^k)^2.A^t . d_v + [(D_{y1}^k)^2 + (D_{y2}^k)^2] . d_w = u \quad (4.6)$$

de (4.6) temos :

$$[(D_{y1}^k)^2 + (D_{y2}^k)^2]^{-1} . (D_{y1}^k)^2.A^t . d_v + d_w = [(D_{y1}^k)^2 + (D_{y2}^k)^2]^{-1} . u$$

$$d_w = [(D_{y1}^k)^2 + (D_{y2}^k)^2]^{-1} . u - [(D_{y1}^k)^2 + (D_{y2}^k)^2]^{-1} . (D_{y1}^k)^2.A^t . d_v$$

substituindo em (4.5) temos :

$$A.(D_{y1}^k)^2.A^t.d_v - A.(D_{y1}^k)^2.[(D_{y1}^k)^2 + (D_{y2}^k)^2]^{-1} . (D_{y1}^k)^2 . A^t . d_v =$$

$$= b - A(D_{y1}^k)^2.[(D_{y1}^k)^2 + (D_{y2}^k)^2]^{-1} . u$$

que pode ser reescrito como :

$$A\{(D^k_{y1})^2.[(D^k_{y1})^2+(D^k_{y2})^2]^{-1}.[(D^k_{y1})^2+(D^k_{y2})^2] -$$

$$- (D^k_{y1})^2.[(D^k_{y1})^2+(D^k_{y2})^2]^{-1}.(D^k_{y1})^2).A^t.d_v \;=$$

$$b - A.(D^k_{y1})^2.[(D^k_{y1})^2 + (D^k_{y2})^2]^{-1}.u$$

portanto :

$$A.[(D^k_{y1})^2.[(D^k_{y1})^2+(D^k_{y2})^2]^{-1}.(D^k_{y2})^2] \;.A^t.d_v =$$

$$b - A.(D^k_{y1})^2.[(D^k_{y1})^2+(D^k_{y2})^2]^{-1}. \; u$$

Fazendo $\bar{D} = (D^k_{y1})^2.[(D^k_{y1})^2+(D^k_{y2})^2]^{-1}.(D^k_{y2})^2$ ficamos com :

$$\begin{cases} A.\bar{D}.A^t.d_v = b - A.(D^k_{y1})^2.[(D^k_{y1})^2 + (D^k_{y2})^2]^{-1}.u = \bar{b} \\ \qquad d_w = [(D^k_{y1})^2+ (D^k_{y2})^2]^{-1}.u - [(D^k_{y1})^2+ (D^k_{y2})^2]^{-1}.(D^k_{y1})^2.A^t.d_v \end{cases}$$

note que :

$$\bar{D} = [(D^k_{y1})^2 \; . \; (D^k_{y2})^2] \; . \; [(D^k_{y1})^2 + (D^k_{y2})^2]^{-1} =$$

$$\text{diag}\left\{ \frac{\dfrac{1}{(y^k_1)^2_i.(y^k_2)^2_i}}{\dfrac{1}{(y^k_1)^2_i} + \dfrac{1}{(y^k_2)^2_i}} \right\} = \text{diag}\left\{ \frac{1}{(y^k_1)^2_i + (y^k_2)^2_i} \right\}$$

$$\bar{b} = b - A \cdot \mathrm{diag}[1/(y_1^k)^2_i] \cdot \mathrm{diag}[1/(1/(y_1^k)^2_i + 1/(y_2^k)^2_i)] \cdot u$$

$$\bar{b} = b - A \cdot \mathrm{diag}\left[\frac{1}{(y_1^k)^2_i \dfrac{(y_1^k)^2_i + (y_2^k)^2_i}{(y_1^k)^2_i \cdot (y_2^k)^2_i}}\right] \cdot u$$

$$\bar{b} = b - A.\bar{D}.\mathrm{diag}[(y_2^k)^2_i].u$$

$$d_w = \mathrm{diag}\left[\frac{(y_1^k)^2_i \cdot (y_2^k)^2_i}{(y_1^k)^2_i + (y_2^k)^2_i}\right] \cdot u - \bar{D} \cdot \mathrm{diag}[(y_2^k)^2_i].A^t.d_v =$$

$$\mathrm{diag}[(y_1^k)^2_i.(y_2^k)^2_i].\bar{D}.u - \bar{D}.\mathrm{diag}[(y_2^k)^2_i].A^t.d_v$$

$$d_w = \mathrm{diag}[(y_1^k)^2_i].\bar{D}.\mathrm{diag}[(y_2^k)^2_i].u - \bar{D}.\mathrm{diag}[(y_2^k)^2_i].\ A^t.d_v$$

$$d_w = \bar{D}.\mathrm{diag}[(y_2^k)^2_i].[\ \mathrm{diag}[(y_1^k)^2_i.u] - A^t.d_v\ ]$$

Passo DA.4:

$$\begin{bmatrix} dy_1 \\ dy_2 \end{bmatrix} = \begin{bmatrix} -A^t & -I \\ 0 & -I \end{bmatrix} \begin{bmatrix} d_v \\ d_w \end{bmatrix} = \begin{bmatrix} -A^t.d_v - d_w \\ -d_w \end{bmatrix}$$

Passo DA.5:

$$x = -(D^k_{y1})^2.dy_1 \quad \Rightarrow \quad x_i = -(dy_1)_i \ / \ (y_1^k)^2_i$$

Passo DA.6:

$$(y_1^k)_i + \alpha.(dy_1)_i \geq 0 \Rightarrow \alpha \leq \frac{-(y_1^k)_i}{(dy_1)_i} \quad [p/\ (dy_1)_i < 0]$$

$$(y_2^k)_i + \alpha.(dy_2)_i \geq 0 \Rightarrow \alpha \leq \frac{-(y_2^k)_i}{(dy_2)_i} \quad [p/\ (dy_2)_i < 0]$$

portanto, para $(dy_1)_i$ e $(dy_2)_i$ estritamente negativos temos,

$$\alpha = \gamma \ . \ \min \left[\frac{-(y_1^k)_i}{(dy_1)_i} , \frac{-(y_2^k)_i}{(dy_2)_i} \right] , \ i = 1..n$$

Passo DA.7:

$$\begin{bmatrix} v^{k+1} \\ w^{k+1} \end{bmatrix} = \begin{bmatrix} v^k \\ w^k \end{bmatrix} + \alpha \ . \begin{bmatrix} d_v \\ d_w \end{bmatrix}$$

Passo DA.8:

k := k + 1

Em resumo, ficamos com o seguinte algoritmo :

início

Sejam v^o, w^o e γ dados tais que :

$$\begin{bmatrix} A^t & I \\ 0 & I \end{bmatrix} . \begin{bmatrix} v^o \\ w^o \end{bmatrix} < \begin{bmatrix} c \\ 0 \end{bmatrix} \quad \text{e } 0 < \gamma < 1$$

faça k = 0

enquanto "critério de parada não satisfeito" faça :

início

$$y_1^k = c - A^t.v^k - w^k \qquad \text{(DAC.1)}$$

$$y_2^k = - w^k \qquad \text{(DAC.2)}$$

$$D_{y1}^k = \text{diag}[\ 1/(y_1^k)_1, \ \ldots\ , \ 1/(y_1^k)_n] \qquad \text{(DAC.3)}$$

$$D_{y2}^k = \text{diag}[\ 1/(y_2^k)_1, \ \ldots\ , \ 1/(y_2^k)_n] \qquad \text{(DAC.4)}$$

$$A.\bar{D}.A^t.d_v = b - A.\bar{D}.\text{diag}[(y_2^k)_i^2].u \qquad \text{(DAC.5)}$$

$$d_w = \bar{D}.\text{diag}[(y_2^k)_i^2].(\ \text{diag}[(y_1^k)_i^2.u] - A^t.d_v\ ) \qquad \text{(DAC.6)}$$

