FERNANDO AUGUSTO SILVA MARINS

ENGENHEIRO MECÂNICO, FACULDADE DE ENGENHARIA DE GUARATINGUETÁ-UNESP, 1976

MESTRE EM CIÊNCIAS, INSTITUTO TECNOLÓGICO DE AERONÁUTICA, 1981

ESTUDOS DE PROGRAMAS EM REDES LINEARES POR PARTES

ORIENTADOR : PROF. DR. CLOVIS PERIN FILHO

Trabalho apresentado à Comissão de Pós-Graduação da Faculdade de Engenharia Elétrica, como parte dos requisitos para a obtenção do título de Doutor em Engenharia Elétrica.

CAMPINAS, 1987

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELETRICA
DEPARTAMENTO DE ENGENHARIA DE SISTEMAS

ESTUDOS DE PROGRAMAS EM REDES LINEARES POR PARTES

Este exemplar corresponde à redação final da tese defendida por Fernando Augusto Silva Marins e aprovada pela Comissão Julgadora em 18/12/87

Trabalho apresentado à Comissão de Pós-Graduação da Faculdade de Engenharia Elétrica, como parte dos requisitos para a obtenção do título de Doutor em Engenharia Elétrica.

CAMPINAS, 1987

A meus filhos Lívia e Fábio e particularmente a minha esposa Maria Luísa, que me proporcionaram a estrutura necessária para a elaboração deste trabalho.

AGRADECIMENTOS

Transmito, por este meio, meus sinceros agradecimentos:

ao professor *Clovis Perin Filho*, pela orientação e esforço constante durante o desenvolvimento deste trabalho;

aos *colegas e amigos* do Departamento de Produção da Faculdade de Engenharia de Guaratinguetá, que assumiram cargas didáticas adicionais durante meu afastamento da FEG para elaborar este trabalho;

aos *amigos* pós-graduandos da FEC, FEE e IMECC, pelo apoio e incentivo dados nos momentos difíceis passados;

à *Margarida Corrêa Leite* pela compreensão e dedicação demonstradas durante o excelente trabalho de datilografia executado.

A todos, que direta ou indiretamente, contribuiram na elaboração deste trabalho.

RESUMO

Este trabalho propõe um refinamento do método simplex especializado para Programas em Redes Lineares por Partes, denominado MSFV. Este refinamento é uma extensão do conceito de bases fortemente viáveis para Programas em Redes, desenvolvido por W.H. Cunningham. A viabilidade forte é mantida por meio de uma regra de saída específica, para escolha da variável básica que deve deixar a base em cada iteração do simplex. Prova-se que, o uso de viabilidade forte em conjunto com regras de entrada adequadas, evita os fenômenos de ciclagem ("cycling") e de empacamento ("stalling").

Além disto são apresentados resultados computacionais testando o MSFV combinado com várias regras de entrada. Adicionalmente, é realizada uma investigação do desempenho do MSFV incorporando a Técnica de Mudança de Escala, proposta por Edmonds e Karp.

ABSTRACT

This work proposes a refinement of the simplex method especialized for solving Piecewise-Linear Network Programs, named MSFV. Such a refinement is an extension of the strongly feasible bases concept for Network Programs, developed by W.H. Cunningham. Strongly feasibility is preserved by a specific leaving variable selection rule at each simplex iteration. It is proved that the use of strong feasibility together with adequate entering variable selection rules prevents two phenomena cycling (ciclic sequence of degenerate iterations) and stalling (exponentially long sequence of degenerated iterations).

Moreover it is reported a computational testing of MSFV linked with several entering variable selection rules. In addition, it is investigated the performance of MSFV with the Scaling Technique, proposed by Edmonds and Karp.

INDICE DA MATÉRIA

CAPITULO 1 - PRELIMINARES

CAPITULO 2 - PROGRAMAS EM REDES LINEARES POR PARTES

CAPITULO 3 - ANÁLISE COMPUTACIONAL

ÍNDICE DAS FIGURAS

ÍNDICE DAS TABELAS

CAPITULO 1

PRELIMINARES

1.1. INTRODUÇÃO

Neste trabalho objetiva-se estudar a Programação em Redes Convexas Lineares por Partes, desenvolver algoritmos especializados e obter resultados computacionais relevantes.

De modo a melhor situar esta área de pesquisa escolhida, inicialmente procura-se, apresentar a Programação Linear por Partes em geral, efetuar uma revisão de alguns dos principais trabalhos publicados e oferecer uma descrição de suas aplicações.

1.1.1. Origens e Revisão da Literatura em Programação Linear por Partes

Em geral a Programação Linear por Partes (PLP) consiste em minimizar uma função objetivo, separável em funções lineares por partes, respeitando um conjunto de restrições lineares. Desta forma, a função objetivo é expressa por uma soma de funções lineares por partes, onde cada parcela é função de apenas uma variável do problema.

A PLP pode ser enfocada por três maneiras, que acabam por oferecer critérios de classificação para as pesquisas desenvolvidas.

No primeiro enfoque a função objetivo satisfaz as condições de convexidade de modo que um problema em PLP pode ser reformulado como um problema em Programação Linear (PL). Várias transformações com este fim tem sido propostas, podendo ser destacadas as propostas por Charnes e Lemke /1/ (1954), Dantzig /2/ (1956), Dantzig, Johnson e White /3/ (1959) e por Ho /4/ (1985).

Todas estas transformações produzem um aumento no número original de variáveis e restrições, pois atuam definindo ao menos uma variável e/ou restrição adicional para cada termo linear ("linear piece") de todas as parcelas (lineares por partes) que compõem a função objetivo original. Evidentemente cada método leva a uma formulação diferente em PL mas que são equivalentes, no sentido de fornecerem a mesma solução ótima.

Ocorre que estes problemas lineares equivalentes possuem estrutura e propriedades especiais que os diferenciam de um problema linear geral de mesmas dimensões. Alguns pesquisadores têm se concentrado em elaborar procedimentos, com características idênticas às do simplex, que explorem estas particularidades da formulação equivalente em PL.

Nesta linha pode-se citar os trabalhos recentes de Snyder /5/ (1981), /6/ (1984) e Premoli /7/ (1986).

Snyder apresenta um algoritmo, baseado em condições sobre a geometria dos problemas lineares por partes originais, que divide as variáveis em conjuntos ordenados especiais ("special ordered sets") utilizados no controle do teste da razão ("pivoting"), realizado pelo seu algoritmo, de forma a reduzir o número de mudanças de base (que ocorreriam no método simplex).

O algoritmo proposto por Premoli é uma generalização de métodos empregados na minimização de funções semi-lineares, tais como as somatórias dos valores absolutos (ponderados) de variáveis, comuns nos problemas de Estimação ℓ_1 e Programação por Metas ("Goal Programming"). Este autor estende estes métodos obtendo um procedimento que se aplica a problemas gerais de Programação Convexa Linear por Partes, sendo recomendado particularmente, àqueles em que o número de variáveis é ligeiramente superior ao número de restrições e onde há um número significante de termos lineares em cada parcela linear por partes da função objetivo.

Um segundo enfoque para a PLP consiste em tratá-la como um caso especial da Programação Não-Linear (PNL). Já foram desenvolvidos métodos de minimização de funções convexas não-diferenciáveis, sujeitas a restrições lineares, que podem ser especializados para aproveitarem a forma simples das funções lineares por partes.

Nesta categoria se enquadram os Métodos de Descida para Otimização Monotrópica ("Descent Methods for Monotropic Optimization"), que tem sido considerados, principalmente por Rockafellar /8/ (1984), no desenvolvimento de métodos para a PLP.

Também os métodos de Gradiente Reduzido ("Reduced-Gradient Methods") para otimização com restrições, podem ser adaptados para abordar a PLP; Snyder /5/ faz referência a estes métodos ao propor um algoritmo tipo simplex para problemas lineares por partes.

Alguns autores, como Conn /9/ (1976) e Bartels /10/ (1980), apresentam algoritmos para resolver problemas lineares (Estimação ℓ_1) que usam uma penalidade ("penalty term") linear por partes.

Finalmente, o terceiro enfoque trata a PLP como uma generalização da PL. Com este enfoque, adotado no presente trabalho, objetiva-se desenvolver uma teoria própria para a PLP, estendendo a teoria de PL.

Um dos primeiros trabalhos neste sentido é o de Golstein /11/ (1960), onde é apresentada uma generalização do método simplex para problemas convexos lineares por partes, com restrições lineares de igualdade e canalização nas variáveis. Também é discutido o critério de otimalidade e as vantagens sobre o método simplex comum.

Ainda na mesma década, Orden e Nalbandian /12/ (1968) propuseram um algoritmo para problemas do mesmo tipo; através da implementação do método e vários testes computacionais os autores mostram as vantagens do procedimento proposto sobre a alternativa de aplicar o simplex ao problema linear equivalente.

Ao final dos anos 60, um grupo de pesquisadores da Monash University (Austrália), onde se destaca Rosvany /13/ (1970), /14/ (1971), desenvolveu uma série de estudos sobre Programação Côncava, particularmente para funções lineares por partes, aplicada a cálculos estruturais (Charret /15/ (1970)).

Um dos primeiros textos didáticos sobre o assunto foi publicado no capítulo 7 do livro de Golstein e Youdine /16/(1973); estes autores formalizam uma teoria e descrevem detalhadamente um método de resolução em PLP.

No Brasil, pelo que pôde ser verificado, o esforço nesta área ficou limitado a diversas teses apresentadas na UNICAMP, no período de 1977 a 1979, por pesquisadores da Faculdade de Engenharia Elétrica. Nestes trabalhos foram propostos, para PLP geral,

métodos de resolução primal (Souza /17/ (1977)), de resolução dual (Garcia /18/ (1978)), de resolução primal-dual (Ferreira /19/ (1979)), e para PLP com estrutura bloco-angular, um método de resolução primal (Ribeiro /20/ (1980)) e outro método de resolução dual (Fernandes /21/ (1979).

Recentemente, Fourer /22/ (1985), /23/ (1985),/24/(1986) publicou seus trabalhos, que foram de muita valia para esta tese, onde faz uma abordagem geral e completa da PLP.

No primeiro artigo, são introduzidos os conceitos de funções convexas lineares por partes separáveis, de dualidade (linear por partes) e programas convexos lineares por partes, estabelecendo suas propriedades de maneira rigorosa. Além disto, é definida solução básica para a PLP, e é descrito um algoritmo, mostrando como se dá a evolução de uma solução básica para outra e sendo apresentada a demonstração de que este sempre encontra uma solução ótima (se existir).

No segundo artigo, Fourer apresenta resultados sobre as hipóteses envolvidas com a parada do algoritmo ("finitess"), viabilidade e degenerescência. Na sequência é feita uma análise computacional comparativa do algoritmo proposto versus o método simplex com variáveis canalizadas ("bounded-variable simplex algorithm"), e é citada uma série de aplicações da PLP.

1.1.2. Aplicações da Programação Linear por Partes

Após esta explanação global do que trata e como se situa a PLP, passa-se a descrever, brevemente, algumas de suas apli

cações mais conhecidas. Maiores detalhes e referências podem ser encontrados em Fourer /25/.

Uma das aplicações mais comuns da PLP é determinar a primeira solução viável e/ou ótima em PL, através do uso de penalidades lineares por partes. Deve-se citar também situações em que há atividades reversíveis ("reversible activities") em programas lineares, que podem ser modeladas por termos lineares por partes.

Outra aplicação da PLP é encontrada em problemas de Estimação ℓ_1. Procura-se determinar a solução aproximada mais acurada, para um sistema de equações lineares Ax=b, em termos da norma ℓ_1. Com a introdução de variáveis de folga (s = b-Ax) transforma-se este problema na otimização de uma função linear por partes, dependente das variáveis s_i, com inclinações -1 se $s_i < 0$ e +1 se $s_i \geq 0$ e sujeita a restrições lineares. Neste caso a PLP também é denominada Programação Semi-Linear.

Também a Programação por Metas pode ser transformada em Programação Semi-Linear. Dado um problema onde se deseja minimizar uma função ponderada (com pesos w_i^+ e w_i^-) das variáveis de desvio (s_i^+ e s_i^-), associadas a variação do sistema Ax com relação ao segundo membro b, respeitadas restrições adicionais sobre as variáveis x; pode-se formulá-lo com uma função objetivo separável semi-linear cujas parcelas relativas às variáveis de desvio s_i tem inclinação w_i^+ para $s_i < 0$ e w_i^- para $s_i \geq 0$, e cujas parcelas relativas às variáveis x_j, tem inclinações $-\infty$ para $x_j < 0$ e 0 para $x_j \geq 0$.

Há vários exemplos de uso da PLP em programas de Programação Linear com Variáveis e Restrições Canalizadas ("Interval

Programming"), onde existem limitantes para as variáveis, $\ell \leq x \leq u$, e o mesmo ocorrendo para as restrições, $b^- \leq Ax \leq b^+$, quando da otimização de uma função linear em x. Ocorre que, pela teoria da Dualidade da PLP, há um programa semi-linear que é o dual do programa acima. Neste programa dual a função objetivo é formada por parcelas lineares por partes associadas às variáveis duais y e z, com inclinações respectivamente de, b_i^- se $y_i < 0$ e b_i^+ se $y_i \geq 0$, e ℓ_j se $z_j < 0$ e u_j se $z_i \geq 0$.

Pode-se ainda citar aplicações com respeito a PNL sujeita a restrições lineares com função objetivo separável em funções convexas. Cada uma das parcelas da função objetivo é aproximada por uma função linear por partes. Uma vez obtida uma solução ótima para o problema aproximado, é possível refinar numa vizinhança desta solução ótima os intervalos utilizados para a construção das funções lineares por partes, de modo a obter uma aproximação melhor da função objetivo original.

Ainda nesta linha há as aproximações semi-lineares sucessivas, onde, ao invés de aproximar a função objetivo globalmente, como na anterior, se faz aproximações locais numa vizinhança da última solução tentativa ("trial solution") obtida.

Finalmente, este mesmo procedimento pode ser aplicado em PNL com restrições não-lineares. Nestes casos há a adição, na função objetivo de cada problema aproximado, de uma função penalidade semi-linear associada as inviabilidades que surgem devido as aproximações efetuadas nas restrições.

A seguir é introduzida a Programação em Redes Convexas Lineares por Partes, sendo destacados trabalhos importantes, abordando a Programação em Redes Lineares, que foram úteis nesta tese.

1.1.3. Programação em Redes Convexas Lineares por Partes

Neste trabalho o objeto de estudo foi uma situação particular da PLP, de muito interesse prático, que ainda não foi explorada convenientemente; é o caso de programas em redes convexas com critério (função objetivo) linear por partes sujeitos a restrições lineares.

De fato, são inúmeras as aplicações envolvendo modelos de otimização em redes, onde se deseja minimizar uma função dos fluxos (variáveis) que circulam pelos arcos das redes, podendo ser citados, entre outros, os seguintes casos em que estes modelos de fluxo em redes são úteis:

* sistemas de produção e distribuição de bens;
* sistemas de alocação de facilidades;
* sistemas de comunicações;
* sistemas de redes elétricas;
* sistemas de tráfego urbano.

Nas últimas décadas tem se estudado estensivamente programas em redes lineares; isto resultou numa série de métodos de resolução especializados para explorar as peculiaridades da matriz de coeficientes das variáveis nas restrições, que é uma matriz de incidência nó x arco, de forma a propiciar uma maior rapidez na resolução de problemas de grande porte em redes.

Dentre os diversos trabalhos desenvolvidos com esta orientação podem ser destacados os que se seguem. O trabalho de Glover, Karney e Klingman /25/(1974) sobre implementações e comparações de pacotes computacionais relativos aos métodos primal,

dual e primal-dual para programas em redes lineares, isto é, problemas de fluxo com custo mínimo em redes lineares ("Minimum Cost Network Flow Problems"); o artigo de Barr, Glover e Klingman /26/ (1977) em que propõem um algoritmo simplex para Problemas de Designação ("Assignment Problems") que evita o fenômeno de ciclagem e explora a degenerescência destes problemas; o trabalho de Barr, Elam, Glover e Klingman /27/ (1977) em que apresentam um algoritmo eficiente (extensão do anterior) para Problemas de Transbordo Canalizado ("Capacitated Transshipment Problems") que evita o fenômeno de ciclagem ("cycling"); o artigo de Bradley, Brown e Graves /28/ (1977) em que fornecem uma descrição completa do projeto, implementação e uso de uma família de pacotes computacionais eficientes para programas em redes lineares de grande porte; nos estudos desenvolvidos por Cunningham /29/(1977), /30/(1979), Cunningham e Klincewicz /31/(1983), apresenta-se um algoritmo simplex para redes lineares e suas propriedades com respeito aos fenômenos de ciclagem e de empacamento ("stalling"), sendo este último fenômeno tão danoso para o desempenho do simplex quanto o primeiro, apesar de ser pouco conhecido; estes trabalhos foram de particular importância nesta dissertação.

Nestes artigos ainda são apresentadas regras de escolha da variável a entrar na solução básica que são próprias para redes, além de outras regras de caráter mais geral, inclusive uma regra especial devida a Bland /32/(1977) que tem como característica principal evitar a ciclagem.

Dentre a vasta literatura didática sobre otimização em redes, cumpre citar o livro de Kennington e Helgason /33/ (1980), onde se apresenta um enfoque completo e unificado de vários aspec

tos relevantes na área incluindo: modelos de fluxo com custo mínimo em redes lineares, modelos de fluxo com custo mínimo em redes lineares generalizadas (com ganhos e perdas), modelos de multifluxo com custo mínimo em redes lineares, modelos de fluxo com restrições adicionais em redes lineares e modelos de fluxo com custo convexo em redes.

Uma grande virtude deste texto é o destaque dado a implementação de algoritmos em redes, a partir de estruturas de dados próprias, expostas no apêndice B da referência /33/, que também podem ser encontradas no artigo de Ali, Helgason, Kennington e Lall /34/(1978) onde se apresenta o estado-da-arte das técnicas eficientes de implementação de algoritmos simplex em redes.

Uma outra linha de pesquisa, com respeito ao aumento da eficiência de algoritmos para problemas de fluxo em redes lineares, foi iniciada por Edmonds e Karp /35/(1970), /36/(1972) envolvendo o uso de Técnica de Mudança de Escala ("Scaling") conjuntamente com algoritmos especializados para redes.

No artigo publicado por Gabow /37/(1985) pode-se encontrar para diversos problemas em redes uma comparação, em termos da complexidade, entre o algoritmo mais conhecido disponível na literatura e um algoritmo com a Técnica de Mudança de Escala (TME); pelo o que ali se observa, este refinamento é muito recomendado.

Ainda com respeito a TME, recentemente Ikura e Nemhauser /38/(1986) publicaram suas experiências com esta técnica aplicada com um algoritmo dual-simplex ao Problema do Transporte Simples (em rede bipartite). Estes autores obtêm um algoritmo onde o número de pivoteamentos resultante de seu uso é limitado polino

mialmente pelas dimensões dos dados de entrada. O resultado empírico mais interessante é que se consegue reduzir, com a TME, o tempo computacional de 30% a 50% na resolução de tais redes.

Quanto a Programação em Redes Convexas Lineares por Partes, não foi publicado nenhum artigo tratando especificamente deste assunto nas revistas técnicas disponíveis, que incluem as mais relevantes coleções existentes em Programação Matemática.

Este trabalho procura oferecer contribuições para o desenvolvimento desta importante área de pesquisa.

1.2. OBJETIVOS E JUSTIFICATIVAS

A constatação da ausência de um tratamento formal e de um algoritmo simplex especializado para redes lineares por partes, motivou o desenvolvimento deste trabalho de pesquisa com o objetivo primordial de preencher esta lacuna. Este trabalho foi estendido na direção de obter um algoritmo com propriedades especiais para evitar os fenômenos de ciclagem e empacamento.

Além disto, pretende-se efetuar um estudo comparativo, a partir de implementações de pacotes computacionais, do desempenho de várias regras de entrada conhecidas (tanto de caráter geral como específicas para redes), propiciando assim informações úteis para pesquisadores da área, acerca de qual regra de entrada melhor se comporta perante diferentes configurações de redes lineares por partes.

Objetiva-se, igualmente, fornecer subsídios para uma efetiva comparação entre o algoritmo, proposto no trabalho, aplicado a redes lineares por partes, e o método simplex aplicado a redes lineares equivalentes; espera-se uma vantagem do algoritmo especializado para redes lineares por partes.

Finalmente, pretende-se iniciar uma investigação sobre o comportamento do algoritmo proposto aliado a Técnica de Mudança de Escala, na expectativa de melhorar seu desempenho.

1.3. SUMÁRIO

No capítulo 1 apresenta-se a PLP em geral; é feita uma revisão da literatura e são discutidas suas principais áreas de atuação. A seguir é introduzida a Programação em Redes Convexas Lineares por Partes, dando-se ênfase aos resultados válidos para redes lineares que deverão ser estendidos para abranger também critérios lineares por partes. Apresenta-se, ao final, os objetivos a serem atingidos.

A formulação de problemas em redes lineares por partes é estabelecida no capítulo 2, onde se apresenta toda a notação utilizada e a fundamentação do algoritmo que foi desenvolvido; apresenta-se uma discussão sobre os fenômenos de ciclagem e empacamento, que são evitados pelo algoritmo proposto, em conjunto com regras de entrada especiais e uma regra de saída (escolha da

variável que deixa a solução básica em cada iteração), que é uma extensão daquela apresentada por Cunningham /30/, /31/ e Barr et alli /27/.

Ainda neste capítulo apresenta-se uma descrição completa dos passos do algoritmo, bem como se comenta cada uma de suas características especiais. Por último é introduzido o enfoque da Técnica de Mudança de Escala para redes lineares por partes.

Já no capítulo 3, constam os resultados computacionais obtidos com o algoritmo desenvolvido e as várias regras de entrada implementadas, com e sem a Técnica de Mudança de Escala. São descritos, também, os geradores de redes pseudo-aleatórias e especiais (conhecidas da literatura) utilizados nos diversos testes desenvolvidos, e a estrutura de dados utilizada na implementação do algoritmo simplex especializado para redes lineares por partes.

Descreve-se e comenta-se todos os experimentos realizados visando atingir os objetivos destacados na seção 1.2, com a apresentação de tabelas contendo os resultados computacionais médios, obtidos para várias corridas do algoritmo para cada classe de problemas.

No capítulo 4 são propostas possíveis linhas de pesquisa a partir deste trabalho, compara-se os objetivos iniciais com os resultados conseguidos e efetua-se uma discussão geral sobre a tese.

Finalmente, são apresentadas as referências bibliográficas consideradas de relevância para o desenvolvimento deste trabalho.

CAPITULO 2

PROGRAMAS EM REDES LINEARES POR PARTES

2.1. INTRODUÇÃO

Apresenta-se, aqui, primordialmente uma formulação de Programas em Redes com Critério Separável Linear por Partes Convexas (PRLP) que são o objeto deste trabalho. Além disto, discute-se os conceitos e definições pertinentes objetivando-se o desenvolvimento de um método simplex especializado para PRLP, que tenha características especiais para evitar os fenômenos de ciclagem ("cycling") e empacamento ("stalling").

A preocupação com a possibilidade de ocorrência destes fenômenos vem motivando diversos pesquisadores que têm estudado classes de problemas onde eles são observados (Klee e Minty /39/ (1972), Goldfarb e Sit /40/(1979) para a Programação Linear em geral e Gassner /41/(1964), Zadeh /42/(1973) e Cunningham /30/ (1979) para a Programação em Redes Lineares - PRL), e que têm apresentado refinamentos no método simplex para evitá-los (Bland /32/ (1977), Cunningham /30/, Barr et alli /26/ (1977), Hung /43/(1983) para a PRL e Fourer /22/, /23/(1985/86) para a PRLP).

O principal conceito a ser exposto é o de *vetor básico fortemente viável*, que é uma extensão para a PRLP da definição de

base fortemente viável, introduzida por Cunningham /29/(1976) e com pequena variação, independentemente, por Barr et alli /26/. Implícito nesta definição encontra-se um refinamento do teste da razão para escolha do arco que deve deixar a base (regra de saída) em cada iteração do método simplex. Será provado que o uso deste refinamento, de implementação simples, evita o fenômeno de ciclagem. Além disto, será demonstrado que o seu uso em conjunto com determinadas regras de escolha da variável que deve entrar na base (regras de entrada), a cada iteração, previne também o fenômeno de empacamento.

Este método, aqui denominado Método Simplex Fortemente Viável (MSFV) para a PRLP, será descrito adiante e suas propriedades principais provadas.

Finalmente faz-se considerações acerca de implementações do MSFV aliado à Técnica de Mudança de Escala proposta por Edmonds e Karp /36/(1970).

2.2. *DEFINIÇÕES E NOTAÇÃO*

Programas em Redes Lineares por Partes

Considere uma rede orientada conectada com conjunto de *nós* $N = \{1,2,\ldots, m\}$ e conjunto de *arcos* $A = \{a_1, a_2,\ldots, a_n\}$.

Denote por a_j^t , a_j^h a *cauda* e a *cabeça* do arco $a_j = (a_j^t , a_j^h)$. Suponha que para cada arco a_j é associada uma variável de *fluxo* não-negativa x_j, um vetor de *intervalos de fluxo* $d^j = (d_k^j)$ satisfazendo a condição $0 = d_0^j < d_1^j < \ldots < d_{k_j}^j = \infty$, um vetor de *custos* nos intervalos de fluxo $c^j = (c_k^j)$ satisfazendo a condição $c_1^j < c_2^j < \ldots < c_{k_j}^j$ e uma função de custo separável linear por partes convexa definida por

$$f_j(x_j) = \sum_k \left[c_k^j (d_k^j - d_{k-1}^j) : k = 1,2,\ldots, h_j - 1 \right] + c_{h_j}^j (x_j - d_{h_j - 1}^j),$$

onde h_j é o *intervalo corrente* de a_j e satisfaz $d_{h_j-1}^j \leq x_j \leq d_{h_j}^j$.

Deve-se observar que esta conceituação é bastante flexível e abrange todas as situações de interesse relativas a existência ou não de canalizações na variável de fluxo (figura 2.1), bastando definir convenientemente os vetores d^j e c^j. Além disto, a condição admitida para os intervalos correntes, que aparentemente conduz a uma indefinição quando x_j assume os valores extremos, será explorada adequadamente adiante.

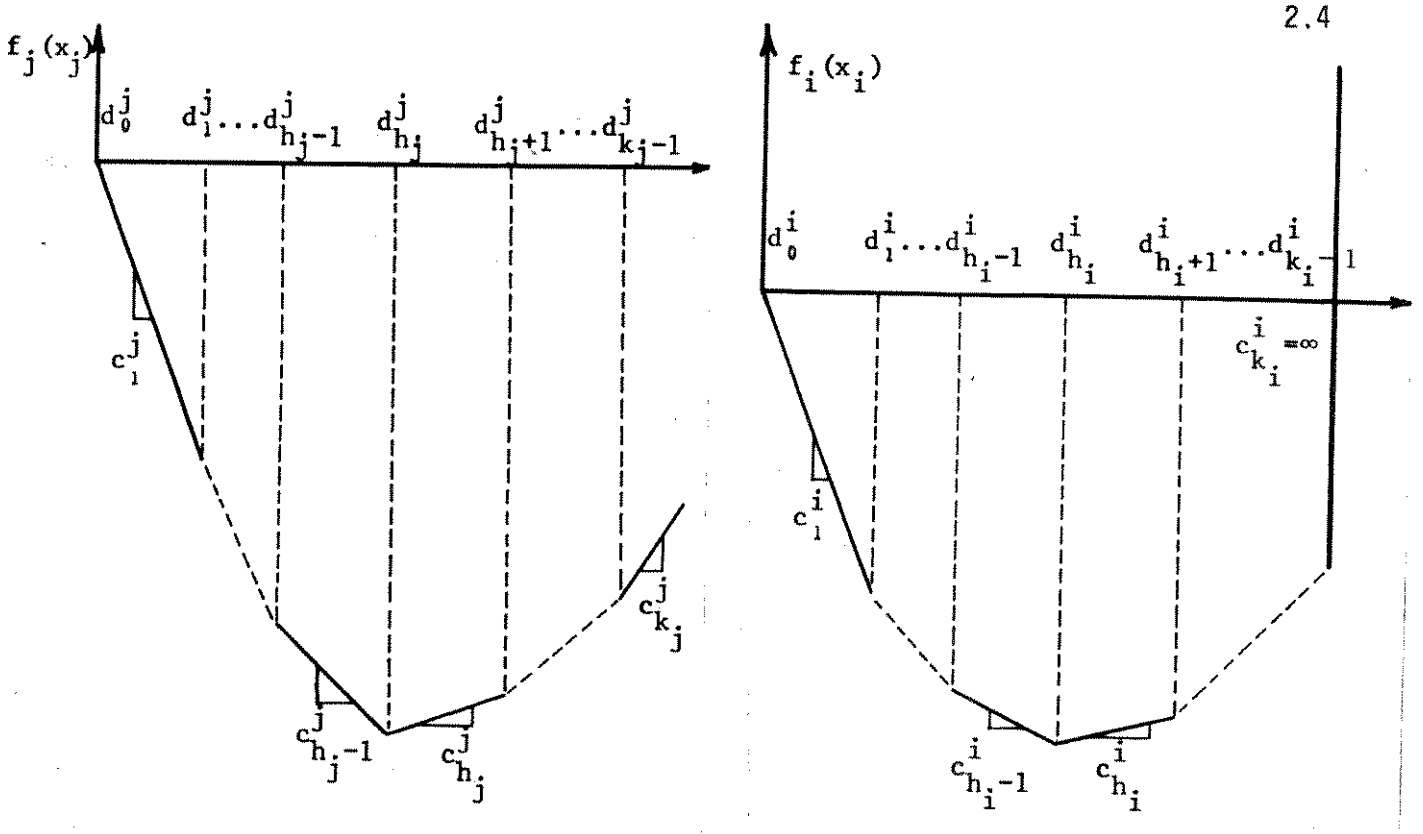


Figura 2.1. Funções de Custo Separável Linear por Partes Convexas para as Variáveis de Fluxo x_j e x_i

Um Programa em Rede Linear por Partes Convexa (PRLP) pode ser estabelecido como sendo a determinação de um vetor $x = (x_j)$, que otimize o problema

$$P : \text{Min}\{f(x) \quad \text{s.a: } Mx = b \ , \ x \geq 0\}$$

onde $f(x) = \sum_j \{f_j(x_j), \ j = 1..n\}$ é a função objetivo separável linear por partes convexa;

$b = b_{mx1} = (b_i)$ é um vetor de demandas nos nós;

$M = M_{mxn} = (M_{ij})$ é a matriz de incidência nó x arco da rede sendo definida por :

$M_{ij} = -1$ se $i = a_j^t$,

$M_{ij} = +1$ se $i = a_j^h$ e

$M_{ij} = 0$ caso contrário.

Observação: De acordo com esta definição da matriz de incidência, pode-se interpretar o vetor de demandas b, com respeito a um problema de transporte, associando a nós geradores de fluxo (nós origens) valores de demanda negativos, associando a nós consumidores de fluxo (nós destinos) valores de demanda positivos e associando a nós de transbordo valores de demanda nulos.

Acerca da função objetivo f(x), sob as condições impostas às sequências de intervalos de fluxo (d_k^j) e de custos nestes intervalos (c_k^j), há uma série de propriedades gerais relatadas no trabalho de Fourer /22/.

Particularmente, aqui é interessante citar que devido ao fato de (d_k^j) e (c_k^j) formarem *sequências crescentes complementares* ("Complementary Increasing Sequences") ao se intercambiar seus papéis em $f_j(x_j)$ obtém-se uma outra função $g_j(z_j)$, também separável linear por partes e convexa, dada por:

$$g_j(z_j) = \sum_k \left[d_k^j \, (c_{k+1}^j - c_k^j) \, : \, k = 1,2,\ldots, h_j - 1 \right] + d_{h_j}^j \left[z_j - c_{h_j}^j \right]$$

com h_j tal que $c_{h_j}^j \leq z_j \leq c_{h_j+1}^j$ (supor $c_{h_j+1}^j = \infty$ se não for definido)

Desta forma $f_j(x_j)$ e $g_j(z_j)$ são denominadas *funções conjugadas*.

Pode-se, neste momento, introduzir uma formulação do *programa dual* do PRLP, que consiste em determinar um par $y = (y_i)$ com $i = 1..m$ e $z = (z_j)$ com $j = 1..n$, denominados respectivamente de *vetor de preços* (também denominados *potenciais*) nos nós e vetor de *custos correntes* (também denominados *inclinações de referência*) nos arcos, que otimiza o problema

$$D : \text{Max } \{yb - g(z) \quad \text{s.a: } yM - z \leq 0,\ z_j \geq c_1^j \quad j = 1..n\}$$

onde $g(z) = \sum_j g_j(z_j)$.