$$\begin{bmatrix} dy_1 \\ dy_2 \end{bmatrix} = \begin{bmatrix} -A^t & -I \\ 0 & -I \end{bmatrix} \begin{bmatrix} d_v \\ d_w \end{bmatrix} = \begin{bmatrix} -A^t.d_v - d_w \\ - d_w \end{bmatrix}$$

$$x_i = - (dy_1)_i \ / \ (y_1^k)_i^2 \qquad \text{(DAC.7)}$$

para $(dy_1)_i$ e $(dy_2)_i$ negativos e $i = 1, \ldots, n$ (DAC.8)

$$\alpha = \gamma \ . \ \min \left[\frac{-(y_1^k)_i}{(dy_1)_i} \ , \ \frac{-(y_2^k)_i}{(dy_2)_i} \right]$$

$$\begin{bmatrix} v^{k+1} \\ w^{k+1} \end{bmatrix} = \begin{bmatrix} v^k \\ w^k \end{bmatrix} + \alpha \ . \ \begin{bmatrix} d_v \\ d_w \end{bmatrix} \qquad \text{(DAC.9)}$$

$$k = k + 1 \qquad \text{(DAC.10)}$$

fim

fim

Vejamos agora como podemos tratar o caso onde nem todas as variáveis são canalizadas. Uma primeira idéia seria atribuir às variáveis que não são canalizadas um limite superior grande(fácil de determinar nos casos práticos), porém durante as etapas de uma iteração do algoritmo dual afim para variáveis canalizadas seriam realizados cálculos com esses limites grandes.

A mistura de números grandes e pequenos em cálculos computacionais pode ser problemática pois alguns resultados intermediários que surgem durante a avaliação de expressões podem não caber na representação utilizada pela máquina. Por isso é conveniente que sejam evitadas tais situações. No caso do algoritmo dual afim para variáveis canalizadas, basta notar que às restrições geradas pelas canalizações de variáveis estão associadas as variáveis $\underline{w}$ do problema dual. Se fizermos $w_i = 0$ para as variáveis livres eliminamos os cálculos envolvendo limites grandes.

Por exemplo, na etapa (DAC.5) temos,

$$A.\bar{D}.A^t.d_v = b - A.\bar{D}.diag[(y_2^k)^2].u \qquad \text{(DAC.5)}$$

para as variáveis livres, $w_i = 0$, o que implica que $(y_2^k)_i = 0$. Assim se $g = diag[(y_2^k)^2].u$, cada compontente de g possui a forma $g_i = (y_2^k)_i^2.u_i$ e no caso $w_i = 0$ não vai importar o valor de u_i na expressão da etapa (DAC.5). Para as demais etapas vale um raciocínio parecido.

A única ressalva a esta idéia ocorre ao testarmos se o vetor x, solução do problema primal numa certa iteração, satisfaz as restrições de canalização. Se um componente de x não possui limite superior devemos levar isso em conta nos testes (no nosso caso atribuo o valor -1 como limite superior para tais variáveis).

4.2.3 OBTENÇÃO DE UM PONTO INTERIOR INICIAL

É fácil obter um ponto interior inicial para o algoritmo dual afim para variáveis canalizadas. De fato, observando o sistema

$$A^t v + w \leq c$$

$$w \leq 0$$

$$v \text{ , } w \text{ irrestritos}$$

verificamos que um ponto $[\, v^o \; w^o \,]$ com $v_i = 0$ e w_i dado por,

$$w_i = \begin{cases} -2 & \text{se} \quad c_i = 0 \\ 2.c_i & \text{se} \quad c_i < 0 \\ -2.c_i & \text{se} \quad c_i > 0 \end{cases} \qquad (4.7)$$

é um ponto interior inicial.

Entretanto testes realizados em [6] indicam que o procedimento tipo fase 1 descrito em [1] permite uma convergencia mais rápida. Assim, adotamos um procedimento análogo, descrito a seguir.

Supomos a princípio que não temos limites superiores nas variáveis. Isso acarreta que não temos as variaveis w_i e o problema assume a seguinte forma :

$$P': \quad \text{maximize } b^t.v$$

$$\text{sujeito a : } A^t.v \leq c$$

$$v \text{ irrestrito}$$

Note que no final da fase 1 temos um ponto interior para P'. Neste momento recolocamos os limites superiores que haviamos retirado. Isto é feito fazendo-se $w_i = \lambda$, onde λ é um valor negativo. É imediato verificar que o novo ponto $[v^o \; w^o]$ é interior ao problema (D).

É interessante observar que para obter o ponto v^o utilizamos explicitamente a matriz A^t, o que não era o caso da alternativa proposta nas formulas (4.7). Por outro lado, a eliminação temporária das canalizações não altera muito o problema já que só as variáveis inteiras precisarão de canalizações para o funcionamento do "branch and bound", e estas variáveis são minoria (felizmente).

Na prática, como já vimos, a eliminação dos limites superiores é feita zerando-se a correspondente folga w_i e utilizando uma marca ($u_i = -1$) para indicar que não há limite superior. Assim evita-se cálculos desnecessários.

De acordo com [1] adotamos o seguinte procedimento para obtenção do ponto interior inicial :

$$v^o = \left(\| c \| / \| A^t . b \| \right) . b$$

se $x^o = c - A^t . v^o > 0$, então v^o é um ponto interior inicial, caso contrário, adicionamos uma variável artificial ,

$$v_a^o = -2 \ . \ \min \left\{ \left[x_i^o : i = 1, \ \ldots \ , n \right], -1 \right\}$$

e $[v^o \quad v_a^o]$ será um ponto interior inicial para o seguinte problema :

$$P_a : \max \ b^t . v - M . v_a$$

$$\text{s.a. } A^t . v - e . v_a \leq c$$

$$v \ , \ v_a \ \text{ irrestritos, } e = (1, 1, \ldots, 1)^t,$$

$$M = \mu \ . \ | \ b^t . v^o | \ / \ v_a^o \ , \ \mu \text{ uma constante grande}$$

Aplicamos o algoritmo a P_a e iteramos até que $v_a < 0$. Se o critério de parada normal é satisfeito e $v_a > 0$ o problema original é infactivel. Se $v_a \Rightarrow 0$ quando $v^k \Rightarrow v^*$ (v^* solução ótima) significa que P' não tem ponto interior. Neste caso, continuamos normalmente com a fase 2 sem eliminar a variável artificial.

4.2.4 CRITÉRIO DE PARADA

As iterações do algoritmo dual afim podem ser interrompidas quando as condições de otimalidade estiverem satisfeitas segundo alguma tolerância fornecida, quando o acréscimo relativo da função objetivo é muito pequeno, ou quando se verifica que o problema é ilimitado.

Adler e outros[1] sugerem que podemos iterar até que :

$$| c^t.x^k - c^t.x^{k-1} | \; / \; \max (1, | c^t.x^{k-1} |) < \varepsilon$$

onde $c^t = [\, b \quad u \,]$, $x^k = [\, v^k \quad w^k \,]$, ε real positivo pequeno.

Note que no processo de "branch and bound" estaremos resolvendo problemas que possuem a característica de serem inicialmente primais infactíveis (lembre das restrições acrescentadas na geração de subproblemas). Assim o critério de parada que utilizamos contém também um teste para verificar se a solução de problema primal é factivel, pois pode ocorrer a desigualdade acima sendo o primal infactível.

Quanto ao valor utilizado para ε, testamos valores não menores do que 10^{-4}, pois estamos trabalhando com custos de planejamento que são obtidos por parâmetros aproximados (não teria significado nos preocuparmos com frações de centavos).

4.2.4.1 TESTE DE INFINITUDE

Quando o aumento relativo da função objetivo começa a crescer muito, podemos suspeitar que o problema alvo do algoritmo dual afim é ilimitado. Nos casos que tratamos, o aumento relativo da função objetivo a cada iteração é quase sempre menor do que um. Então se o aumento relativo ultrapassa um certo valor alto (no nosso caso 100) interrompemos o processo de solução e declaramos o problema ilimitado. Este procedimento mostrou-se adequado em 100% dos casos testados.