Os teoremas da dualidade (fraca e forte) e seus corolários, válidos para a Programação Linear por Partes em geral, podem ser encontrados na referência Fourer /22/.

Passa-se a apresentar vários conceitos importantes para a PRLP, que serão de utilidade no desenvolvimento a seguir.

Definição: Um vetor de fluxos $x = (x_j)$ $j = 1..n$ é uma *solução primal* do problema P se satisfaz $Mx = b$.

Definição: Um par de vetores $y = (y_i)$ $i = 1..m$ e $z = (z_j)$ $j = 1..n$ é uma *solução dual* do problema P se satisfaz $yM - z \leq 0$.

Definição: Um vetor de fluxos $x = (x_j)$ $j = 1..n$ é uma *solução primal viável* do problema P se satisfaz $Mx = b$ e $x_j \geq 0$ $j = 1..n$.

Definição: Um par de vetores $y = (y_i)$ $i = 1..m$ e $z = (z_j)$ $j = 1..n$ é uma *solução dual viável* do problema P se satisfaz $yM - z \leq 0$, $z_j \geq c_1^j$ $j = 1..n$.

Definição: Um vetor de fluxos $x^* = (x^*_j)$ $j = 1..n$ é uma *solução primal ótima* do problema P se satisfaz $Mx^* = b$, $x^*_j \geq 0$, $j = 1..n$ e $f(x^*) \leq f(x)$ para toda solução primal viável x.

Definição: Um par de vetores $y^* = (y^*_i)$ $i = 1..m$ e $z^* = (z^*_j)$ $j = 1..n$ é uma *solução dual ótima* do problema P se satisfaz em $y^*M - z^* \leq 0$, $z^*_j \geq c^j_1$ $j = 1..n$ e $y^*b - g(z^*) \geq yb - g(z)$ para toda solução dual viável (y, z).

Condições de Folgas Complementares

A partir do teorema 2.1 - Capítulo 7 de Golstein e Youdine /16/, pode-se estabelecer as condições de folgas complementares (CFC) que associam as soluções dos problemas P e D:

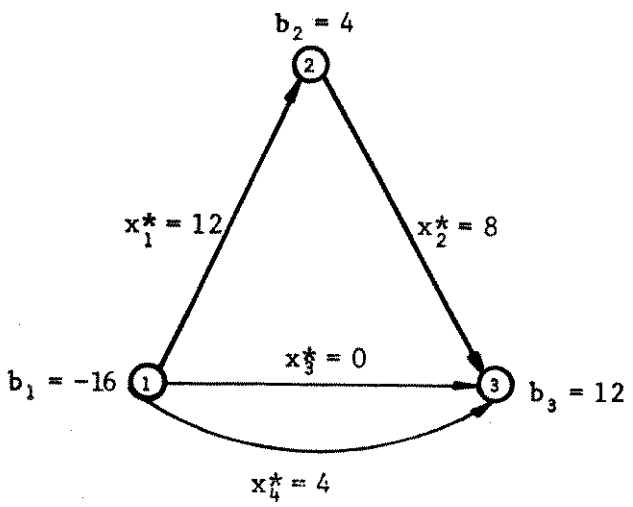
sejam x^* , (y^*, z^*) soluções primal ótima e dual ótima, respectivamente, do problema P, tem-se que

se $d^j_{h_j-1} < x^*_j < d^j_{h_j}$ então $y^*_{h_{a_j}} - y^*_{t_{a_j}} = c^j_{h_j}$;

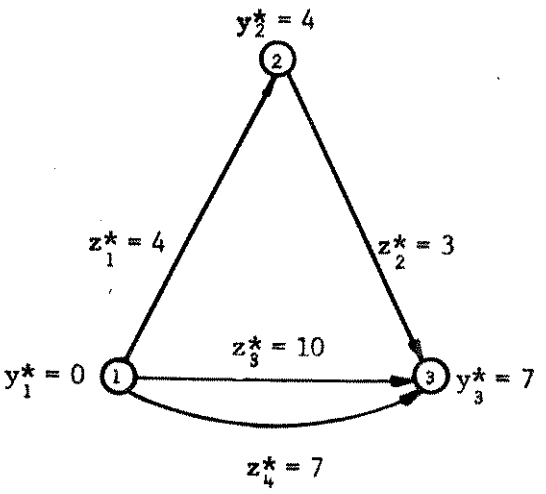
se $c^j_{h_j} < y^*_{h_{a_j}} - y^*_{t_{a_j}} < c^j_{h_j+1}$ então $x^*_j = d^j_{h_j}$;

se $y^*_{h_{a_j}} - y^*_{t_{a_j}} < c^j_1$ então $x^*_j = 0$.

No exemplo a seguir procura-se dar uma visualização destas condições.



(a) solução primal



(b) solução dual

Figura 2.2. Soluções Primal e Dual para o Problema P, relativas ao Exemplo 2.1

Exemplo 2.1. Considere a rede da figura 2.2(a) com os dados adicionais.

$c^1 = (2 \quad 4 \quad \infty)$ $\qquad$ $c^2 = (3 \quad 5 \quad 7)$

$d^1 = (0 \quad 10 \quad 20 \quad \infty)$; $\qquad$ $d^2 = (0 \quad 100 \quad 200 \quad \infty)$;

$c^3 = (10 \quad 20)$ $\qquad$ $c^4 = (5 \quad 9)$

$d^3 = (0 \quad 1 \quad \infty)$; $\qquad$ $d^4 = (0 \quad 4 \quad \infty)$;

$$f_1(x_1) = \begin{cases} 2x_1 & 0 \leq x_1 \leq 10 \\ 20 + 4(x_1 - 10) & 10 \leq x_1 \leq 20 \\ \infty & 20 < x_1 \end{cases} ;$$

$$f_2(x_2) = \begin{cases} 3x_2 & 0 \leq x_2 \leq 100 \\ 300 + 5(x_2 - 100) & 100 \leq x_2 \leq 200 \\ 800 + 7(x_2 - 200) & 200 \leq x_2 \end{cases} ;$$

$$f_3(x_3) = \begin{cases} 10x_3 & 0 \leq x_3 \leq 1 \\ 10 + 20(x_3 - 1) & 1 \leq x_3 \end{cases}$$

$$f_4(x_4) = \begin{cases} 5x_4 & 0 \leq x_4 \leq 4 \\ 20 + 9(x_4 - 4) & 4 \leq x_4 \end{cases}$$

Pode-se estabelecer o problema P na forma:

$$P: \text{Min } \{f(x) = f_1(x_1) + f_2(x_2) + f_3(x_3) + f_4(x_4)\}$$

$$\text{s.a:} \begin{cases} -x_1 - x_3 - x_4 = -16 \\ x_1 - x_2 = 4 \\ x_2 + x_3 + x_4 = 12 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

Conseqüentemente, pode-se escrever o problema D como sendo:

$$D: \text{Max } \{g(y,z) = -16y_1 + 4y_2 + 12y_3 - (g_1(z_1) + g_2(z_2) + g_3(z_3) + g_4(z_4))\}$$

$$\text{s.a:} \begin{cases} -y_1 + y_2 - z_1 \leq 0 & (1) \\ -y_2 + y_3 - z_2 \leq 0 & (2) \\ -y_1 + y_3 - z_3 \leq 0 & (3) \\ -y_1 + y_3 - z_4 \leq 0 & (4) \\ z_1 \geq 2\,,\ z_2 \geq 3\,,\ z_3 \geq 10\,,\ z_4 \geq 5 & (5) \end{cases}$$

$$\text{onde: } g_1(z_1) = \begin{cases} \infty & z_1 < 2 \\ 10(z_1 - 2) & 2 \leq z_1 \leq 4 \\ 20 + 20(z_1 - 4) & 4 \leq z_1 \end{cases} ;$$

$$g_2(z_2) = \begin{cases} \infty & z_2 < 3 \\ 100(z_2 - 3) & 3 \leq z_2 \leq 5 \\ 200 + 200(z_2 - 5) & 5 \leq z_2 \leq 7 \\ \infty & 7 \leq z_2 \end{cases} ;$$

$$g_3(z_3) = \begin{cases} \infty & z_3 < 10 \\ z_3 - 10 & 10 \leq z_3 \leq 20 \\ \infty & 20 \leq z_3 \end{cases} ;$$

$$g_4(z_4) = \begin{cases} \infty & z_4 < 5 \\ 4(z_4 - 5) & 5 \leq z_4 \leq 9 \\ \infty & 9 \leq z_4 \end{cases}$$

Considere a solução primal de P, dada na figura 2.2(a):

$x_1^* = 12$, $x_2^* = 8$, $x_3^* = 0$, $x_4^* = 4$ com $f(x^*) = 72$

Pelas CFC tem-se:

$x_1^* = 12 \implies 10 < x_1^* < 20 \implies y_2^* - y_1^* = 4$, para $y_1^* = 0 \therefore y_2^* = 4$;

$x_2^* = 8 \implies 0 < x_2^* < 100 \implies y_3^* - y_2^* = 3 \therefore y_3^* = 7$;

$x_3^* = 0 \implies y_3^* - y_1^* \leq 10$ (verificada)

$x_4^* = 4 \impliedby 5 \leq y_3^* - y_1^* \leq 9$ (verificada)

A determinação de z^* é feita de modo que a solução (y^*, z^*) seja viável:

de (1) e (5) tem-se $z_1^* \geq y_2^* - y_1^*$ e $z_1^* \geq 2$ $\therefore$ $z_1^* \geq 4$;

de (2) e (5) tem-se $z_2^* \geq y_3^* - y_2^*$ e $z_2^* \geq 3$ $\therefore$ $z_2^* \geq 3$;

de (3) e (5) tem-se $z_3^* \geq y_3^* - y_1^*$ e $z_3^* \geq 10$ $\therefore$ $z_3^* \geq 10$;

de (4) e (5) tem-se $z_4^* \geq y_3^* - y_1^*$ e $z_4^* \geq 5$ $\therefore$ $z_4^* \geq 7$.

Sabe-se, pelo Teorema da Dualidade Forte (ver Fourer /22/), que para x^*, (y^*,z^*) serem soluções ótimas para os problemas P e D, respectivamente, deve-se ter $f(x^*) = g(y^*, z^*)$, dado que x^* é solução primal viável para P e (y^*, z^*) é solução dual viável para D.

Assim, passa-se a determinar quais são os valores de z_1^*, z_2^*, z_3^* e z_4^* que levam a esta situação de otimalidade destas soluções. Pretende-se Max $\{g(y, z)\}$ para tanto, uma vez que yb pode ser determinado para y^*, deve-se ter Min $\{\sum_{j=1}^{n} g_j(z_j)\}$, ou seja:

$\underset{z_1 \geq 4}{\text{Min}}\ g_1(z_1) = g_1(4) = 20 \quad \therefore \quad z_1^* = 4$;

$\underset{z_2 \geq 3}{\text{Min}}\ g_2(z_2) = g_2(3) = 0 \quad \therefore \quad z_2^* = 3$;

$\underset{z_3 \geq 10}{\text{Min}}\ g_3(z_3) = g_3(10) = 0 \quad \therefore \quad z_3^* = 10$;

$\underset{z_4 \geq 7}{\text{Min}}\ g_4(z_4) = g_4(7) = 8 \quad \therefore \quad z_4^* = 7.$ (Observe que $z_j^* = \max\{c_1^j , y^*_{a_j^h} - y^*_{a_j^t}\}$).

Desta forma obtém-se, $g(y^*, z^*) = 100 - 28 = 72 = f(x^*)$, concluindo-se que x^* e (y^*, z^*) são soluções ótimas para os problemas P e D, respectivamente.

Soluções Básicas

Definição: Uma *base* de um PRLP é uma árvore geradora H, isto é, um subconjunto de m-1 arcos, da rede associada ao PRLP, que não forma ciclos. Observe que o posto da matriz de restrições é m-1 e que cada conjunto de m-1 colunas linearmente independentes desta matriz constitue uma árvore geradora.

Definição: Dada uma base de um PRLP, denominam-se *arcos básicos* àqueles arcos da rede pertencentes à árvore geradora H, os demais denominam-se *arcos não-básicos*.

Observação: Na figura 2.2(a) os arcos em destaque (a_1, a_2) são os arcos básicos referentes a uma base associada à solução x^* que compõem uma árvore geradora; os demais (a_3, a_4) são os arcos não-básicos.

Definição: Considere uma árvore geradora H e um nó especial da rede chamado *raiz* da árvore H. Defina por $R(H, a_j)$ o conjunto formado pelos nós da subárvore que contêm a raiz se o arco básico a_j é removido de H.

Definição: Um arco básico $a_j = (a_j^t , a_j^h) \in H$ é denominado *direcionado para a raiz* se $a_j^h \in R(H, a_j)$ e denominado *direcionado da raiz* se $a_j^t \in R(H, a_j)$.

Definição: Para cada nó i, i = 1..m da rede associada ao PRLP existe um único caminho em H conectando a raiz com i.

Seja $P(H, i) = (p_j^i)$ o *vetor indicador deste caminho*, definido por:

$p_j^i = +1$ se $a_j \in H$ é arco direcionado da raiz para i;

$p_j^i = -1$ se $a_j \in H$ é arco direcionado de i para a raiz;

$p_j^i = 0$ se $a_j \in H$ não é arco do caminho da raiz para i.

Exemplo 2.2. Admita a rede da figura 2.3, onde estão destacadas a árvore H e o nó raiz.

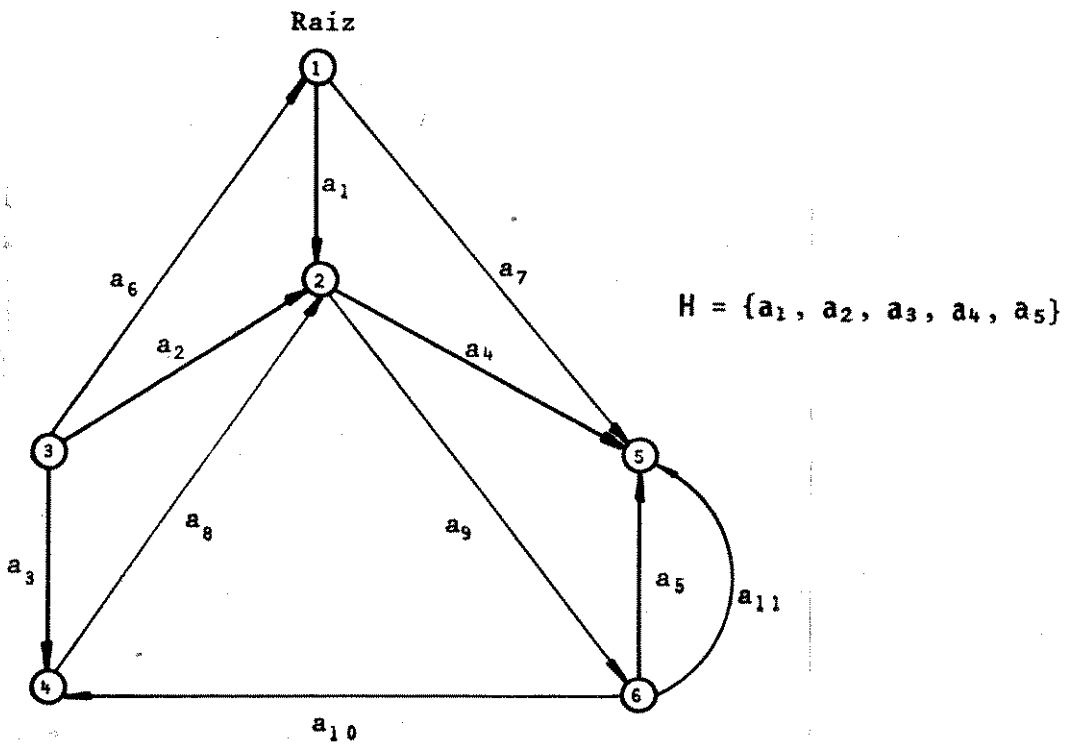


Figura 2.3. Rede e Árvore Geradora para o Exemplo 2.2

Nesta situação tem-se:

$R(H, a_4) = \{1, 2, 3, 4\}$; $R(H, a_3) = \{1, 2, 3, 5, 6\}$;

arcos direcionados para a raiz : a_2 e a_5;

arcos direcionados da raiz : a_1, a_3 e a_4;

vetor indicador do caminho da raiz ao nó 4:

$P(H, 4) = (1 \quad -1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)^T$;

vetor indicador do caminho da raiz ao nó 5:

$P(H, 5) = (1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)^T$.