O algoritmo foi programado na linguagem C. A maior parte das funções estão relacionadas ao método dual afim. Foram também criadas funções que inicializam e manipulam as estruturas de dados utilizadas(ver tópico seguinte).

O programa principal consiste no mecanismo de "branch and bound". Ele chama uma função que executa o algoritmo dual afim e gera problemas que são armazenados na memória principal.

5.1 ESTRUTURA DE DADOS

A estrutura de dados utilizada no método dual afim é a mesma sugerida por Adler e outros em [2]. São mantidas na memória as matrizes A , A^t , $A\bar{D}A^t$ além de mantermos espaço adicional para o armazenamento da triangularização resultante da resolução do sistema linear em cada iteração do método dual afim. Deste modo, podemos atualizar $A\bar{D}A^t$ numa iteração ao invés de a calcularmos novamente a cada iteração.

Para o armazenamento da matriz A, adota-se uma estrutura de vetores coluna esparsos, mencionados a seguir.

a : contém os elementos não nulos da matriz A em ordem de colunas.

ja : cada posição indica o número da linha na qual o elemento na mesma posição no vetor a, ocorre na matriz A.

ia : cada posição indica o início de cada coluna no vetor a.

Também utiliza uma posição extra para indicar o término da última coluna.

Para o armazenamento da matriz A^t, utilizamos vetores semelhantes aos utilizados para o armazenamento de A, trocando-se linhas por colunas. Os vetores correspondentes à a, ia e ja são at, iat, e jat.

No caso da matriz $A\bar{D}A^t$ utilizamos o armazenamento sob a forma de vetores linhas esparsos para os elementos fora da diagonal principal.

aat : contém os elementos não nulos de $A\bar{D}A^t$ em ordem de linhas.

aat_u : contém os elementos do fator triangular resultante da eliminação gaussiana feita em $A\bar{D}A^t$.

jaat :contém os números das colunas respectivos a cada elemento em aat.

iaat : contém a posição inicial de cada linha nos vetores aat e jaat.

diag : contém os elementos da diagonal principal de $A\bar{D}A^t$.

diag_u : contém os elementos da diagonal principal do fator triangular resultante da eliminação gaussiana em $A\bar{D}A^t$.

No espaço utilizado para a matriz $A\bar{D}A^t$ estão previstas posições para o fill-in gerado durante a eliminação gaussiana.

As figuras a seguir ilustram as estruturas de dados utilizadas.

Figura 5.1 : matriz A

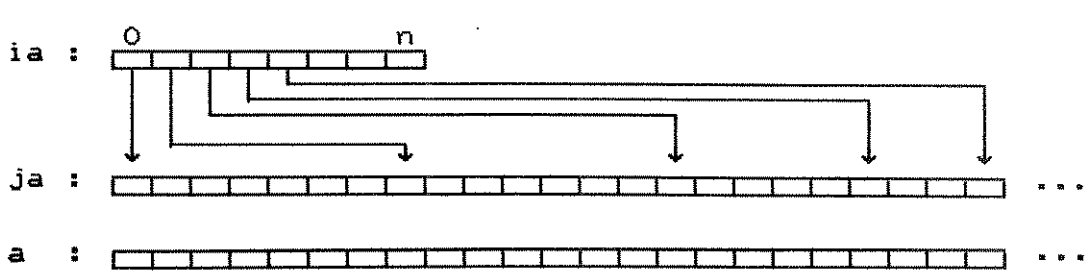


Figura 5.2 : matriz A^t

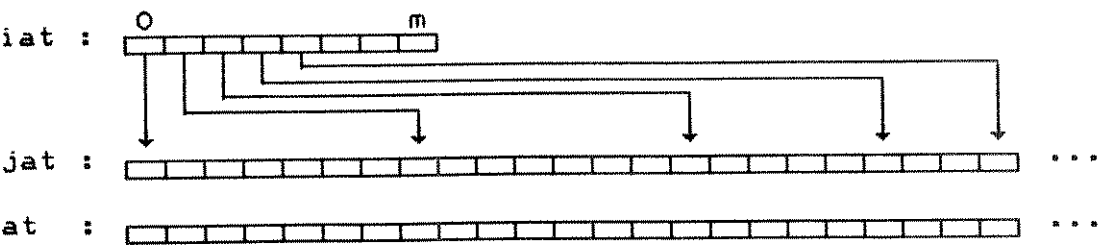
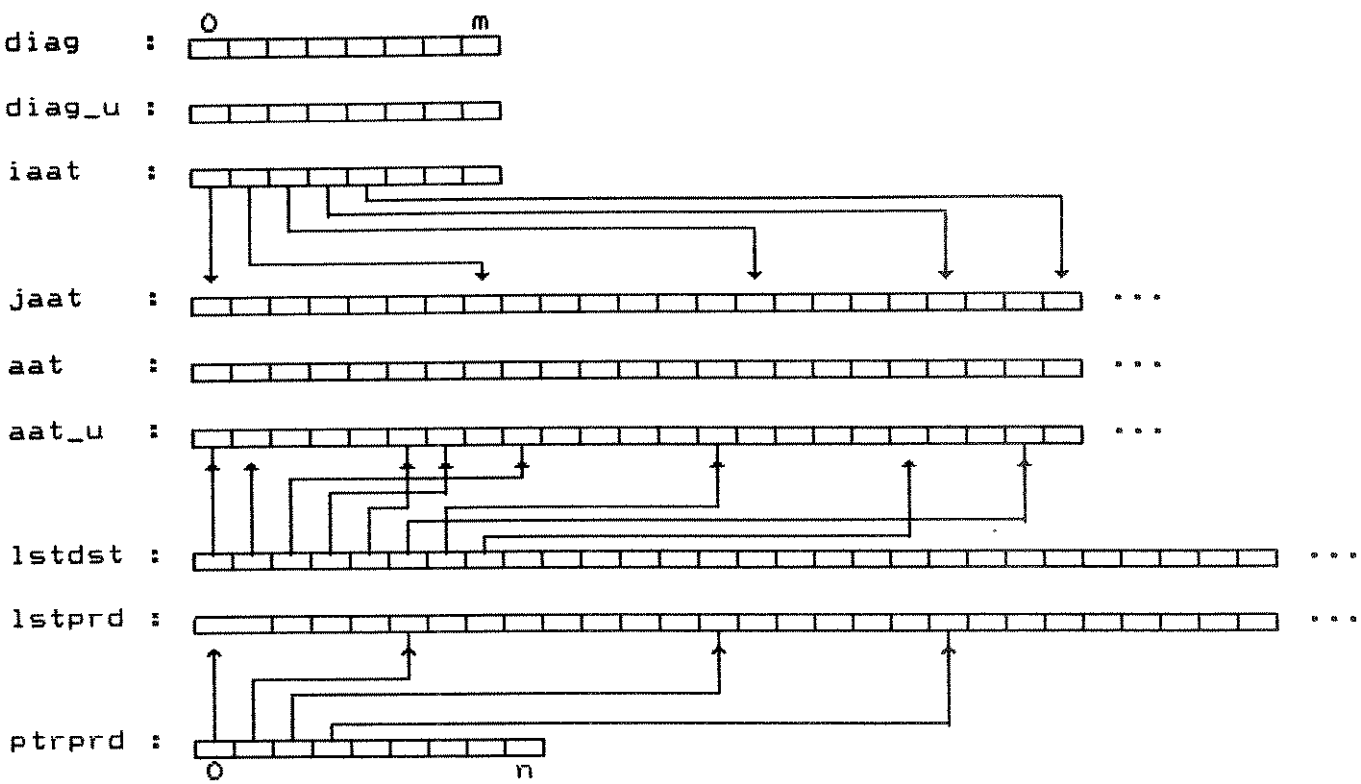