Defina para cada arco básico $a_j \in H$, os subconjuntos de arcos não-básicos incidentes em nós de $R(H, a_j)$:

$$E(H, a_j) = \{a_k \in A \setminus H : a_k^t \in R(H, a_j) \text{ e } a_k^h \notin R(H, a_j)\};$$

$$\bar{E}(H, a_j) = \{a_k \in A \setminus H : a_k^t \notin R(H, a_j) \text{ e } a_k^h \in R(H, a_j)\}.$$

Para a figura 2.3 tem-se, a título de exemplificação:

$E(H, a_5) = \{a_9\}$; $\bar{E}(H, a_5) = \{a_{10}, a_{11}\}$; $E(H, a_3) = \{a_{10}\}$; $\bar{E}(H, a_3) = \{a_8\}$.

Definição: Seja um vetor de intervalos correntes $h = (h_j)$, associado a uma árvore geradora H, cujas componentes h_j, relativos a $a_j \in H$, não estão fixados. Existe uma solução primal (dual) denominada *solução básica* (x, y, z) correspondente a este vetor h, que pode ser obtida da expressão (1) a seguir.

$$x_j = \begin{cases} d_{h_j - 1} & \text{para} \quad a_j \notin H \\ -\sum_i \left[b_i : i \notin R(H, a_j) \right] + \sum_k \left[x_k : a_k \in E(H, a_j) \right] - \sum_k \left[x_k : a_k \in \bar{E}(H, a_j) \right] & \\ \quad \text{para} \quad a_j \in H \quad \text{e} \quad a_j \quad \text{direcionado para a raiz ;} & \\ \sum_i \left[b_i : i \notin R(H, a_j) \right] - \sum_k \left[x_k : a_k \in E(H, a_j) \right] + \sum_k \left[x_k : a_k \in \bar{E}(H, a_j) \right] & \\ \quad \text{para} \quad a_j \in H \quad \text{e} \quad a_j \quad \text{direcionado da raiz ;} & \end{cases}$$

$\forall a_j \in A$

(1):

$$y_i = \sum_j p_j^i \, c_k^j \quad ; \quad \text{para} \ \{ a_j \in H \quad \text{e} \quad d_{k-1}^j \leq x_j \leq d_k^j \}, \ \forall \ i \in N$$

$$z_j = \begin{cases} c_k^j & \text{para} \quad a_j \in H \ ; \\ \text{Max} \{ c_1^j \ , \ y_{a_j}^h - y_{a_j}^t \} & \text{para} \quad a_j \notin H. \end{cases}$$

$\forall a_j \in A$

Observação: *(a)* O vetor h, assim constituído, é denominado *vetor básico;*

(b) Saliente-se que (x, y, z) calculados por (1) satisfazem as condições de folgas complementares e f(x) = yb - g(z). Particularmente para z_j, ver final do exemplo 1.

Definição: A solução básica (x, y, z) associada a um vetor básico h é denominada *solução básica primal (dual) viável* se são verificadas as restrições do problema P (D). O *vetor básico* h é denominado, neste caso, *primal viável (dual viável).*

Definição: A solução básica primal (dual) viável associada a um vetor básico h é denominada *solução ótima* se as restrições de P e D sao verificadas conjuntamente, e neste caso h é denominado *vetor básico ótimo*.

Definição: Para cada arco não-básico $a_r \notin H$, existe um único ciclo em $H \cup \{a_r\}$; $Q(H, a_r) = (q_j^r)$ é denominado *vetor indicador deste ciclo* sendo que $q_r^r = +1$, $q_j^r = +1$ se $a_j \in H$ e a_j, a_r têm a mesma orientação no ciclo, $q_j^r = -1$ se $a_j \in H$ e a_j, a_r têm orientações contrárias no ciclo, e $q_j^r = 0$ se $a_j \in H$ não é arco do ciclo.

A título de ilustração, considere a rede e a árvore da figura 2.2, admitindo $a_r = a_9$ tem-se que

$$Q(H, a_9) = (0 \quad 0 \quad 0 \quad -1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0)^T.$$

Definição: Para um arco não-básico a_r, com fluxo $x_r = d^r_{h_r - 1}$, define-se o *custo relativo à direita* como sendo $C_r^+ = c^r_{h_r} + y_{a_r^t} - y_{a_r^h}$ e o *custo relativo à esquerda* por $C_r^- = c^r_{h_r - 1} + y_{a_r^t} - y_{a_r^h}$, admitindo $c^r_{h_r - 1} = -\infty$ se não for definido.

Observação: Analogamente ao que ocorre nos Programas em Redes Lineares Canalizadas, se $C_r^+ < 0$ e for aumentado o fluxo x_r, a função objetivo f(x) decrescerá; caso $C_r^- > 0$ e for diminuído o fluxo x_r, f(x) decrescerá também.

Passa-se, adiante, a se definir os fenômenos de ciclagem e empacamento no método Simplex e a estudá-los sob a ótica da PRLP.

2.3. *FENÔMENOS DE CICLAGEM E EMPACAMENTO*

Ciclagem e Empacamento

O método simplex para Programação em Redes Lineares explora as características da matriz dos coeficientes das variáveis nas restrições do problema, que no caso é uma matriz de incidência nó x arco. Por exemplo, não há necessidade, de em cada iteração, trabalhar com a inversa da base explicitamente.

Essencialmente o método simplex trabalha com uma sequência de árvores geradores (básicas), cada uma obtida da anterior por pivoteamento (adição de um arco não-básico simultaneamente à retirada de um arco básico). Em cada iteração procura-se melhorar o valor da solução básica associada, de modo a não permitir a repetição de uma solução básica. Este procedimento é repetido até encontrar-se uma solução básica ótima ou até detectar-se a inexistência de tal solução.

Ocorre que em pivoteamentos degenerados, o valor da solução básica não se altera, de forma que não se pode assegurar que nenhuma solução básica será repetida. De fato, pode ocorrer

o fenômeno de *ciclagem*, com a formação de uma seqüência cíclica de soluções básicas, todas com o mesmo valor, impedindo que uma solução ótima seja alcançada. Exemplos de ciclagem são apresentados em Gassner /41/ e Cunningham /30/.

Existem vários refinamentos do método simplex que eliminam a ciclagem e garantem que o método alcance um ponto terminal (solução ótima, ilimitada ou constatação da inexistência de solução ótima). O mais conhecido é o refinamento lexicográfico(método das perturbações), ver referência Fourer /23/ para a PLP, que atua na regra de saída de variável da base, escolhendo adequadamente uma entre as variáveis candidatas a sair. Outros dois refinamentos são propostos em Bland /32/, ambos atuam simultaneamente nas regras de entrada e saída de variáveis da base.

Um refinamento especial para redes, dos mais eficientes, que atua apenas na regra de saída é proposto por Cunningham /29/ e Barr et alli /26/. Ele previne ciclagem no método simplex trabalhando apenas com *bases fortemente viáveis* associadas a árvores geradoras básicas (viáveis) cujos arcos possuem certos requisitos adicionais de orientação. Este refinamento corresponde a uma regra simples, com características gráfico-teóricas, para decidir, em cada iteração, qual arco deve ser retirado da árvore geradora para a entrada de outro, de modo a manter sucessivamente bases fortemente viáveis.

Para o caso de Programação Linear por Partes, existem poucas publicações, mesmo assim limitadas, que abordam este problema da ciclagem (Fourer /23/), sendo que não há nada específico para a PRLP.

Eliminado o fenômeno de ciclagem, existe a possibilidade do método simplex admitir seqüências de soluções básicas, obtidas através de uma seqüência de pivoteamentos degenerados, cujo comprimento não pode ser limitado por um polinômio no tamanho do problema (número de nós e arcos); isto é mostrado em Cunningham /30/. De fato, isto pode acontecer mesmo quando as regras de Bland /32/ ou as bases fortemente viáveis estiverem sendo utilizadas (ver exemplo 2.4, adiante). Este fenômeno é conhecido como *empacamento* ("stalling") e é uma das preocupações maiores deste trabalho, já que nada foi publicado, com respeito a esse comportamento indesejável do método simplex, relativo a PRLP.

Para evitar o empacamento em redes lineares, Cunningham /30/ apresenta um método simplex que alia a manutenção de bases fortemente viáveis a determinados tipos de regras de entrada (Teorema 3 - Cunningham /30/), que ele demonstra atuarem no sentido de tornarem limitadas polinomialmente, no tamanho do problema (número de nós ou arcos dependendo da regra), as seqüências de pivoteamentos degenerados sucessivos que podem ser geradas pelo método simplex.

Em muitos aspectos ciclagem e empacamento se assemelham, contudo há uma diferença evidente quando se pretende mostrar exemplos de um e outro fenômeno. Um exemplo de ciclagem consiste em expor um problema e uma seqüência cíclica de soluções básicas viáveis degeneradas correspondentes. Já para exemplificar um caso de empacamento se faz necessário apresentar uma família de problemas e correspondentes seqüências de soluções básicas viáveis degeneradas, para que se perceba exatamente a relação não-polinomial,

existente entre o número de iterações até se obter uma solução terminal e as dimensões do problema.

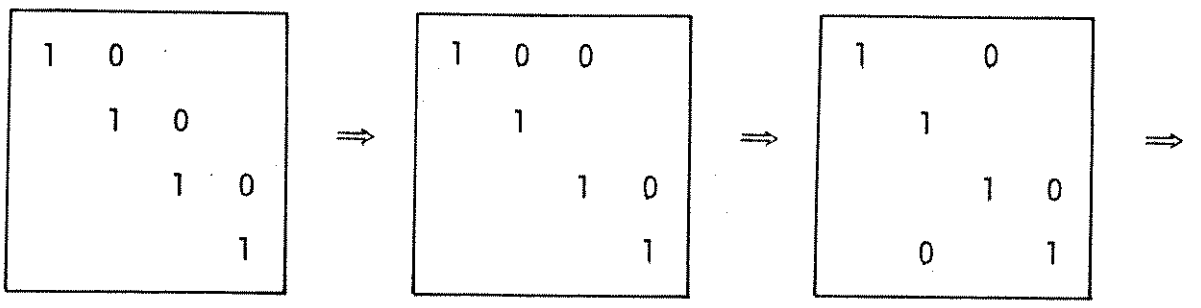
A seguir apresenta-se um exemplo de ciclagem, um exemplo de família de redes onde ocorre empacamento e um exemplo ruim para o método simplex em redes.

Exemplo 2.3. Ciclagem no Problema do Transporte (Gassner /41/)

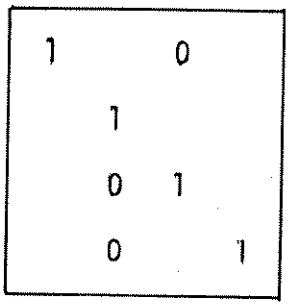
Considere o problema da designação (caso mais degenerado do problema do transporte) com dimensão 4 x 4 e com a seguinte matriz de custos:

$$\begin{array}{c} \\ \text{ORIGENS} \end{array} \begin{array}{c} \text{DESTINOS} \\ \begin{bmatrix} 3 & 5 & 5 & 11 \\ 9 & 7 & 9 & 15 \\ 7 & 7 & 11 & 13 \\ 13 & 13 & 13 & 17 \end{bmatrix} \end{array}$$

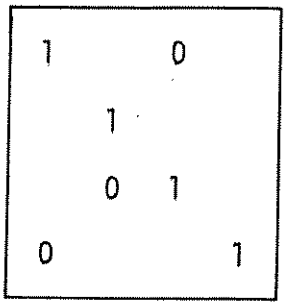
Gassner apresenta como exemplo de ciclagem a seguinte seqüência cíclica de árvores básicas viáveis degeneradas: (são apresentados os valores das variáveis básicas x_{ij}, i, j = 1..4).



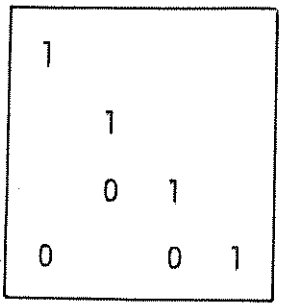
SOLUÇÃO 1 *SOLUÇÃO 2* *SOLUÇÃO 3*



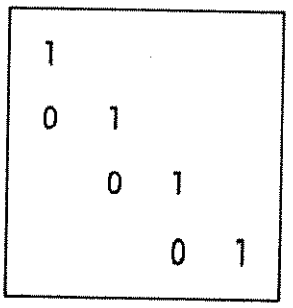
SOLUÇÃO 4



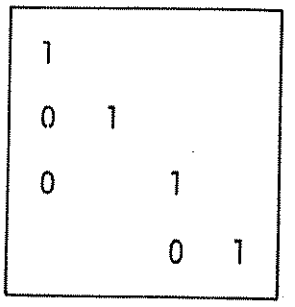
SOLUÇÃO 5



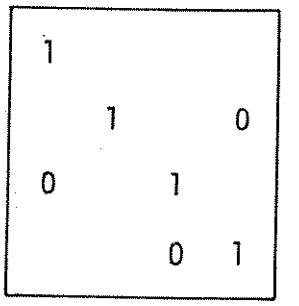
SOLUÇÃO 6



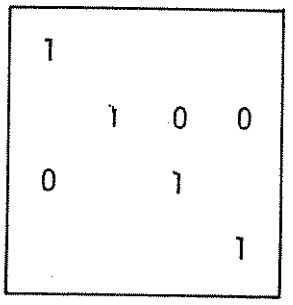
SOLUÇÃO 7



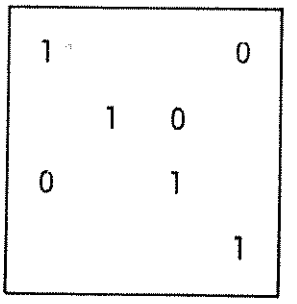
SOLUÇÃO 8



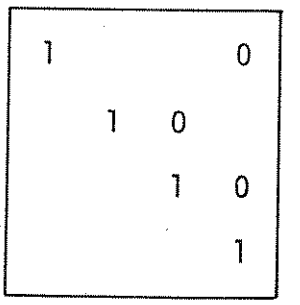
SOLUÇÃO 9



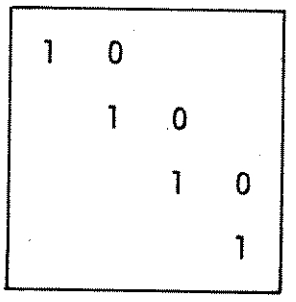
SOLUÇÃO 10



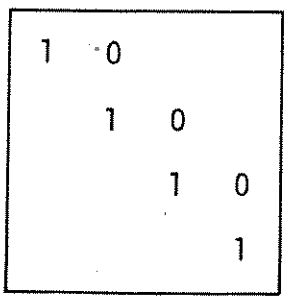
SOLUÇÃO 11



SOLUÇÃO 12



SOLUÇÃO 13



SOLUÇÃO 1 + 12 k

$k = 0, 1, 2, \ldots$

Exemplo 2.4. Empacamento no Problema do Transbordo (Cunningham /30/)

Cunningham considera uma família de problemas de transbordo com $2n$ nós e $4n - 2$ arcos para $n = 2, 3, 4, \ldots$ representados na figura 2.4. Para esta rede tem-se o conjunto de arcos $A = \{a_0, q_0, a_1, b_1, p_1, q_1, \ldots, a_{n-1}, b_{n-1}, p_{n-1}, q_{n-1}\}$, todas as demandas nos nós são nulas, cada arco dos tipos b_i e q_i tem custo 0 e dos tipos a_i e p_i tem custo 2^i.

Considere a árvore fortemente viável H_0 formada pelos arcos $\{a_0, a_1, p_1, \ldots, a_{n-1}, p_{n-1}\}$. Percebe-se que a árvore formada pelos arcos que complementam H com relação a A ($A \setminus H$) é a única árvore ótima (pois tem os menores custos associados).