Figura 5.3 : matriz $A.\bar{D}.A^t$



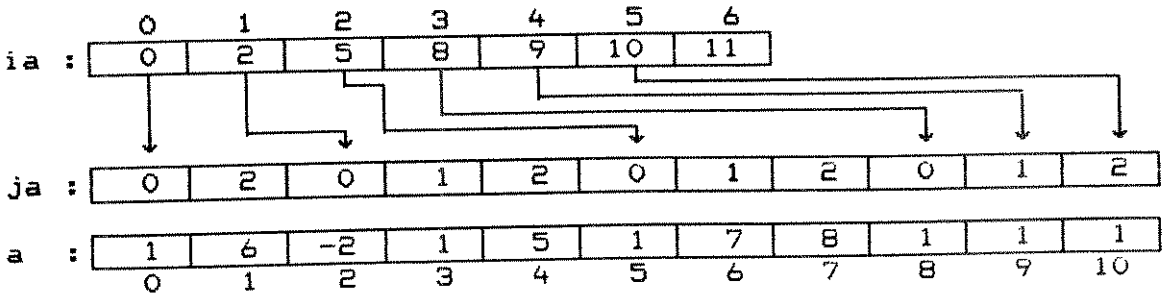
Obs.: os vetores lstdst, lstprd e ptrprd serão comentados no próximo item. Eles estão incluidos agora na figura 5.3 por questão de comodidade.

Para exemplificar, considere a seguinte matriz :

$$A = \begin{bmatrix} 1 & -2 & 1 & 1 & 0 & 0 \\ 0 & 1 & 7 & 0 & 1 & 0 \\ 6 & 5 & 8 & 0 & 0 & 1 \end{bmatrix} \qquad m = 3,\ n = 6$$

A matriz A será então armazenada da seguinte maneira :

Figura 5.4 : exemplo de estruturas de dados



5.2 MONTAGEM DA MATRIZ $A\bar{D}A^t$

Esta etapa do algoritmo dual afim exige um esforço razoável. Para acelerá-la, utilizamos a técnica encontrada em [2], que consiste em calcular parte dos produtos internos entre cada linha de A, mantendo esses produtos num vetor.

Note que cada elemento de $A\bar{D}A^t$ é formado pelo produto interno entre duas linhas, de $A\bar{D}$ e A^t , respectivamente. Ou seja,

$$B(i,j) = \sum_{k=1}^{n} A(i,k) * A(j,k) * \bar{D}(k)$$

onde B(i,j) é um elemento de $A\bar{D}A^t$.

Cada produto A(i,k) * A(j,k) é calculado uma única vez no início do algoritmo, sendo os mesmos armazenados nos vetores descritos a seguir.

lstprd : contém cada produto A(i,k) * A(j,k) em ordem crescente de colunas.

lstdst : contém a posição de aat e aat_u onde deve ser acumulado o resultado de cada produto com elementos em lstprd.

ptrprd : contém a posição inicial, para cada coluna de A, da seqüência de produtos em lstprd e lstdst.

Os vetores acima são inicializados no começo do algoritmo e utilizados na montagem ou atualização de $A\bar{D}A^t$. Na primeira iteração calculamos $A\bar{D}A^t$ exatamente e nas iterações posteriores a atualizamos, visando diminuir o esforço computacional (no próximo tópico descreveremos este aspecto com mais detalhes).

Devido à resolução do sistema linear no método dual afim envolvendo $A\bar{D}A^t$, temos que alocar zeros em $A\bar{D}A^t$ sempre que numa linha já houver ocorrido a geração de um elemento não nulo. Por exemplo , se na linha j de $A\bar{D}A^t$ o primeiro elemento não nulo é a_{jk}, então guardamos as posições para os elementos que surgirem na decomposição; isto é feito gravando-se zeros nas posições $a_{(j+1)k}$, $a_{(j+2)k}$, etc.

É justamente este o motivo de existirem técnicas de reordenamento das linhas da matriz A visando à obtenção de uma matriz $A\bar{D}A^t$, de forma que o número de novos elementos gerados na eliminação gaussiana seja mínimo.

5.3 ATUALIZAÇÃO DE $A\bar{D}A^t$

A atualização da matriz $A\bar{D}A^t$ é obtida com o cálculo aproximado da matriz diagonal $\bar{D}$, mantendo alguns componentes $\bar{D}(i,i)$ de uma iteração para a outra, desde que o desvio relativo destas componentes entre iterações subseqüentes seja pequeno.

Para a atualização de $A\bar{D}A^t$ utilizamos a relação descrita em Adler e outros [2] , $A.\bar{D}_k.A^t = A.D_{k-1}.A^t + A.\Delta.A^t$, onde Δ é uma matriz diagonal. Cada elemento Δ_i de Δ é determinado através de :

$\Delta_i = 0$

se $|\bar{D}_k(i,i) - \bar{D}_{k-1}(i,i)| \;/\; |\bar{D}_{k-1}(i,i)| < \varepsilon_u$

$\Delta_i = \bar{D}_k(i,i) - \bar{D}_{k-1}(i,i)$

se $|\bar{D}_k(i,i) - \bar{D}_{k-1}(i,i)| \;/\; |\bar{D}_{k-1}(i,i)| \geq \varepsilon_u$

Adler e outros[2] relatam ganhos de 10% no tempo de processamento graças ao uso desta forma de atualização. O valor de $\varepsilon_u = 0.001$ foi usado em nossos testes.

5.4 PONTO INTERIOR INICIAL PARA SUBPROBLEMAS

Um resultado fundamental da programação linear é que uma solução ótima de um problema de programação linear se encontra num extremo do politopo de restrições[24]. Isso significa computacionalmente que algumas das componentes das folgas y_1 e y_2 assumirão, no ótimo, valores pequenos. Por conseguinte, a matriz D pode perder a sua

precisão de maneira a inviabilizar o cálculo das projeções(d_w e d_v). Detalhes sobre erros em operações de ponto flutuante podem ser encontrados em [22] e [23].

Para evitar tais problemas fizemos o seguinte :

Ao aplicarmos o algoritmo dual afim ao primeiro subproblema relaxado verificamos quando o decréscimo relativo da função objetivo é menor do que uma tolerância ε_0. Quando isso acontece gravamos a solução atual (do problema dual) num vetor, sendo esses valores utilizados como ponto de partida para todos os problemas gerados no processo de "branch and bound". Este procedimento é encarado como uma fase 3 do algoritmo. Então temos as seguintes etapas iniciais :

Fase 1 : obter um ponto interior inicial para o problema relaxado.

Fase 3 : iterar até que o descréscimo relativo seja menor do que ε_0, quando isso acontece salve a solução corrente e passe para a fase 2.

Fase 2 : iterar até que o critério de otimalidade seja satisfeito.

A idéia desse procedimento é aproveitar uma parte do esforço feito para obter a primeira solução, ou seja, partindo-se de uma solução inicial o caminho seguido pelo algoritmo até a solução ótima de um problema seria parecido para os diversos problemas que diferem entre si na canalização de algumas poucas variáveis. Além disso, a lista mestra não precisará conter soluções de partida, o que representa uma economia considerável em termos de memória necessária para a mesma.

Outra motivação à idéia da fase 3 é o fato de mantermos como

ponto inicial para cada problema, um ponto interior mais longe das paredes do politopo de restrições do que uma solução ótima de um problema prévio no processo de "branch and bound". Sendo o dual afim um algoritmo de ponto interior, não é sensato partir de pontos próximos da fronteira da região definida pelas restrições.

5.5 MANIPULAÇÃO DA MATRIZ TECNOLÓGICA

Sabemos que o maior esforço por iteração no algoritmo de ponto interior ocorre na resolução do sistema linear simétrico envolvendo a matriz $A\bar{D}A^t$. Assim é muito importante melhorar o desempenho desta etapa.

Organizamos as linhas da matriz A de modo que a matriz $A\bar{D}A^t$ fique com o menor número possível de elementos diferentes de zero. Para ver como cada elemento de $A\bar{D}A^t$ é formado, suponha que $\bar{D} = I$ e note que os elemento da matriz AA^t são formados pelos produtos internos de cada par de linhas. Disso inferimos que se uma linha muita densa está nas primeiras posições, o seu produto interno com as outras terá mais chance de produzir um elemento não nulo em AA^t, ou seja, criar "fill-in". Para contermos este problema adotamos o procedimento descrito a seguir.

As restrições (2.2) , de área , e (2.3) , de precedência, são colocadas como as últimas linhas da matriz tecnológica,pois elas envolvem variáveis de muitos períodos; conseqüentemente são mais densas. As restrições (2.4) , de tratores, e (2.5) , de implementos, são colocadas em primeiro lugar, pois envolvem períodos isolados (são menos densas).

Além dessa organização da matriz A, executa-se um algoritmo chamado Minimal Degree [20] (descrito no ítem 5.8), procurando-se melhorias adicionais na esparsidade de $A\bar{D}A^t$.

5.6 RESOLUÇÃO DO SISTEMA LINEAR

O maior esforço computacional do algoritmo dual afim reside na resolução do sistema linear $A\bar{D}A^t.dv = \bar{b}$. A matriz $A\bar{D}A^t$ é simétrica e definida positiva (veja observação no fim deste ítem). Assim, só precisamos guardar metade dos elementos de $A\bar{D}A^t$, e para resolver o sistema podemos utilizar decomposição de Cholesky. Entretanto, esta opção requer a utilização de raízes quadradas, o que torna a decomposição mais lenta e imprecisa. Preferimos então, utilizar a tradicional eliminação gaussiana como mencionada em [2].

No inicio de cada eliminação gaussiana copiamos as elementos de aat para aat_u e diag para diag_u, passando então a trabalhar com aat_u e diag_u, aplicando as operações elementares também no vetor $\bar{b}$, até obtermos o fator triangular superior U e o vetor transformado b'. Finalmente obtemos a solução do sistema por substituição regressiva.

Observação : $\bar{ADA}^t$ é simétrica e definida positiva pois,

i) $(\bar{ADA}^t)^t = A(\bar{AD})^t = \bar{ADA}^t$

ii) para qualquer vetor x não nulo temos : $x^t\bar{ADA}^t x > 0 \Rightarrow y^t\bar{D}y > 0$

pois cada elemento de $\bar{D}$ é estritamente positivo.

5.7 ECONOMIZANDO MEMORIA

Pode-se economizar algum espaço na memória principal se notarmos que :

i) não é preciso conhecer , na prática, os valores assumidos pelas variáveis y.

ii) $y_2 = - w$ (logo y_2 não é usado explicitamente)

iii) $dy_2 = - d_w$ (idem)

Numa fase inicial fazemos um pré-processamento da matriz tecnológica eliminando as restrições que possuem somente um elemento. Isso é feito igualando-se a variável correspondente ao valor do seu recurso b_i e a seguir atualizando-se o vetor b. Esta tarefa é feita por um pré-processador que tem também a função de aplicar o algoritmo Minimal Degree à estrutura simbólica de AA^t (veja tópico seguinte). Quando ocorre a eliminação de uma variável desta maneira o usuário é informado para que possa levar em conta esse fato novo. Num sistema integrado seria fácil gerar um arquivo com as variáveis eliminadas e seus valores para depois remontarmos a solução do problema completo.

5.8 ACELERAÇÃO DO ALGORITMO

No início de cada iteração calculamos e armazenamos as matrizes diagonais $diag((y_1)_i^2)$ e $diag((y_2)_i^2)$, pois as mesmas são utilizadas na montagem do sistema linear e solução do problema primal.

Assim, economizamos algumas multiplicações. Por um motivo análogo calculamos uma única vez em cada iteração o produto $\bar{D}.diag((y_2)_i^2)$, pois o mesmo é utilizado no cálculo de d_w e $\bar{b}$.

Como o maior esforço por iteração do algoritmo dual afim reside na resolução do sistema $A.\bar{D}.A^t.d_v = \bar{b}$, aplicamos o algoritmo "Minimal Degree" à matriz A, visando minimizar o fill-in gerado na eliminação gaussiana.

O nome "Minimal Degree" origina-se no fato de podermos representar a esparsidade de uma matriz sob a forma de um grafo. O termo "Degree" (do inglês grau) se refere ao grau de cada nó desse grafo [20].

O algoritmo "Minimal Degree" na verdade é uma eliminação gaussiana simbólica, ou seja, não são realizadas operações aritméticas sendo somente verificados se os elementos são nulos ou não. Como a matriz $A\bar{D}A^t$ é simétrica e definida positiva não olhamos para a magnitude dos pivôs [23]. Note também que permutar duas linhas de A equivale a fazer uma permutação simétrica em $A\bar{D}A^t$, ou seja, permutar seqüencialmente as linhas e colunas de $A\bar{D}A^t$.

O algoritmo "Minimal Degree" foi programado como um pré-processador da matriz tecnológica. Assim, antes de aplicarmos o "branch and bound" o pré-processador cria outro arquivo com a matriz tecnológica reordenada.

Existem outros algoritmos para diminuir o esforço na resolução de um sistema linear. Alguns exemplos seriam "Minimum local fill-in" e o algoritmo de Tarjan para transformar uma matriz na forma bloco triangular, facilitando desta forma a solução do sistema[20].

6. ESTUDO DE CASO

6.1 CONSIDERAÇÕES INICIAIS

A seguir passaremos a mostrar os resultados dos testes feitos com as idéias do presente trabalho. Os testes estão divididos em duas classes, testes com o dual afim e testes com o "branch and bound".

Os testes como o dual afim visam avaliar o qualidade da implementação utilizada e para isso o dual afim é comparado ao simplex. Os testes com o "branch and bound" visam comparar o seu desempenho com um algoritmo "branch and bound" que percorre os pontos extremos do politopo do problema (método dual simplex canalizado).

Portanto, para complementar os testes feitos implementamos versões do algoritmo simplex [26] e do algoritmo dual simplex canalizado [27].

Tais algoritmos utilizam estruturas de dados as mais parecidas possíveis às utilizadas pelo dual afim e foram também compilados pelo mesmo compilador.

O algoritmo simplex utiliza a forma produto da inversa da base e para reinversões foi utilizado o algoritmo de Larsen [26], o que torna o simplex mais robusto. Uma das finalidades de implementarmos o algoritmo simplex foi utilizá-lo como inicializador de um algoritmo "branch and bound" baseado no algoritmo dual simplex canalizado.

A algoritmo dual simplex canalizado é descrito em [27], e permite que, partindo-se de uma solução dual factível (porém primal infactível devido à alguma canalização violada), tentemos obter a factibilidade do problema primal sem acrescentar restrições explicitamente. A

finalidade de sua implementação foi utilizá-lo no lugar do algoritmo dual afim no processo de "branch and bound", e com isso fazermos comparações.

Este "branch and bound" tem muitas partes em comum com o baseado no dual afim. Utiliza estruturas de dados parecidas e o mecanismo de "branch and bound" apresenta as mesmas características descritas no item 4.1.

As maioria das medições foram feitas num computador da linha PC-XT com clock de 10 Mhz e co-processador 8087, o que representa uma máquina 215% mais rápida do que um PC padrão. Os tempos medidos não incluem o pré-processamento da matriz tecnológica, o tempo gasto na sua leitura, assim como o tempo gasto na inicialização das estruturas utilizadas pelo algoritmo. Os problemas de maior porte, nos testes com o "branch and bound", foram otimizados num computador VAX 11/785.

6.2 TESTES COM O DUAL AFIM

Para averiguarmos se a nossa implementação do algoritmo dual afim está razoável, fizemos a sua comparação com o simplex mencionado, utilizando alguns problemas da série NETLIB.