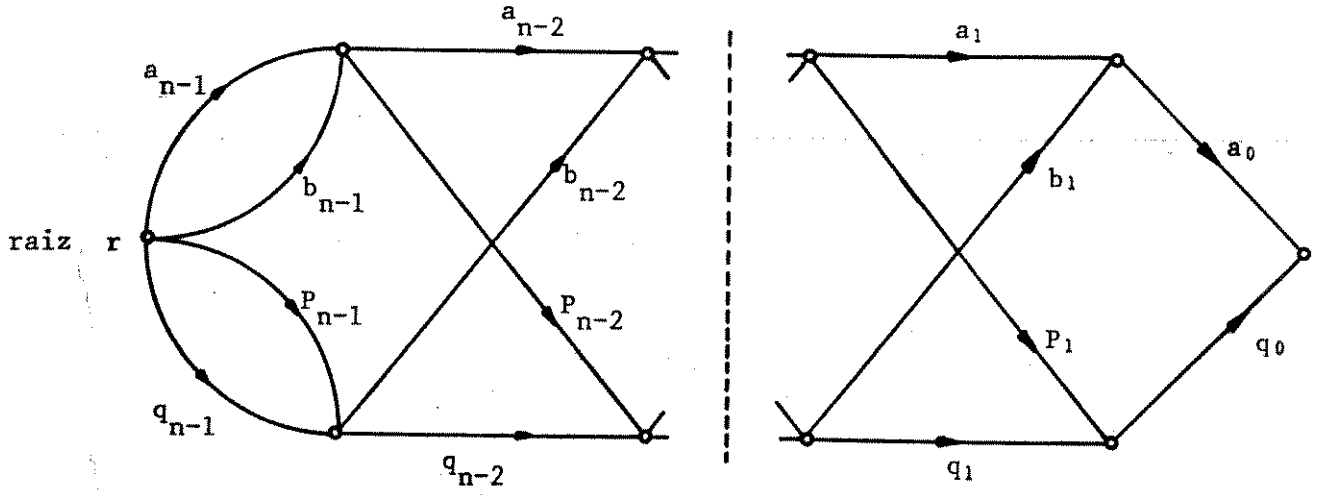


Figura 2.4. Família de Problemas de Transbordo que Apresentam Empacamento

Cunningham demonstra que para problemas desta família, com $n \geq 2$, existe uma sequência (não cíclica) de árvores fortemente viáveis, iniciando com H^0 que tem comprimento superior a 2^n ,

constituindo-se portanto em um exemplo de empacamento. Além disto mostra que a regra proposta em Bland /32/ admite uma seqüência com comprimento $3 \cdot 2^n - 2n - 2$.

No capítulo 3 volta-se a analisar este exemplo sob o ponto de vista da PRLP.

Exemplo 2.5. Redes Ruins para o Método Simplex (Zadeh /42/)

Zadeh apresenta uma família de problemas de transporte com $2n + 2$ nós e $n^2 + n + 3$ arcos de acordo com a figura 2.5. As demandas nos nós s, t, 1, 1', 2, 2' são, respectivamente, 0, 0, -1, 2, -3, 2. Para os nós j, j' (j, j' = 3..n) as demandas são $-(2^{j-1} + 2^{j-3})$ e $(2^{j'-1} + 2^{j'-3})$. Os arcos $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n, q_1, q_2$ têm custo 0. O arco q_0 tem custo ∞. Os arcos do tipo $p_{ij'}$, ligando o nó i ao nó j', tem custo $2^{k-1} - 1$ onde $k = \max \{i, j'\}$. Todos os arcos tem capacidade ∞. Observe que existe um arco $p_{ij'}$ para cada par $i,j' = 1, 2, \ldots, n$ com $i \neq j'$.

A árvore inicial proposta por Zadeh é constituída pelos arcos $q_0, a_1, a_2, \ldots, a_n, b_1, \ldots, b_n$. Ele afirma que o método simplex realiza $2^n + 2^{n-2} - 2$ iterações para obter a solução ótima. Já o Método de Mudança de Escala, proposto por Edmonds e Karp /36/, requer apenas $4n - 7$ iterações; este método é comentado ao final da seção 2.4.

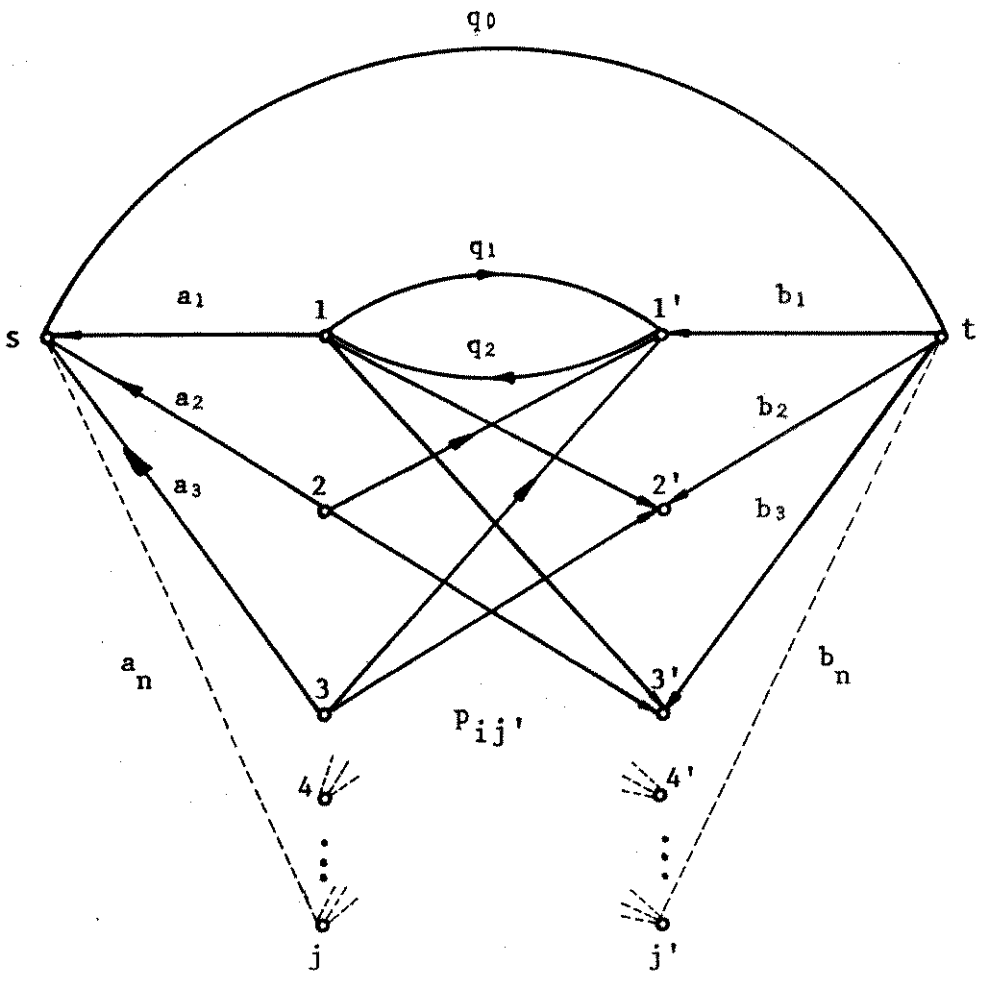


Figura 2.5. Família de Problemas de Transporte Ruins para o Método Simplex

Esta família também será enfocada adiante pela PRLP.

2.4. O MÉTODO SIMPLEX FORTEMENTE VIÁVEL PARA A PRLP

Antes de descrever-se o Método Simplex Fortemente Viável (MSFV) para a PRLP, é apresentado um novo método simplex especializado para a PRLP, possuindo características especiais.

Método Simplex Especializado para PRLP

Passo 0. Seja h um vetor básico de intervalos correntes associado à solução básica primal viável (x, y, z).

Passo 1. Se $C_j^+ \geq 0$ e $C_j^- \leq 0$ para todo arco não-básico $a_j \notin H$ a solução atual (x, y, z) é ótima.

Caso contrário, escolha $a_r \notin H$ tal que $C_r^+ < 0$ ou $C_r^- > 0$.

Se $C_r^+ < 0$ faça $w = +1$ e $\bar{C}_r = C_r^+$.

Se $C_r^- > 0$ faça $w = -1$ e $\bar{C}_r = C_r^-$.

(Notar que $C_r^+ \cdot C_r^- \geq 0$ pois a função objetivo f(x) é convexa).

Passo 2. Seja $Q(H, a_r) = (q_j^r)$ o vetor indicador do ciclo formado em $H \cup \{a_r\}$.

Calcule $v_1 = \text{Min } \{d^j_{h_j} - x_j \ : \ wq_j^r = +1 \ , \ j \neq r\}$

$v_2 = \text{Min } \{x_j - d^j_{h_j - 1} \ : \ wq_j^r = -1 \ , \ j \neq r\}$

$$v_3 = \begin{cases} d^r_{h_r} - d^r_{h_r - 1} & : \ w = +1 \\ d^r_{h_r - 1} - d^r_{h_r - 2} & : \ w = -1 \ \text{(suponha } d^r_{h_r - 2} = -\infty \text{ se não definido).} \end{cases}$$

$v = \text{Mjn } \{v_1, v_2, v_3\}$.

Se $v = \infty$ PARE , não há solução ótima finita.

Caso contrário, selecione a_s do ciclo em $H \cup \{a_r\}$ que conduza ao valor mínimo de v.

Passo 3. Se $v > 0$ faça $x_j = x_j + vq_j^r$ para todo $a_j \in H \cup \{a_r\}$.

Se $r = s$ e $w = +1$ faça $h_r = h_r + 1$ e calcule

$C_r^+ = c_{h_r}^r - y_{a_r}h + y_{a_r}t$; caso $C_r^+ < 0$ faça $\bar{C}_r = C_r^+$ e vá ao Passo 2.

Se $r = s$ e $w = -1$ faça $h_r = h_r - 1$ e calcule

$C_r^- = c_{h_r-1}^r - y_{a_r}h + y_{a_r}t$; caso $C_r^- > 0$ faça $\bar{C}_r = C_r^-$ e vá ao Passo 2.

Se $r \neq s$, substitua a_s por a_r em H e faça $y_i = y_i + p_r^i \bar{C}_r$, onde p_i^r é o indicador de a_r no caminho da raiz para cada nó i.

Caso $w = -1$ faça $h_r = h_r - 1$.

Caso $wq_s^r = +1$ faça $h_s = h_s + 1$ e calcule

$C_s^+ = c_{h_s}^s - y_{a_s}h + y_{a_s}t$; caso $C_s^+ < 0$ faça $r = s$, $\bar{C}_r = C_s^+$, $w = +1$ e vá ao Passo 2.

Caso $wq_s^r = -1$ calcule $C_s^- = c_{h_s-1}^s - y_{a_s}h + y_{a_s}t$; caso $\bar{C}_s > 0$ faça $r = s$, $\bar{C}_r = C_s^-$, $w = -1$ e vá ao Passo 2.

Vá ao Passo 1.

Definição: Denomina-se *pivoteamento* cada execução do Passo 2 e denomina-se *iteração* cada seqüência de execução dos Passos 2 e 3 entre duas execuções sucessivas do Passo 1.

Observação: Uma iteração envolve no máximo $\sum_j [k_j^r : q_j^r \neq 0]$ pivoteamentos, relativos ao mesmo ciclo, onde k_j é o número de intervalos do arco a_j. Note que em cada iteração, o arco que sai da base não é candidato a retornar à base na iteração seguinte.

Passa-se a comentar as características principais deste procedimento.

(a) *Exploração Continuada de Variável Básica:*

Um detalhe que diferencia a PRL da PRLP é que nesta última, um arco básico que acabou de sair da base pode retornar à base no pivoteamento imediatamente posterior, ainda na mesma iteração. Pode-se interpretar esta possibilidade de retorno à base, através da situação em que o método simplex é aplicado à *rede linear equivalente* ao PRLP, onde cada arco a_s do PRLP é substituído por arcos paralelos associados a cada um dos seus intervalos de fluxo (d_k^s). A exploração continuada de variável básica visa aproveitar-se das informações disponíveis, numa dada iteração, sobre a árvore geradora, para efetuar a maior mudança possível (no momento) no fluxo x_s associado ao arco a_s (a ser substituído na base). O que é feito é averigüar se na árvore geradora básica atual (com a_r no lugar de a_s) o arco a_s não é candidato a voltar à base

(Passo 3 do algoritmo), caso isto ocorra, torna-se a aplicar o teste da razão para esta situação (Passo 2). Trabalhando desta forma, economiza- se iterações na PRLP. O número de vezes que esta exploração continuada for realizada corresponderá a pivoteamentos do método simplex quando aplicado a rede linear equivalente.

(b) *Exploração Continuada de Variável Não-Básica:*

Pode ocorrer que, numa dada iteração, se constate pelo teste da razão (Passo 2) que não há mudança de base, isto é, a variável a_r escolhida para entrar na árvore geradora (Passo 1) de fato não se tornou básica (o valor do fluxo x_r em a_r passou de um limitante de intervalo de fluxo para outro x'_r).

Novamente, a intenção da exploração continuada de variável não-básica é tirar proveito das informações que estão à mão sobre a árvore geradora atual (idêntica a anterior pois não houve alteração na base), para reduzir o esforço computacional posterior. O que se faz é verificar se o arco a_r não permanece candidato a entrar na base (nesta sua nova situação de valor de fluxo x'_r), caso isto aconteça, retorna-se ao teste da razão.

(c) Para se obter um vetor básico h inicial (Passo 0), pode-se considerar uma árvore geradora qualquer com intervalo adicional para cada arco básico $a_j \in H$ com valores $d^j_{-1} = -\infty$, $c^j_0 = -\infty$ (permitindo ao fluxo x_j associado a a_j ser negativo) e partir com a solução básica correspondente. Desta forma, a fase I do método simplex não necessita ser considerada em separado.

(d) Este método possue as mesmas deficiências, já apontadas anteriormente, do método simplex com relação a ciclagem e empacamento. Há necessidade de se incorporar refinamentos para evitar estes fenômenos, é o que se passa a discutir.

Ciclagem na PRLP

Um *pivoteamento* é denominado *degenerado* se o teste da razão produz $v = 0$ e uma *iteração* é chamada *degenerada* quando todos os pivoteamentos associados são degenerados. Nestas iterações não há mudança no vetor de fluxos e nenhuma melhoria é obtida na função objetivo. Um *vetor básico* h é denominado *degenerado* se há uma variável básica x_j com valor $x_j = d_k^j$ para algum k.

Considere uma seqüência(finita ou não) de pivoteamentos degenerados representada pelos vetores básicos $h^0, h^1, h^2, \ldots$. O fenômeno da *ciclagem* em PRLP ocorre quando $h^t = h^{t'}$ com $t \neq t'$, para dois elementos desta seqüência, indicando que uma determinada solução básica foi reproduzida após a realização de alguns pivoteamentos degenerados. Sob tais condições não é possível assegurar que o método simplex descrito para a PRLP seja finito.

Assim passa-se a desenvolver um refinamento deste método do simplex, através da extensão do conceito de bases fortemente viáveis para a PRLP.

Definição: Um vetor básico h, associado à solução básica (x, y, z), é denominado *vetor básico fortemente viável* se todo arco básico a_j, com $d^j_{k-1} \leq x_j \leq d^j_k$, é direcionado para a raiz se $x_j = d^j_k$, e é direcionado da raiz se $x_j = d^j_{k-1}$.

Observação: Notar que esta definição faz uso da flexibilidade com que se estabeleceu (seção 2.2) intervalo corrente h_j de um arco a_j ($d^j_{h_j-1} \leq x_j \leq d^j_{h_j}$). O intervalo corrente, que é associado a cada arco básico, caso o fluxo x_j atinja algum valor extremo de intervalo de fluxo ($d^j_{h_j-1}$ ou $d^j_{h_j}$) dependerá da orientação do arco a_j na árvore geradora básica H.

Exemplo 2.6. Considere a árvore H da figura 2.6, onde:

$c^1 = (c^1_1 \quad c^1_2 \quad c^1_3 \quad \infty)$ $\quad c^2 = (c^2_1 \quad c^2_2 \quad c^2_3 \quad \infty)$ $\quad c^3 = (c^3_1 \quad c^3_2 \quad c^3_3 \quad \infty)$

$d^1 = (0 \quad 1 \quad 5 \quad 6 \quad \infty)$; $\quad d^2 = (0 \quad 3 \quad 9 \quad 10 \quad \infty)$; $\quad d^3 = (0 \quad 1 \quad 2 \quad 9 \quad \infty)$;

$c^4 = (c^4_1 \quad c^4_2 \quad c^4_3)$ $\quad c^5 = (c^5_1 \quad c^5_2 \quad c^5_3)$

$d^4 = (0 \quad 5 \quad 10 \quad \infty)$; $\quad d^5 = (0 \quad 1 \quad 7 \quad \infty)$; com c^i_j adequados e finitos. As demandas nos nós são: $b_1 = -5$; $b_2 = +4$; $b_3 = -15$; $b_4 = +6$; $b_5 = +20$; $b_6 = -10$. Considere os seguintes valores de fluxo nos arcos básicos: $x_1 = 5$; $x_2 = 9$; $x_3 = 6$; $x_4 = 10$; $x_5 = 10$.

Desta forma para que o vetor básico h seja fortemente viável, deve-se ter h = (3 2 3 3 3), vetor dos intervalos dos arcos a_j, j = 1..5.