Na tabela 1, temos o resultado da aplicação dos algoritmos dual afim e simplex a alguns problemas da série NETLIB. Nesta tabela temos as seguintes informações :

tempo : tempo gasto em segundos.

média : tempo gasto pelo simplex dividido pelo tempo gasto pelo dual afim.

Problema	Simplex tempo(s)	Dual Afim tempo(s)	Média
AFIRO	10	10	1.0
ADLITTLE	90	53	1.7
SCAGR7	194	73	2.6
SHARE2B	166	110	1.5

Tabela 1 : comparação entre o dual afim e o simplex

A seguir temos os resultados da aplicação dos algoritmos dual afim e simplex a alguns problemas do nosso modelo de planejamento agrícola. Tais problemas apresentam dimensões maiores do que os utilizados para testar o "branch and bound", e apresentaram a característica de seus duais não apresentarem pontos interiores. Observe que, neste caso não consideramos variáveis inteiras.

Na tabela 2 representamos os dados básicos destes problemas e na tabela 3 os resultados das comparações. Na tabela 2 representamos os seguintes parâmetros :

M : número de linhas do problema.

N : número total de variáveis do problema.

NZ(A) : quantidade de elementos não nulos na matriz A.

$NZ(AA^t)$: quantidade de elementos não nulos na matriz AA^t, incluindo o fill-in gerado na eliminação gaussiana.

%A : percentual de esparsidade da matriz A.

$\%AA^t$: percentual de esparsidade da matriz AA^t.

Problema	M	N	NZ(A)	NZ(AA^t)	%A	%AA^t
COPERADL.001	52	93	209	941	4.32	34.8
COPERADL.002	87	161	384	2536	2 74	33.5
COPERADL.003	121	222	526	5027	1.96	34.3
COPERADL.004	139	272	652	7182	1.72	37.2

Tabela 2 : dimensões dos problemas

Na tabela 3 temos as seguintes informações :

tempo : tempo gasto em segundos.

média : tempo gasto pelo simplex dividido pelo tempo gasto pelo dual afim.

Problema	Simplex tempo(s)	Dual Afim tempo(s)	Média
COPERADL.001	31	54	0.57
COPERADL.002	63	180	0.35
COPERADL.003	124	467	0.26
COPERADL.004	176	794	0.22

Tabela 3 : comparação entre o dual afim e o simplex

6.3 TESTES COM O "BRANCH AND BOUND"

Para testar o algoritmo "branch and bound" utilizamos dados gentilmente fornecidos pela Cooperativa Paulista COPERSUCAR, empresa na qual o autor deste trabalho atuou como estagiário. Nessa oportunidade participamos do desenvolvimento de um sistema de planejamento que utilizava programação linear [15].

Através de pequenas alterações no programa de geração da matriz tecnológica efetuamos a geração de matrizes para o problema misto envolvido no presente trabalho.

Na tabela 4 temos as estatísticas, em termos de dimensões, dos problemas testados. Nesta tabela representamos os seguintes parâmetros:

Z : número de variáveis inteiras do problema.

M : número de linhas do problema.

N : número total de variáveis do problema (inclusive inteiras).

NZ(A) : quantidade de elementos não nulos na matriz A.

NZ(AA^t) : quantidade de elementos não nulos na matriz AA^t, incluindo o fill-in gerado na eliminação gaussiana.

%A : percentual de esparsidade da matriz A.

%AA^t : percentual de esparsidade da matriz AA^t.

Este problemas visam acompanhar a evolução do esforço computacional quando aumentamos as dimensões do problema e também o número de variáveis inteiras.

Problema	Z	M	N	NZ(A)	NZ(AA)	%A	%AA
PSMORD.1Z0	0	10	24	40	39	16.67	39.0
PSMORD.1Z2	2	10	24	40	39	16.67	39.0
PSMORD.1Z4	4	10	24	40	39	16.67	39.0
PSMORD.1Z8	8	10	24	40	39	16.67	39.0
PSMORD.2Z0	0	16	34	64	93	11.76	36.3
PSMORD.2Z2	2	16	34	64	93	11.76	36.3
PSMORD.2Z4	4	16	34	64	93	11.76	36.3
PSMORD.2Z8	8	16	34	64	93	11.76	36.3
PSMORD.3Z0	0	18	41	83	119	11.25	36.7
PSMORD.3Z2	2	18	41	83	119	11.25	36.7
PSMORD.3Z4	4	18	41	83	119	11.25	36.7
PSMORD.3Z8	8	18	41	83	119	11.25	36.7
PSMORD.4Z0	0	19	42	83	145	11.53	40.1
PSMORD.4Z2	2	19	42	83	145	11.53	40.1
PSMORD.4Z4	4	19	42	83	145	11.53	40.1
PSMORD.4Z8	8	19	42	83	145	11.53	40.1

Tabela 4 : dimensões dos problemas

Na tabela 5 temos dados referentes ao tempo gasto na execução completa do algoritmo "branch and bound" com o dual afim, nesta tabela temos :

probs. : quantidade de problemas resolvidos durante o processo de "branch and bound".

it_tot : total de iterações gasto.

it_med : número médio de iterações por problema resolvido no processo de "branch and bound".

tempo(s) : tempo total gasto, em segundos.

As tolerâncias utilizadas nos testes foram as seguintes :

eps = 0.01

γ = 0.95

ε_o = 0.1

ε_u = 0.001

μ = 10^5 (número grande para a fase 1)

problema	probs.	it_tot	it_médio	tempo(s)
PSMORD.1Z0	1	21	21.0	5
PSMORD.1Z2	1	21	21.0	4
PSMORD.1Z4	5	63	12.6	13
PSMORD.1Z8	9	138	15.3	28
PSMORD.2Z0	1	22	22.0	6
PSMORD.2Z2	1	22	22.0	7
PSMORD.2Z4	3	42	14.0	12
PSMORD.2Z8	23	286	12.4	78
PSMORD.3Z0	1	23	23.0	7
PSMORD.3Z2	1	23	23.0	7
PSMORD.3Z4	não converge	idem	idem	idem
PSMORD.3Z8	39	488	12.5	151
PSMORD.4Z0	1	23	23.0	9
PSMORD.4Z2	1	23	23.0	9
PSMORD.4Z4	13	169	13.0	59
PSMORD.4Z8	33	383	11.6	132

Tabela 5 : resultados dos testes com o dual afim no branch and bound

A tabela 6 apresenta o resultado da aplicação do "branch and bound" baseado no dual simplex aos mesmos problemas da tabela 4.

problema	subprobs.	it_tot	it_médio	tempo(s)
PSMORD.1Z0	1	13	13.0	3
PSMORD.1Z2	1	13	13.0	3
PSMORD.1Z4	5	17	3.4	4
PSMORD.1Z8	9	25	2.7	7
PSMORD.2Z0	1	29	29.0	7
PSMORD.2Z2	1	29	29.0	6
PSMORD.2Z4	3	34	11.3	8
PSMORD.2Z8	21	69	3.3	20
PSMORD.3Z0	1	29	29.0	6
PSMORD.3Z2	1	29	29.0	7
PSMORD.3Z4	9	42	4.6	12
PSMORD.3Z8	29	77	2.6	26
PSMORD.4Z0	1	29	29.0	6
PSMORD.4Z2	1	29	29.0	7
PSMORD.4Z4	9	38	4.2	12
PSMORD.4Z8	29	69	2.4	25

Tabela 6 : resultados dos testes com o dual simplex no branch and bound.

Na tabela 7 temos uma comparação entre os dados ds tabelas 5 e 6, nesta tabela temos as seguintes informações :

tempo : tempo gasto em segundos.

média : tempo gasto pelo "branch and bound" utilizando o dual simplex dividido pelo tempo gasto pelo "branch and bound" utilizando o dual afim.

problema	Branch and bound : dual simplex tempo(s)	Branch and bound : dual afim tempo(s)	media
PSMORD.1Z0	3	5	0.60
PSMORD.1Z2	3	4	0.75
PSMORD.1Z4	4	13	0.31
PSMORD.1Z8	7	28	0.25
PSMORD.2Z0	7	6	1.17
PSMORD.2Z2	6	7	0.86
PSMORD.2Z4	8	12	0.67
PSMORD.2Z8	20	78	0.26
PSMORD.3Z0	6	7	0.86
PSMORD.3Z2	7	7	1.00
PSMORD.3Z4	12	--	----
PSMORD.3Z8	26	151	0.17
PSMORD.4Z0	6	9	0.67
PSMORD.4Z2	7	9	0.78
PSMORD.4Z4	12	59	0.20
PSMORD.4Z8	25	132	0.19

Tabela 7 : comparação entre o branch and bound com o dual simplex e branch and bound com dual afim

Para finalizar, a tabela 8 apresenta os resultados da aplicação do algoritmo "branch and bound" com o dual afim e com dual simplex aos problemas da tabela 2, aos quais exigimos que cinco de suas variáveis fossem tratadas como inteiras. Lembramos que tais testes foram feitos num computador VAX 11/785. Nesta tabela temos a seguinte notação :

BBDA : "Branch and Bound" com Dual Afim.

BBDS : "Branch and Bound" com Dual Simplex.

num. probls. resolvidos : número de problema resolvidos durante o processo de "branch and bound".

total de iterações : soma das quantidades de iterações gastas por todos os problemas resolvidos durante o processo de "branch and bound".

média iter. por probl. : número médio de iterações por problema resolvido durante o "branch and bound".

tempo(s) : tempo total gasto, em segundos.

média BBDS/BBDA : tempo gasto em BBDS dividido pelo tempo gasto em BBDA.

problema	num.probls. resolvidos		total de iterações		média iter. por probl.		tempo(s)		média BBDS/ BBDA
	BBDA	BBDS	BBDA	BBDS	BBDA	BBDS	BBDA	BBDS	
COPERADL.001	15	13	157	88	10.5	6.8	137	49	0.36
COPERADL.002	15	17	162	166	10.8	9.8	1631	131	0.08
COPERADL.003	7	7	102	220	14.6	31.4	1742	65	0.04
COPERADL.004	1	1	35	249	35.0	249.0	599	84	0.14

Tabela 8 : outra comparação entre o branch and bound com o dual afim e branch and bound com o dual simplex

7. DISCUSSÃO E COMENTÁRIOS

O método dual afim gasta menos iterações na resolução de problemas quando o ponto de partida é o obtido na fase 3, do que se tivesse sido resolvido do início. Também notamos que o fato de alterarmos as canalizações de uma dada variável exige que o critério de parada do algoritmo dual afim inclua um teste de factibilidade do primal , pois somente o decréscimo relativo da função objetivo não garante esta última condição.

Notamos que se a solução ótima de um subproblema é utilizada como ponto interior inicial para outro problema, o algoritmo dual afim falha. A razão disso está no fato de que uma solução ótima de um subproblema se encontra num extremo do politopo de restrições e portando as folgas y_1 e y_2 se tornam muito pequenas numericamente. Assim, no caso de uma solução num extremo a matriz $\bar{D}$ pode apresentar elementos muito grandes acarretando que a $A\bar{D}A^t$ se torne mau condicionada (para ver isso basta notar que cada elemento de $A\bar{D}A^t$ envolve produtos internos de linhas de A juntamente com elementos (grandes) de $\bar{D}$).

Outro fato digno de nota é a baixa esparsidade de AA^t. Observamos que a média desta esparsidade é de 38%. Adler e outros [1] enfatizam que o método dual afim é extremamente sensível à densidade de $A\bar{D}A^t$ - manter esta matriz esparsa é essencial ao bom desempenho do algoritmo. Através das tabelas encontradas em [1] e [6], relacionadas ao conjunto de problemas teste NETLIB , onde o dual afim mostrou bom desempenho, verificamos que a média de esparsidade é de 5.2 % .Isso significa que o modelo de planejamento agrícola não mostra o alto desempenho do

método dual afim, pois a resolução do sistema linear é um grande consumidor de tempo em cada iteração.

Observamos que a medida que a dimensão dos problemas crescem, o desempenho do algoritmo dual afim diminui, e portanto o "branch and bound", que utiliza o dual afim, também tem seu desempenho reduzido.

O valor da tolerância $\varepsilon_o = 0.1$ foi obtido através de muitos testes. Se ε_o é muito próximo de eps, observam-se os problemas numéricos já mencionados. Se ε_o é muito grande o número de iterações gasto em cada subproblema aumenta, tornando o "branch and bound" ineficiente.

8. CONCLUSÃO

A idéia de se resolver os subproblemas de um algoritmo "branch and bound" utilizando o método dual afim foi explorada no modelo de planejamento agrícola. Uma das motivações para essa idéia foi o grande sucesso da aplicação do algoritmo dual afim a problemas de programação linear.

Concluímos que as principais dificuldades da idéia residem na baixa esparsidade apresentada pela matriz $A\bar{D}A^t$ e no fato de uma solução ótima de um problema relaxado se encontrar perto de um extremo.

Como vantagem, assinalaria a facilidade com que o método dual afim se aproxima do ótimo já que as soluções do mesmo são interiores ao politopo de restrições do problema. A introdução de restrições de canalização não afeta o método de obtenção de um ponto inicial dual factível.

Finalmente, acreditamos que para problemas onde a estrutura de AA^t é muito esparsa, a idéia de utilizar o dual afim no "branch and bound" deve garantir bons ganhos em termos de tempo de processamento.

Nosso comentário sugere a montagem de um algoritmo "branch and bound" de uso geral. Neste, os problemas gerados durante o processo são resolvidos ou pelo dual afim, ou pelo dual simplex, sendo a decisão tomada automaticamente através da avaliação da esparsidade da matriz AA^t. Tal avaliação pode ser feita numa fase inicial de preprocessamento, tal como fizemos para reordenar as linhas da matriz A.

Inspirando-se na idéia de utilizar algoritmos de ponto interior

para resolver problemas de programação inteira, gostaria de sugerir como assunto para pesquisas futuras testes sobre a utilização de um algoritmo de ponto interior, adaptado para resolver um problema quadrático, como método para resolver um problema com variáveis zero-um. Alguns detalhes sobre como transformar um problema com variáveis zero-um num problema quadrático, estão no apêndice 3.

9. BIBLIOGRAFIA

[1] I. Adler, N. Karmarkar, M. G. C. Resende e G. Veiga, "An implementation of Karmarkar's algorithm for linear programming", Report No. ORC 86-8, Oper. Res. Center, Univer. California

[2] I. Adler, N. Karmarkar, M. G. C.Resende e G. Veiga, "Data structures and programming techniques for the implementation of Karmarkar's algorithm", ORSA Journal on Computing 1 (1989) 84-106

[3] R. J. Vanderbei, "Affine-scalling for linear programs with free variables", Mathematical Programming 43 (1989) 31-44

[4] P.E.Gill, W. Murray, M. A. Saunders, J. A. Tomlin and M. H. Wright, "On projected newton barrier methods for linear programming and an equivalence to Karmarkar's projective method", Mathematical Programming 36 (1986) 183-209

[5] V. Chandru, B. S. Kochar, "Exploiting special structures using a variant of Karmarkar's algorithm", Research Memorandum No.86-10, junho 1986, Purdue University

[6] A. R. L. Oliveira, "Métodos de ponto interior em programação Linear - Estudo e Implementação", tese de mestrado FEE - UNICAMP - 1989

[7] R. J. Vanderbei, M. S. Meketon, B. A. Freedman, "A modification of Karmarkar's linear programming algorithm", Algorithmica 1 (1986) 395-407

[8] N. Karmarkar, "A new polynomial-time algorithm for linear programming", Combinatorica 4 (4) (1984) 373-395

[9] E. R. Barnes, "A variation on Karmarkar's algorithm for solving linear programming problems", Mathematical Programming 36 (1986) 174-182