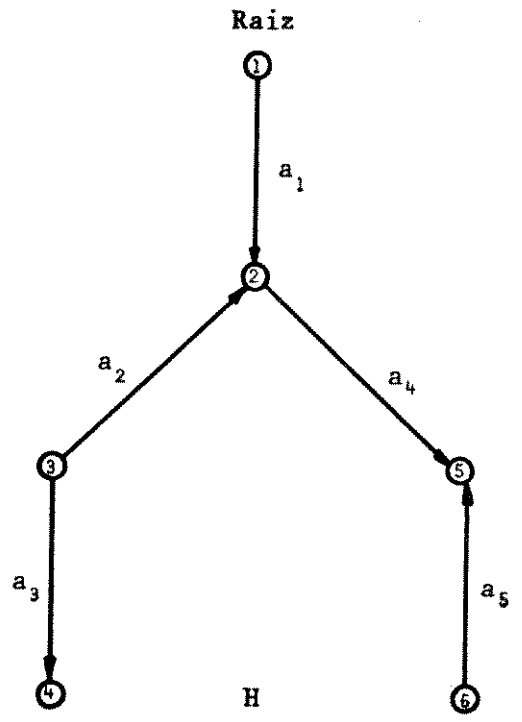


Figura 2.6. Exemplo de Vetor Básico Fortemente Viável (admitindo fluxo nulo para os arcos não-básicos)

Observação: (a) Os arcos a_1, a_2 e a_3 são canalizados, respectivamente, entre $[0, 6]$, $[0, 10]$ e $[0, 9]$;

(b) As orientações dos arcos a_3 e a_5 não têm influência na determinação de seus intervalos de fluxo.

(c) Os arcos a_4 e a_5 não são canalizados ($x_4, x_5 \geq 0$).

Mostra-se a seguir que vetores básicos fortemente viáveis do PRLP são vetores básicos viáveis de um (PRLP)' não degenerado, obtido de PRLP pela introdução de uma perturbação nas demandas dos nós.

Seja $\bar{\varepsilon} = \text{Min} \left\{ \left| \left| - \sum_{i \notin R(H,a_j)} b_i + \sum_{a_k \in E(H,a_j)} d^k_{h_k} - \sum_{a_k \in \bar{E}(H,a_j)} d^k_{h_k} \right| - d^j_{h_j} \right| > 0 \right\}$

para toda árvore geradora H, todo $a_j \in H$, todo $h_k : a_k \in E(H,a_j)$, todo $h_k : a_k \in \bar{E}(H,a_j)$ e todo h_j.

Dado um número ε, suficientemente pequeno, $0 < \varepsilon < \bar{\varepsilon}/m$, considere o vetor das demandas nos nós perturbado $b' = (b'_i)$ onde $b'_i = b_i + \varepsilon$ se $i \neq$ raiz e $b'_i = b_i - (m-1)\varepsilon$ se $i =$ raiz, e defina o problema perturbado, não degenerado, (PRLP)' por:

P': Min $\{f(x)$ s.a: $Mx = b'$, $x \geq 0\}$.

Teorema 1. Um vetor básico h é um vetor básico viável para o (PRLP)' se e somente h é um vetor básico fortemente viável para PRLP.

Prova:

Sejam $x = (x_j)$, $x' = (x'_j)$ soluções básicas primais associadas ao ao vetor h em PRLP e em (PRLP)', respectivamente. Portanto, para os arcos não-básicos

$$x_j = d^j_{h_j - 1} \quad \text{e} \quad x'_j = d^j_{h_j - 1} \implies x'_j - x_j = 0 \ ;$$

para os arcos básicos direcionados para a raiz

$$x_j = -\sum_i \left[b_i : i \notin R(H,a_j)\right] + \sum_k \left[x_k : a_k \in E(H,a_j)\right] - \sum_k \left[x_k : a_k \in \bar{E}(H,a_j)\right]$$

e

$$x'_j = -\sum_i \left[b'_i : i \notin R(H,a_j)\right] + \sum_k \left[x_k : a_k \in E(H,a_j)\right] - \sum_k \left[x_k : a_k \in \bar{E}(H,a_j)\right]$$

$$\implies x'_j - x_j = -\varepsilon \cdot (m - |R(H,a_j)|) \text{ ou ainda } 0 > x'_j - x_j > \bar{\varepsilon} \qquad (A);$$

para os arcos básicos direcionados da raiz

$$x_j = + \sum_i \left[b_i : i \notin R(H,a_j) \right] - \sum_k \left[x_k : a_k \in E(H,a_j) \right] + \sum_k \left[x_k : a_k \in \bar{E}(H,a_j) \right]$$

e

$$x'_j = + \sum_i \left[b'_i : i \notin R(H,a_j) \right] - \sum_k \left[x_k : a_k \in E(H,a_j) \right] + \sum_k \left[x_k : a_k \in \bar{E}(H,a_j) \right]$$

$$\Rightarrow \quad x'_j - x_j = + \varepsilon \, (m - |R(H,a_j)|) \quad \text{ou ainda} \quad 0 < x'_j - x_j < \bar{\varepsilon} \qquad \text{(B)};$$

Assim tem-se:

(a) Se $x_j < 0$ (inviável) então de (A) e (B) segue que $x'_j < 0$ (inviável)

(b) Se $d^j_{h_j-1} < x_j < d^j_{h_j}$ então de (A) e (B) segue que $d^j_{h_j-1} < x'_j < d^j_{h_j}$

(c) Se $x_j = d^j_{h_j-1}$ e a_j é arco direcionado da raiz então $d^j_{h_j-1} < x'_j < d^j_{h_j}$

(d) Se $x_j = d^j_{h_j}$ e a_j é arco direcionado para a raiz então $d^j_{h_j-1} < x'_j < d^j_{h_j}$.

O que encerra esta demonstração.

Incorporando este conceito de vetor básico fortemente viável ao método Simplex especializado para PRLP, aliado a uma regra para escolha do arco que deve sair da árvore H de modo a manter a sua estrutura fortemente viável, obtém-se um novo método denominado Método Simplex Fortemente Viável para PRLP, descrito a seguir.

Método Simplex Fortemente Viável (MSFV) para PRLP

Este procedimento possue as mesmas características do método simplex especializado para PRLP com as seguintes alterações:

(a) Inicializar o método com um vetor básico fortemente viável h.

(b) *Regra para escolha da variável a sair da base (Passo 3):* Seja a_r escolhido para entrar na árvore geradora H e seja u o último nó comum dos caminhos da raiz para a cabeça a_r^h e para a cauda a_r^t do arco a_r. Considere o ciclo formado em H U a_r. Atravesse este ciclo partindo do nó u com a orientação de a_r se w=1 ou com a orientação oposta se w=-1. Selecione o arco a sair da base como sendo o primeiro arco a_s encontrado que satisfaça o teste da razão (cálculo de v no Passo 2).

Observação: O MSFV trabalha somente com vetores básicos fortemente viáveis. Isto pode ser constatado pelo desenvolvimento a seguir.

Seja h um vetor básico fortemente viável para a PRLP e (x, y, z) a solução básica associada. Considere o vetor básico $\hat{h}$ como sendo o próximo vetor básico viável gerado pelo MSFV a partir de h, associado com a solução $(\hat{x}, \hat{y}, \hat{z})$. Assim $\hat{H} = (H \cup \{a_r\}) - \{a_s\}$, onde a_r é o arco escolhido para entrar na base e a_s é o arco escolhido pela regra acima para sair da base.

Observa-se que os arcos de H não pertencentes ao ciclo H U $\{a_r\}$ não terão suas orientações e fluxos alterados quando da entrada de a_r e saída de a_s, assim basta analisar os arcos do ci

clo. Na figura 2.7 apresenta-se ilustrações de como a Regra em questão escolhe o arco que deve deixar a base, mantendo a estrutura fortemente viável em $\bar{H}$. Indica-se o arco a_r escolhido para entrar (tracejado), quais os novos valores de fluxo nos arcos do ciclo $H \cup \{a_r\}$ que são responsáveis (empate no teste da razão) pelo valor de v no Passo 2 do MSFV (com $\overline{\downarrow}$ e $\underline{\downarrow}$ significando, respectivamente, que o fluxo atingiu um limitante superior ou inferior de algum intervalo de fluxo), o nó u, o sentido em que se deve percorrer o ciclo a partir de u para obter o arco a_s que deve deixar a árvore e qual deve ser o arco a_s a deixar a base (assinalado com x).

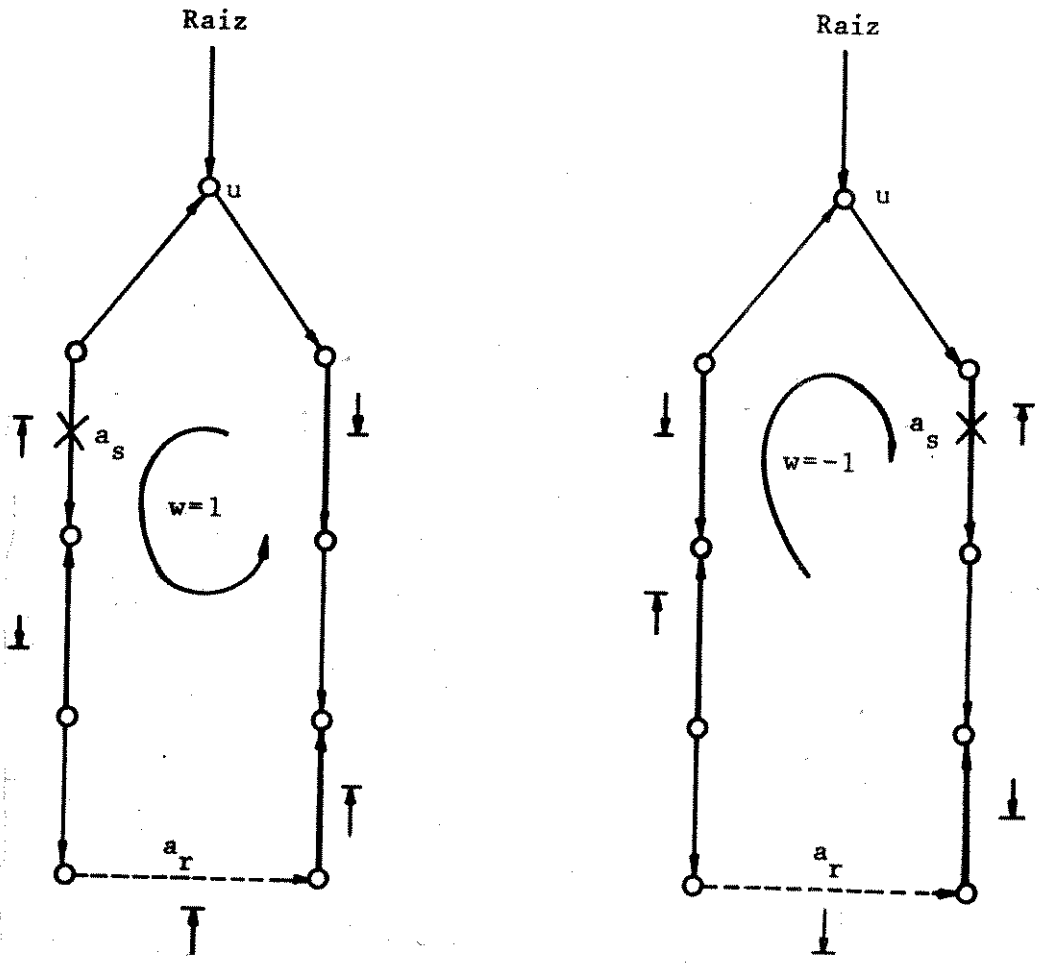


Figura 2.7. Ilustração da Aplicação da Regra que Mantém Viabilidade Forte

Percebe-se que os novos valores básicos $\bar{h}$, para cada caso, manterão a estrutura fortemente viável que h apresentava.

Lema 1. Sejam h, $\bar{h}$ dois vetores básicos onde $\bar{h}$ é obtido de h através de um pivoteamento onde o arco a_r substituiu o arco a_s na base H. Então: Sejam y, $\bar{y}$ os vetores duais associados a h, $\bar{h}$, respectivamente e $\bar{c}_r$ como no Passo 1 do MSFV.

(a) $\bar{y}_i = y_i$ se $i \in R(H,a_s)$;

(b) $\bar{y}_i = y_i - \bar{c}_r$ se $i, a_r^t \notin R(H,a_s)$;

(c) $\bar{y}_i = y_i + \bar{c}_r$ se $i, a_r^h \notin R(H,a_s)$.

Além disto, se $\bar{h}$ é obtido de h através de um pivoteamento degenerado, então:

(d) $\bar{y}_i = y_i$ se $i \in R(H,a_s)$;

(e) $\bar{y}_i = y_i + w\bar{c}_r$ se $i \notin R(H,a_s)$.

Prova: Deve-se ressaltar que $R(H,a_s) = R(\bar{H},a_r)$ com $\bar{H}$ associado a $\bar{h}$.

Dado um intervalo básico h as variáveis duais básicas satisfazem a expressão $y_i = \sum_j p_j^i \, c_{h_j}^j$ (ver expressão (1)).

O pivoteamento que transforma h em $\bar{h}$ altera os valores de $\bar{y}$ da forma $\bar{y}_i = y_i + \bar{p}_r^i \, \bar{c}_r$ (ver Passo 3 do MSFV)

onde $\bar{p}_r^i = 0$ se $i \in R(\bar{H},a_r)$;

$\bar{p}_r^i = 1$ se $i \notin R(\bar{H},a_r)$ e a_r é direcionado da raiz;

$\bar{p}_r^i = -1$ se $i \notin R(\bar{H}, a_r)$ e a_r é direcionado para a raiz.

Disto resultam as expressões (a), (b) e (c). A expressão (d) segue da validade de (a).

Em caso de pivoteamento degenerado, as propriedades de viabilidade forte implicam que:

$w=+1 \implies a_r$ é direcionado da raiz em $\bar{H}$

$w=-1 \implies a_r$ é direcionado para a raiz em $\bar{H}$.

Desta forma, obtém-se a expressão (e).

Teorema 2. O MSFV é finito.

Prova: Para pivoteamentos não degenerados o MSFV é finito pois o valor da solução primal torna-se estritamente decrescente, como para qualquer método simplex. Será provado que para pivoteamentos degenerados $\Sigma\{\bar{y}_i : i \in N\} < \Sigma\{y_i : i \in N\}$, considerando a notação do Lema 1.

Considere o caso em que w=1, isto é, o arco a_r escolhido para entrar na base tem o custo relativo $\bar{c}_r = c_r^+ < 0$ e o valor x_r deve aumentar para melhorar a função objetivo. Pelas expressões (d), (e) do Lema 1 pode-se concluir que $\Sigma \bar{y}_i < \Sigma y_i$.

Caso w=-1 então $\bar{c}_r = c_r^- > 0$ e novamente segue que $\Sigma \bar{y}_i < \Sigma y_i$.

Observação: (a) Com este resultado fica provado que a manutenção de vetores básicos fortemente viáveis, pelo MSFV, evita a ocorrência de ciclagem na PRLP, pois existe um número finito de árvores geradoras e um número finito de intervalos para cada um dos arcos básicos.

(b) Segue também que em qualquer pivoteamento degenerado tem-se $\bar{y}_i \leq y_i \ \forall_i$. Em palavras, isto significa que sob a medida de distância $y_i = \Sigma_j p_j^i \ c_{h_j}$, cada nó i ou permanece à mesma distância da raiz ou aproxima-se dela, em pivoteamentos degenerados.

Empacamento em PRLP

Foi visto portanto, que o uso de viabilidade forte evita ciclagem pois toda seqüência de iterações degeneradas será finita. Porém, o comprimento de tais seqüências pode ser exponencial com relação ao tamanho do problema (número de nós e arcos) ver Cunningham /30/, caracterizando o fenômeno de *empacamento*. Assim se faz necessário incorporar ao MSFV um segundo refinamento, relativo a escolha do arco a_r candidato a entrar na árvore, que evite este fenômeno.

Seja $h^0, h^1, \ldots, h^t$ uma sequência de vetores básicos fortemente viáveis obtidos através de iterações degeneradas do MSFV. Sejam $C_j^-(s)$ e $C_j^+(s)$, respectivamente, os valores dos cus

tos relativos à esquerda e à direita do arco a_j associado ao vetor básico h^s.

Assuma que $S_0, S_1, \ldots, S_{k-1}$ são números inteiros conhecidos satisfazendo $0 = S_0 < S_1 < \ldots < S_{k-1}$ e defina S_k (se existir) como sendo o menor inteiro maior do que S_{k-1} tal que

$$\max_s \{c_j^+(s) : s = S_{k-1} \ldots S_k\} \geq 0 \quad \text{e} \quad \min_s \{c_j^-(s) : s = S_{k-1}, \ldots, S_k\} \leq 0$$

para todo $a_j \in A$.

Definição: Se S_k definido como acima existir, então a sequência $\{h^s : s = S_{k-1}, \ldots, S_k\}$ é denominada de *estágio*.

Observação: A principal característica de um estágio é que nenhum arco permanece candidato a entrar na base durante todas as iterações que o compõe; isto é, em alguma iteração ele deixa de sê-lo.

Um dos principais resultados deste trabalho é o estabelecimento do próximo teorema.

Teorema 3. Uma seqüência de vetores básicos $h^0, h^1, \ldots, h^t$ fortemente viáveis, obtidos através de uma seqüência de iterações degeneradas do MSFV e associadas a (x, y, z) pode ter no máximo u estágios onde $u = |\{a_j \in H : x_j = d^j_{h_j-1} \quad \text{ou} \quad x_j = d^j_{h_j}\}|$ é o número de arcos básicos degenerados em cada vetor básico h.

Prova: É uma extensão daquela apresentada por Cunningham /30/.

Seja G(x) a floresta formada pelos arcos básicos não degenerados $\{a_j \in H : d^j_{h_j-1} < x_j < d^j_{h_j}\}$. Um arco $a_j \in H$ é dito ser degenerado se $x_j = d^j_{h_j-1}$ ou $x_j = d^j_{h_j}$. Do Lema 1 segue que para quaisquer dois nós v e w da mesma componente de G(x) e para quaisquer iterações p e q, tem-se $y^p_v - y^p_w = y^q_v - y^q_w$, onde y^p_i denota a variável dual (preço) no nó i associada ao vetor básico h^p.

Para provar o teorema basta mostrar que se o caminho na árvore básica, associada a h^t, da raiz para v tem, no máximo, i arcos degenerados então $y^{S(i)}_v = y^t_v$ se S(i) for definido; pois tais caminhos podem ter no máximo u arcos degenerados. A demonstração é feita por indução finita.

Para i=0, a afirmação é válida pois v é um nó da componente de G(x) que contém a raiz e, portanto, y_v não sofre alterações durante a seqüência de pivoteamentos degenerados.

Suponha que a afirmação é válida para 0, 1, 2,..., i-1,

Admita, por absurdo, que a afirmação não vale para i. Então para algum nõ v cujo caminho, associado a h^t, da raiz a v possue i arcos degenerados tem-se $y_v^{S(i)} > y_v^t$ (observe que $y_v^{S(i)} \geq y_v^t$ pelo Lema 1).

Seja a_j o último arco degenerado no caminho da raiz a v na árvore básica associada a h^t. Suponha que a_j é um arco direcionado da raiz; para o caso contrário a prova é similar. Pela hipótese de indução, o caminho da raiz a a_j^t nesta mesma árvore, possui (i-1) arcos degenerados e $y^k_{a_j t} = y^t_{a_j t}$ com $S(i-1) \leq k \leq S(i)$. Além disto, como v e a_j^h estão na mesma componente de G(x) tem-se $y^k_{a_j h} > y^t_{a_j h}$ com $S(i-1) \leq k \leq S(i)$, pela hipótese feita e pelo Lema 1.

Segue, portanto, que para $S(i-1) \leq k \leq S(i)$:

$$C_j^+(k) = C^j_{h_j} + y^k_{a_j t} - y^k_{a_j h} = C^j_{h_j} + y^t_{a_j t} - y^k_{a_j h} < C^j_{h_j} + y^t_{a_j t} - y^t_{a_j h} = 0,$$

ou seja, $C_j^+(k) < 0$, significando que por todo o estágio i, o arco a_j permaneceu candidato a entrar na base, contradizendo a definição de estágio i.

Assim, a afirmação que $y_v^t = y_v^{S(i)}$ é válida para i e em geral por indução, ficando provado o teorema.

Observação: (a) Este teorema sugere o uso de regras de entrada no MSFV que examinem os arcos de forma que cada um venha a ser examinado uma vez em cada estágio.

(b) No estágio k+1, todos os arcos básicos de qualquer caminho da raiz, na árvore final com no máximo k arcos degenerados, são mantidos básicos como conseqüência do fato de que os valores dos preços de y_i dos nós destes caminhos não são alterados. Assim com relação à medida de distância $y_i = \sum_j p_j^i \, C_{h_j}^j$ as componentes de G(x) "aproximam-se" da raiz em iterações degeneradas.

Descreve-se a seguir algumas das regras de entrada mais comuns e alguns resultados para a PRLP com relação a prevenção de empacamento com o MSFV.

Regra nº 1 - Maior Melhoria

Simula a entrada de cada um dos arcos candidatos ($C_r^+ < 0$ ou $C_r^- > 0$) e escolhe aquele que ocasiona a maior melhoria na função objetivo. Não é computacionalmente eficiente.

Regra nº 2 - Maior Taxa de Melhoria (Dantzig)

Escolhe o arco candidato a_r que possua o maior custo relativo (em valor absoluto) dado por $\bar{C}_r = \text{Max}\{\{|C_i^+| : \text{para todo arco } a_i \text{ não-básico com } C_i^+ < 0\} \cup \{|C_i^-| : \text{para todo arco } a_i \text{ não-básico com } C_i^- > 0\}\}$. Esta regra é também denominada de método da aresta de máximo declive ("Steepest-edge"). Também não é eficiente com rela

ção a tempo computacional, mas costuma sê-lo com respeito ao número de iterações.

Regra nº 3 - Maior Taxa Normalizada de Melhoria

Escolhe o arco candidato a_r que possua o maior custo relativo normalizado em valor absoluto dado por

$$\bar{C}_r = \text{Max} \left\{ \frac{|C_j^+|}{\sqrt{C_j^{+2} + \sum_i (q_i^j)^2}} \text{ ; para todo } a_j \text{ não básico com } c_j^+ < 0 \right.,$$

$$\left. \frac{|C_j^-|}{\sqrt{C_j^{-2} + \sum_i (q_i^j)^2}} \text{ : para todo arco } a_j \text{ não-básico com } c_j^- > 0 \right\}.$$ Esta regra escolhe a aresta que apresenta o menor ângulo com o gradiente.

Note que, para redes, Σq_j^2 é o número de arcos básicos no ciclo formado por H U $\{a_r\}$. Ver Crowder e Hattingh /46/ para estudos comparativos entre variantes desta regra e a regra nº 2; se conclue que estas variantes diminuem significativamente o número de iterações.

Regra nº 4 - Não-Básica Mais Antiga (LRB)

Escolhe o arco candidato que deixou a base a mais tempo. *Seja $a_1, a_2, \ldots, a_n$ uma ordenação arbitrária pré-fixada dos arcos da rede.*

Regra nº 5 - Menor Índice (Bland)

Escolhe o primeiro arco candidato da seqüência $a_1, a_2, \ldots, a_n$. Pode ser desinteressante do ponto de vista de tempo computacional (o Exemplo 2.4 é um caso em que isto ocorre).

Regra nº 6 - Testada a Mais Tempo (Cíclica)

Supondo que a_r foi o último arco candidato escolhido, selecione o primeiro arco candidato da sequência $a_{r+1}, a_{r+2}, \ldots, a_n, a_1, a_2, \ldots, a_r$.

Seja $v_1, v_2, \ldots, v_m$ uma ordenação arbitrária pré-fixada dos nós da rede.

Regra nº 7 - Maior Taxa de Melhoria no Nó (Nó-Cíclica)

Supondo que v_k foi o último nó considerado, selecione o arco candidato a_r, com maior custo relativo (em valor absoluto) dado por $\bar{C}_r = \text{Max}\{\{|C_i^+|$: para todo arco a_i não básico com $C_i^+ < 0$ e cabeça a_i^h no nó $v\} \cup \{|C_i^-|$: para todo arco a_j não-básico com $C_i^- > 0$ e cauda a_i^t no nó $v\}\}$, onde v é o primeiro nó da lista $v_{k+1}, v_{k+2}, \ldots, v_m, v_1, \ldots, v_k$ que seja cabeça ou cauda de algum arco candidato com a propriedade acima.

Regra nº 8 - Maior Taxa Continuada de Melhoria no Nó (Nó-Cíclica Continuada)

Idêntica a anterior, com a lista de nós alterada para $v_k, v_{k+1}, \ldots, v_m, v_1, v_2, \ldots, v_{k-1}$.

As regras de número 2, 4, 5, 6, 7 e 8 foram implementadas, alternadamente, com o MSFV e seus comportamentos (número de iterações, cpu) comparados. Isto é discutido no capítulo 3.

A seguir apresenta-se, no Teorema 4, resultados para a PRLP envolvendo o MSFV combinado com regras de entrada que evitam empacamento, antes porém prova-se o Lema 2 que será útil nestas demonstrações.

Lema 2. Sejam h, $\hat{h}$ dois vetores básicos onde $\hat{h}$ foi obtido de h a partir de uma iteração degenerada, com arco candidato a_r incidente em nó v e arco de saída a_s. Suponha que $\bar{C}_r = \text{Max}\{\{|C_j^+| : C_j^+ < 0 \text{ e } a_j^h = v\} \cup \{|C_j^-| : C_j^- > 0 \text{ e } a_j^t = v\}\}$ e que a_t é um arco candidato na iteração seguinte com a propriedade ($\hat{C}_t^+ < 0$ e $a_t^h = v$) ou ($\hat{C}_t^- > 0$ e $a_t^t = v$). Então a escolha de a_t como candidato em $\hat{h}$ conduz a uma iteração não degenerada.

Prova: É uma extensão da apresentada por Cunningham /30/. Demonstra-se a validade deste lema para o caso $w=1$, ou seja, $C_r^+ < 0$ e $v = a_r^h$. Além disto, suponha que $\hat{w}=1$, ou seja, $\hat{C}_t^+ < 0$ e $a_t^h = v$. Os demais casos ($w=1$, $\hat{w}=-1$), ($w=-1$, $\hat{w}=1$), ($w=-1$, $\hat{w}=-1$) são similares. Desta forma, $a_r^t \in R(H, a_s)$ em iterações degeneradas com $w=1$.

Do Lema 1, pelo fato de ser uma iteração degenerada, tem-se que $\hat{y}_v = \hat{y}_{a_r^h} = y_{a_r^h} + C_r^+ = y_v + C_r^+$ e $\hat{y}_{a_r^t} = y_{a_r^t}$. Supondo, por

absurdo, que $\hat{y}_{a_t^t} = y_{a_t^t}$ tem-se

$\hat{C}_t^+ = C_{h_t} - \hat{y}_{a_t^h} + \hat{y}_{a_t^t} = C_{h_t} - (y_v + C_r^+) + y_{a_t^t} = C_t^+ - C_r^+ \geq 0$, o que contradiz a hipótese de que $\hat{C}_t^+ < 0$. Desta forma, pelo Lema 1, tem-se que $\hat{y}_{a_t^t} = y_{a_t^t} + C_r^+$, e mais $v = a_r^h = a_t^h$, $a_t^t \notin R(H,a_s) = R(\hat{H},a_r)$.

Considere os caminhos da raiz até a_t^t em H e em $\hat{H}$. Ambos os caminhos contêm o nó v. Assim, o caminho de v a a_t^t não passa pela raiz. Devido a propriedade de viabilidade forte, todos os arcos degenerados direcionados da raiz estão no limitante inferior do intervalo corrente e todos os arcos degenerados direcionados para a raiz estão no limitante superior do intervalo corrente. Em consequência, o ciclo formado por $\hat{H} \cup \{a_t\}$ não permite pivoteamento degenerado (vide figura 2.8), o que finaliza esta demonstração.

Na figura 2.8 apresenta-se um exemplo desta evolução de vetores básicos fortemente viáveis, onde os símbolos $\overline{\mp}$ ($\underline{\pm}$), $\uparrow$ ($\downarrow$) significam, respectivamente, que o arco deveria ter seu fluxo aumentado (diminuído) mas ele já se encontra no maior (menor) valor possível e o arco pode ter seu fluxo aumentado (ou diminuído) sem problemas com a canalização existente.

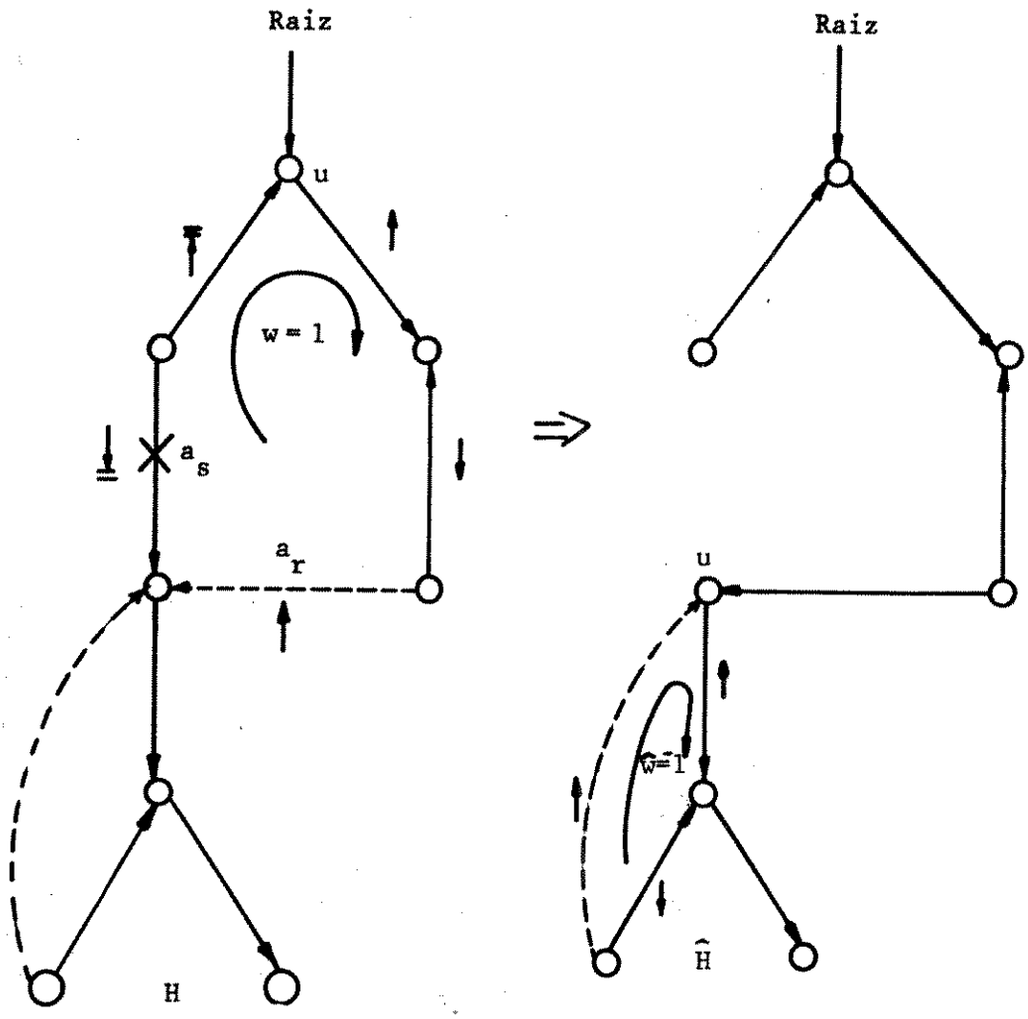


Figura 2.8. Ilustração para o Lema 2

(w=1 e $\hat{w}$=1)

Teorema 4. Seja h^0, h^1,..., h^t uma seqüência de vetores básicos obtidos através de iterações degeneradas associadas à solução (x, y, z). Seja u o número de arcos degenerados da solução básica. Para uma rede com m nós e n arcos tem-se:

(a) Se a Regra nº 4 for usada então $t \leq u \cdot n$;

(b) Se a Regra nº 6 for usada então $t \leq u \cdot n$;

(c) Se a Regra nº 7 for usada então $t \leq u \cdot m$;

(d) Se a Regra nº 8 for usada então $t \leq u \cdot m$.

Prova: É uma extensão da apresentada por Cunningham /30/. A demonstração da validade dos itens (a) e (b) é de certa forma evidente. Comparando-se a definição de estágio, Regra nº 4 e Regra nº 6, percebe-se que cada estágio decorrente da aplicação de uma ou outra regra terá um comprimento menor ou igual a n. De modo que (a) e (b) seguem do Teorema 3.