[10] T. M. Cavalier, K. C. Schall, "Implementing an affine scaling algorithm for linear programming", Comput.Opns.Res. Vol 14, No.5 pp. 341-347, 1987

[11] N. Megiddo, "New Approaches to Linear Programming", Mathematical Programming 36 (1986) 174-182

[12] I. J. Lustig, "A pratical approach to Karmarkar's algorithm", Systems Optimization Laboratory, Stanford University, Technical Report SOL 85-5 Junho 1985

[13] M. Benichou, J. M. Gauthier, P.Girodet, G.Hentges, G.Ribiere, O.Vincent, "Experiments in mixed-integer linear programming", Mathematical Programming 1 (1971) 76-94

[14] O. K. Gupta, A. Ravindran, "Branch and bound experiments in convex nonlinear integer programming", Management Science Vol.31, No.12, Dezembro 1985 pp.1533-1546

[15] A. D. Banchi, C. R. Penteado, "Planejamento otimizado do sistema motomecanizado", IV Seminário de Tecnologia Agronômica - Copersucar - SP pp.515-530

[16] A. D. Banchi, J.R.Lopes, M.C.Machado, R.S.A.Pinto, "Sistema de gerenciamento para mecanização agricola", V Seminário de Tecnologia Agronômica - Copersucar - SP

[17] A. D. Banchi, "Planejamento da utilização de uma frota de máquinas agricolas em exploração policultural, determinando a solução de mínimo custo com auxílio de programação linear", Tese de Mestrado - FEAgr - UNICAMP - 1989

[18] A. Danok, B. McCarl, T. K. White, "Machinery selection and crop planning on a state farm in Iraq", Americam Journal of Agricultural Economics 60 (13) 544-549 (1978)

[19] A. Danok, B. A. McCarl, T. K. White, "Machinery selection modeling: Incorporation of weather variability", Americam Journal of Agricultural Economics 62 700-708 (1980)

[20] I. S. Duff, A. M. Erisman, J. K. Reid, "Direct methods for sparse matrices" (Clarendon Press, Oxford, 1986)

[21] M. Carvalho, "On the minimization of work needed to factor a symmetric positive definite matriz", Final Project

[22] J. H. Wilkinson, "Error analysis of floating-point computation", Numerische Mathematik 2, 319-340 (1960)

[23] J. H. Wilkinson, "Error analysis of direct methods of matrix inversion", J. ACM 8,281-330 (1961)

[24] Luenberger, David, G., "Linear and nonlinear programming", Addison - Wesley (1984)

[25] H. M. Salkin, "Integer programming", Addison - Wesley (1974)

[26] L. S. Lasdon, "Optimization Theory for Large Systems", The Macmillan Company 1970

[27] Murty, Katta G., 1936- "Liner Programming", John Wiley & Sons, Inc 1983

APÊNDICES :

1. FORMATO DOS ARQUIVOS DE DADOS LIDOS PELOS PROGRAMAS DESTE TRABALHO

O problema a ser resolvido deve ser montado num arquivo do tipo texto com o formato descrito a seguir.

Cada linha consiste de três parâmetros, normalmente associados à cada elemento não nulo da matriz A, que chamaremos de pi, pj e px. Eles significam respectivamente o número da linha onde está o elemento, o número da coluna e o próprio elemento.

Os casos especiais são :

a) pi = -1

Neste caso px é um componente, não nulo, do vetor b. O parâmetro pj indica qual o componente.

Default para b : 0

b) pi = -2

Neste caso px é um componente, não nulo, do vetor c. O parâmetro pj indica qual o compontente.

Default para c : 0

c) pi = -3

Neste caso px é um componente, não nulo, do vetor l (limites inferiores). O parâmetro pj indica qual o componente.

Default para l : 0

d) pi = -4

Neste caso px é um componente, não nulo, do vetor u (limites superiores). O parâmetro pj indica qual o componente.

Default para u : -1

e) pi = -5

Neste caso pj é um componente do vetor z (cada componente igual a 1 se a variável respectiva deve ser tratada como inteira e zero caso contrário). O parâmetro pj indica qual o componente e o valor de px não importa mas deve estar presente. A presença de cada linha neste formato basta para indicar que uma dada variável deve ser tratada como inteira.

Default para z : 0

f) pi = -6

Neste caso px é o valor de eps que desejamos utilizar, pj é desprezado mas deve estar presente.

Default para eps : 0.001

g) pi = -7

Neste caso px é o valor de γ que desejamos utilizar, pj é desprezado mas deve estar presente.

Default para γ : 0.95

Observações :

Na leitura de dados, qualquer elemento da matriz cujo valor absoluto for menor do que 10^{-8} é desprezado. Os dados podem estar em qualquer ordem e não é necessário indicar parâmetros como o numero de equações e variáveis. São feitos teste de consistencia e toda a memória é alocada dinamicamente.

O problema primal deve estar na forma padrão, ou seja, minimização da função objetivo, restrições de igualdade e não negatividade.

2. DETERMINAÇÃO DA PRECISÃO NUMERICA DE UM COMPUTADOR

Os computadores que utilizamos efetuam cálculos com precisão precisão dupla. Isso que dizer que o significado de uma dada variável atinge somente até aproximadamente o 16° digito. Na verdade essa precisão depende da forma de representação de números utilizada pelo compilador ou pelo hardware do computador. Seu valor exato pode ser obtido com o seguinte programa simples em linguagem C :

```
#include <stdio.h>
double eps = 1.0;
main()
{
     while ((1.0 + eps) > 1.0) eps /= 2.0;
     eps *= 2.0;
     printf("Eps = %e\n", eps);
}
```

O valor absoluto do expoente de eps assim obtido, fornece o digito mais significativo.

3. TRANSFORMAÇÃO DE UM PROBLEMA COM VARIAVEIS ZERO-UM EM UM PROBLEMA QUADRÁTICO

Sejam M um número positivo grande e $\underline{e}$ um vetor de um's. Considere o seguinte problema com variáveis zero-um (as dimensões de matrizes e vetores estão corretas),

$$\begin{array}{lll} (PZ) & \text{maximize} & c^t.x \\ & \text{sujeito à} & A.x \leq b \\ & & 0 \leq x \leq e \qquad x \text{ inteiro} \end{array}$$

podemos transformá-lo no seguinte problema quadrático :

$$\begin{array}{lll} (PQ) & \text{maximize} & c^t.x - M.x^t.(e - x) \\ & \text{sujeito à} & A.x \leq b \\ & & 0 \leq x \leq e \end{array}$$

O problema (PZ) pode ser solucionado resolvendo-se (PQ) (através de um algoritmo de ponto interior adaptado para o caso quadrático) devido às seguintes propriedades :

(i) Se x^* é uma solução ótima de (PQ) e x^* é inteiro, então é também ótimo de (PZ).

Justificativa :

Como x^* é inteiro, o termo $M.x^t.(e - x)$ na função objetivo de (PQ) se anula e portanto (PQ) se torna igual a (PZ).

(ii) Se x^* é uma solução ótima de (PQ) e x^* não tem todas as componentes inteiras, então (PZ) não tem solução inteira factível.

Justificativa :

Como x^* não tem todas as componentes inteiras, temos que : $c^t.x^* - M.(x^*)^t.(e - x^*) < c^t.x^*$ e portanto x^* é uma solução que maximiza $c^t.x^*$, porém não é inteira.

(iii) Se (PQ) não tem nenhuma solução factível então (PZ) não tem nenhuma solução factível.

Justificativa :

A menos da integralidade, as restrições de (PZ) e (PQ) são as mesmas. Como (PZ) restringe mais o espaço de soluções, se (PQ) não tem solução factível, (PZ) também não tem.

Obs.: Lembramos que é possivel transformar um problema com variáveis inteiras limitadas em um problema com variáveis zero-um. O preço a ser pago é um aumento no número de variáveis do problema.