Para demonstrar o item (d), observe que após uma iteração degenerada com base em um nó v (notação do Lema 2), caso permaneça para a seguinte algum candidato incidente neste nó v com a propriedade exigida pela regra 8, será efetuada uma iteração não degenerada. Assim, no pior caso, podem ocorrer no máximo m iterações degeneradas por estágio o que prova este item.

A demonstração do item (c) é mais elaborada, devendo ser aplicada, como no último Teorema, o recurso da indução finita.

Utiliza-se a notação do Teorema 3. Prova-se que se o caminho P na árvore associada com h^t, da raiz para v, tem no máximo i arcos degenerados e $i.m \leq t$ (isto garante a existência do estágio i) então $y_v^t = y_v^{i.m}$.

Usando a técnica de indução sobre i, vem:

Para i=0, a verificação da validade de $y_v^t = y_v^{i.m}$ é evidente pois y_v não sofre modificação alguma nos pivoteamentos degenerados que geram a seqüência $h^0, \ldots, h^t$, pois v é um nó da componente de G(x) (ver Teorema 3), que contém a raiz.

Admita-se que é válida para 0,1,..., i-1.

Suponha, por absurdo, que não é válida para i. Assim existe v ∈ N tal que o caminho P associado ao vetor básico h^t, da raiz para este nó v, tem i arcos degenerados e $y_v^t < y_v^{i.m}$. Observe que pelo Lema 1 $y_v^t \leq y_v^{i.m}$.

Seja a_j o último arco degenerado de P. Suponha que a_j é direcionado da raiz. A prova para o caso contrário é similar. Segue que, $y^t_{a_j t} = y^{(i-1)m}_{a_j t}$ e (3): $y^t_{a_j h} = y^t_{a_j t} + c^j_{h_j t} < y^{i.m}_{a_j h}$. Do modo que para todo k onde $(i-1)m \leq k \leq i.m$ tem-se $c_j^+(k) < 0$. Suponhado que, para alguma destas iterações, foi escolhido para entrar na árvore um arco a_r, tal que ($a^h_r = a^h_j$ e w=1) ou ($a^t_r = a^h_j$ e w=-1), como estabelecido na Regra nº 7; o pivoteamento degenerado resultante diminui a variável dual $y_{a_j h}$ de, no mínimo, o mesmo valor que o arco a_j o faria com $c_j^+(k)$; ou seja, (4): $y^{i.m}_{a_j h} \leq y^t_{a_j t} + c^j_{h_j t} = y^t_{a_j h}$. Assim chega-se a uma contradição (expressões (3) e (4)), de modo que a afirmação de que $y_v^t = y_v^{i.m}$ com $i.m \leq t$ é válida para todo i.

Como nenhum caminho da raiz para qualquer nó v, na árvore básica de h^t, tem mais que u arcos degenerados, finaliza-se a prova do item (c), encerrando a demonstração deste Teorema.

No capítulo seguinte, apresenta-se resultados computacionais decorrentes da implementação destas regras no MSFV, tendo

sido abordados PRLP gerados aleatoriamente (com diversas caracte rísticas de dados) e PRLP especiais (Exemplos 2.4 e 2.5).

Passa-se a seguir a apresentação de algumas considera ções acerca da utilização do MSFV aliado a Técnica de Mudança de Escala ("Scaling") introduzida por Edmonds e Karp /35/, /36/.

2.5. TÉCNICA DE MUDANÇA DE ESCALA

A Técnica de Mudança de Escala, introduzida por Edmonds e Karp /35/, /36/ para a solução de problemas de fluxo com custo mínimo em redes, consiste em se resolver uma série de problemas aproximados do problema original; a solução ótima obtida em um problema aproximado é utilizada como solução inicial do problema seguinte; espera-se que esta solução inicial esteja próxima da so lução ótima de modo que o esforço computacional necessário para resolver cada problema aproximado seja reduzido.

Ocorre que após a divulgação deste procedimento por Edmonds e Karp /35/, /36/, durante muito tempo ele foi referencia do como sendo simplesmente um instrumento teórico, para se encon trar limitantes polinomiais no tamanho do problema (isto é, núme ro de nós, número de arcos e maior parâmetro numérico da rede) pa ra o tempo de execução de um dado algoritmo (ver Gabow /37/).

Recentemente, alguns pesquisadores vêm mudando este panorama (ver Ikura e Nemhauser /38/), apresentando resultados computacionais para implementações desta técnica em problemas de fluxo em redes altamente interessantes, como foi destacado na se ção 1.1.3.

Com esta motivação resolveu-se investigar o comportamento do MSFV incorporando esta técnica de mudança de escala quando aplicado a PRLP. Este procedimento foi implementado e testado junto a redes geradas aleatoriamente e também a redes especiais, tais como as propostas por Zadeh /42/ (Exemplo 2.4) e por Ikura e Nemhauser /38/.

Foi introduzida no MSFV uma *precisão de escala*, denotada por *prec*. Cada problema aproximado está associado a um valor de prec, que define uma unidade (indivisível) na qual os valores de fluxo nos arcos, as demandas nos nós, os limitantes de intervalos de fluxo e os custos nestes intervalos, são avaliados.

Seguindo, inicialmente, o enfoque de Edmonds e Karp procura-se aplicar a Técnica de Mudança de Escala (TME) sobre as demandas e os limitantes de intervalos (conseqüentemente também sobre os fluxos). Assim, para cada valor da precisão de escala (prec) definiu-se um PRLP aproximado dado por:

$$\bar{P}(p) : \text{Min } \{\bar{f}(x) \quad \text{s.a:} \quad Mx = \bar{b},\ x \geq 0\}$$

onde $\bar{f}(x) = f(x)$, substituindo-se para cada arco a_j, d^j por $\bar{d}^j$ e mantendo-se os custos nos intervalos correspondentes;

$$\bar{b} = (\bar{b}_i) \quad , \quad \text{com} \quad \bar{b}_i = \lfloor b_i/prec \rfloor * prec \ ;$$

$$\bar{d}^j = (\bar{d}^j_k) \quad , \quad \text{com} \quad \bar{d}^j_k = \lfloor d^j_k/prec \rfloor * prec \ .$$

Resolve-se uma sequência de problemas $\bar{P}(p)$ deste tipo, com $prec = 2^p$ para o problema $\bar{P}(p)$ com $p = \lfloor \log_2(\max \{|b_i|, |d^j_k|\}) \rfloor, \ldots, 2, 1,$

isto é, desconsiderando-se os p bits de ordem mais baixa na representação binária de b_i e d_k^j. Observe que para p=0 (prec=1) o problema $\bar{P}(p)$ corresponde ao problema original, desde que este possua dados inteiros. Os resultados teóricos obtidos por Edmonds e Karp se referem apenas a problemas com dados inteiros.

A solução ótima do problema $\bar{P}(p)$ é transformada na solução viável inicial do problema $\bar{P}(p-1)$, pela resolução de $Mx = \bar{b}$ (com prec = 2^{p-1}) e pela atualização dos intervalos de fluxo dos arcos.

Uma vantagem aparente deste método é que, quanto maior o valor de p mais degenerado tende a ser o problema $\bar{P}(p)$; como os problemas que apresentam degeneração podem ser resolvidos mais rapidamente que os não-degenerados de mesmo porte, espera-se resolver o problema original mais rapidamente com o uso da TME.

Esta implementação do MSFV aliado com a TME, foi testada em redes geradas aleatoriamente e em redes especiais, conhecidas da literatura; os resultados são comentados na seção 3.8.

Procurando, agora, explorar a mesma linha de pesquisa desenvolvida por Ikura e Nemhauser /38/, também implementou-se a TME no MSFV atuando apenas sobre os custos de cada intervalo de fluxo de todos os arcos da rede original. Estes autores usam a TME num método dual simplex em redes. Desta forma a solução ótima de um problema aproximado permanece dual viável para o problema aproximado subseqüente e evita a resolução sistemática do sistema de restrições, como no enfoque anterior da TME nas demandas.

Neste trabalho, dada a disponibilidade do MSFV já implementado, foi feito um estudo da TME aplicada aos coeficientes

de custos. Desta forma, a solução ótima de um problema aproximado permanece primal viável para o problema aproximado subseqüente pois não há alteração nas demandas.

No novo método, resolve-se uma sequência de PRLP aproximados, da forma:

$$\tilde{P}(p) : \text{Min } \{\tilde{f}(x) \text{ s.a: } Mx = b,\ x \geq 0\}$$

onde $\tilde{f}(x) = f(x)$ com $\tilde{c}_k^j = \lfloor c_k^j / \text{prec} \rfloor * \text{prec}$.

Os valores da precisão de escala usados foram do mesmo tipo, $\text{prec} = 2^p$ com $p = \lfloor \log_2 \max \{|c_k^j|\} \rfloor, \ldots, 2, 1$, daqueles aplicados no procedimento anterior. Da mesma forma, para p=0 (prec=1) o problema $\tilde{P}(p)$ corresponde ao problema original, desde que este possua coeficientes de custo inteiros. Novamente, os resultados são comentados na seção 3.8.

CAPITULO 3

ANÃLISE COMPUTACIONAL

3.1. INTRODUÇÃO

No que se segue apresenta-se detalhes sobre as implementações do MSFV para a PRLP, descrito na seção 2.4, com e sem a Técnica de Mudança de Escala; comenta-se as implementações de 6(seis) regras de entrada escolhidas dentre as citadas na seção 2.4 e também apresenta-se os geradores de redes utilizados.

Todas as implementações foram feitas visando a linguagem PASCAL, no VAX Cluster da UNICAMP, que é composto por dois sistemas VAX/VMS 785, cada um com 12 Mbytes de memória principal/CPU e discos compartilhados com 2,7 Gbytes.

3.2. IMPLEMENTAÇÃO DO MSFV

O MSFV foi implementado em quatro versões básicas com objetivos distintos.

Na primeira versão, que foi utilizada na maior parte dos experimentos, a entrada de dados é feita através da leitura

de um arquivo, criado pelos geradores de problemas das categorias 1 e 2 (ver adiante).

Este programa possue uma rotina que tem como atribuição escolher uma árvore básica inicial, que seja fortemente viável, e armazená-la através de uma estrutura de dados eficiente (ver detalhes adiante). Para armazenar os dados e para efetuar os cálculos, ele utiliza a precisão simples, o que permite seu uso em microcomputadores.

Para controlar os efeitos devidos a erros de arredondamento, principalmente em presença de soluções básicas não-viáveis, optou-se por separar as iterações em fases, de acordo com a grandeza dos custos relativos dos arcos candidatos a entrar na árvore geradora básica, sob o controle de uma variável não-negativa denominada *tol* (tolerância para candidato).

Assim para uma determinada fase são considerados candidatos os arcos não-básicos a_j com custo relativo à direita $c_j^+ < -\text{tol}$ ou custo relativo à esquerda $c_j^- > \text{tol}$. Para cada uma das fases atribuiu-se os valores:

tol = inft/10 onde inft = 10^8 (fase I do método simplex);

e demais fases com a multiplicação do último valor de tol por 10^{-6}, enquanto tol > 10^{-30}. Este fator 10^{-6} está associado ao uso da precisão simples.

Ao final de cada fase a solução básica obtida é recalculada pela expressão (1) da seção 2.2, para evitar o acúmulo de erros de arredondamento da fase anterior.

A segunda versão do MSFV se diferencia da anterior por adotar precisão quádrupla e não usar a variável tol; a entrada de dados também é feita pela leitura de um arquivo, gerado pelos programas da categoria 3 dos experimentos (ver adiante) onde, além das informações sobre a rede (como na primeira versão), também já se fornece uma árvore básica fortemente viável inicial, ou seja, são dados quais são os arcos básicos e toda a estrutura de dados relativa a esta árvore inicial.

Quanto a terceira versão, ela possue as mesmas características da segunda (precisão quádrupla e entrada de dados), mas tem incorporada a Técnica de Mudança de Escala (TME) por meio da inclusão da variável não-negativa *prec* (precisão da aproximação, já comentada na seção 2.5), atuando sobre as demandas e limitantes de intervalos de fluxo.

A quarta versão é idêntica a terceira, apenas que a TME é aplicada sobre os custos de cada intervalo de fluxo, para todos os arcos. Estas duas últimas versões foram usadas nos testes das categorias 3 e 4.

As quatro versões tem em comum o que se segue:

- podem ser conectadas com todas as regras de entrada implementadas;
- usam as mesmas rotinas internas do VAX para medida do tempo de execução do programa (CPU), que são a FUNCTION LIB$INIT-TIMER e a FUNCTION LIB$SHOW TIMER que constam na Run-Time Libraries Routines vol. 8B, existente no Centro de Computação da UNICAMP.

- usam a mesma estrutura de dados, que é uma variação do "Augmented Threaded Index" - ATI proposto por Glover, Klingman e Stutz /45/(1974).

Esta estrutura se fundamenta em dois "labels", o *predecessor* (denotado por pred) e o *thread* (denotado por tred). A variação empregada usa, ao invés de *nó predecessor*, o *arco básico predecessor*, que liga cada nó da árvore básica ao seu nó predecessor segundo a estrutura ATI.

- As rotinas que efetuam o teste da razão ("ratio test") e a atualização da árvore básica ("update") são derivadas daquelas apresentadas por Ali, Helgason, Kennington e Lall na seção 3.1 de /34/, sendo que o teste da razão tem incluída a regra(de saída) para manutenção da árvore fortemente viável nos seus comandos.

Para a armazenagem e cálculos, segundo esta estrutura de dados, foram utilizados 8 (oito) vetores dimensionados pelo número de nós das redes a serem resolvidas ("node-lenght arrays"), compostos por:

* *para os dados* - valor da demanda no nó, 1º arco incidente com cabeça no nó, 1º arco incidente com cauda no nó;
* *para a solução dual* - valor do preço (y_i) no nó;
* *para a solução primal* - valor do fluxo (x_j) no arco básico predecessor;
* *para a árvore básica* - valor do pred, valor do tred no nó e uma variável auxiliar (flag).

Além destes, utiliza-se 4 (quatro) vetores dimensionados pelo número de intervalos de fluxo nos arcos:

* *para os dados* - valor do limitante de cada intervalo, custo no intervalo, constante no intervalo e arco associado ao intervalo.

Também se usa 5 (cinco) vetores dimensionados pelo número de arcos:

* *para os dados* - arco adjacente com mesma cauda, arco adjacente com mesma cabeça, intervalo corrente do arco, cabeça e caúda do arco;

* *para a solução primal* - vetor de fluxo nas variáveis básicas e não-básicas.

Com estes vetores torna-se eficiente a manutenção e atualização da árvore básica, bem como o uso das diversas regras de entrada implementadas, além de favorecer uma possível (futura) extensão do algoritmo para desempenhar uma análise de pós-otimalidade na solução ótima.

Fornece-se a seguir uma ilustração do relacionamento entre estes diversos vetores na estrutura de dados escolhida.

Sejam:

$N = \{1,2,\ldots, m\}$ o conjunto de nós da rede;

$A = \{a_1, a_2,\ldots, a_n\}$ o conjunto de arcos da rede;

$I = \{1,2,\ldots, |I|\}$ o conjunto (sequencial) de intervalos de fluxo dos arcos;

b_i a demanda no nó i;

y_i o valor da variável dual (preço) associada ao nó i;

inch ∈ A o primeiro arco da lista de arcos com cabeça no nó i;

inct ∈ A o primeiro arco da lista de arcos com cauda no nó i;

tred ∈ N o primeiro nó da lista do "label" thread;

pred ∈ A o arco que liga o nó i ao seu predecessor (*);

fluxo o valor do fluxo no arco pred do nó i;

head ∈ N o nó que é cabeça do arco a_j;

tail ∈ N o nó que é cauda do arco a_j;

adjh ∈ A o próximo arco da lista de arcos com mesma cabeça;

adjt ∈ A o próximo arco da lista de arcos com mesma cauda;

valor o valor do fluxo no arco (básico e não-básico);

itrv ∈ I o número do intervalo corrente do arco a_j;

d_k^j os limitantes inferiores de cada intervalo de fluxo do arco a_j;

c_k^j os custos em cada intervalo de fluxo do arco a_j;

hc_k^j as constantes em cada termo linear por partes associados aos intervalos de fluxo do arco a_j;

arc ∈ A identificação do arco a_j que está associado a cada valor de itrv;

Na figura 3.1 apresenta-se um exemplo do funcionamento da estrutura de dados, com os respectivos ponteiros:

(*) Neste caso o arco tem associado um sinal (+) se é direcionado da raiz para o nó i e (-) se é direcionado do nó i para a raiz.

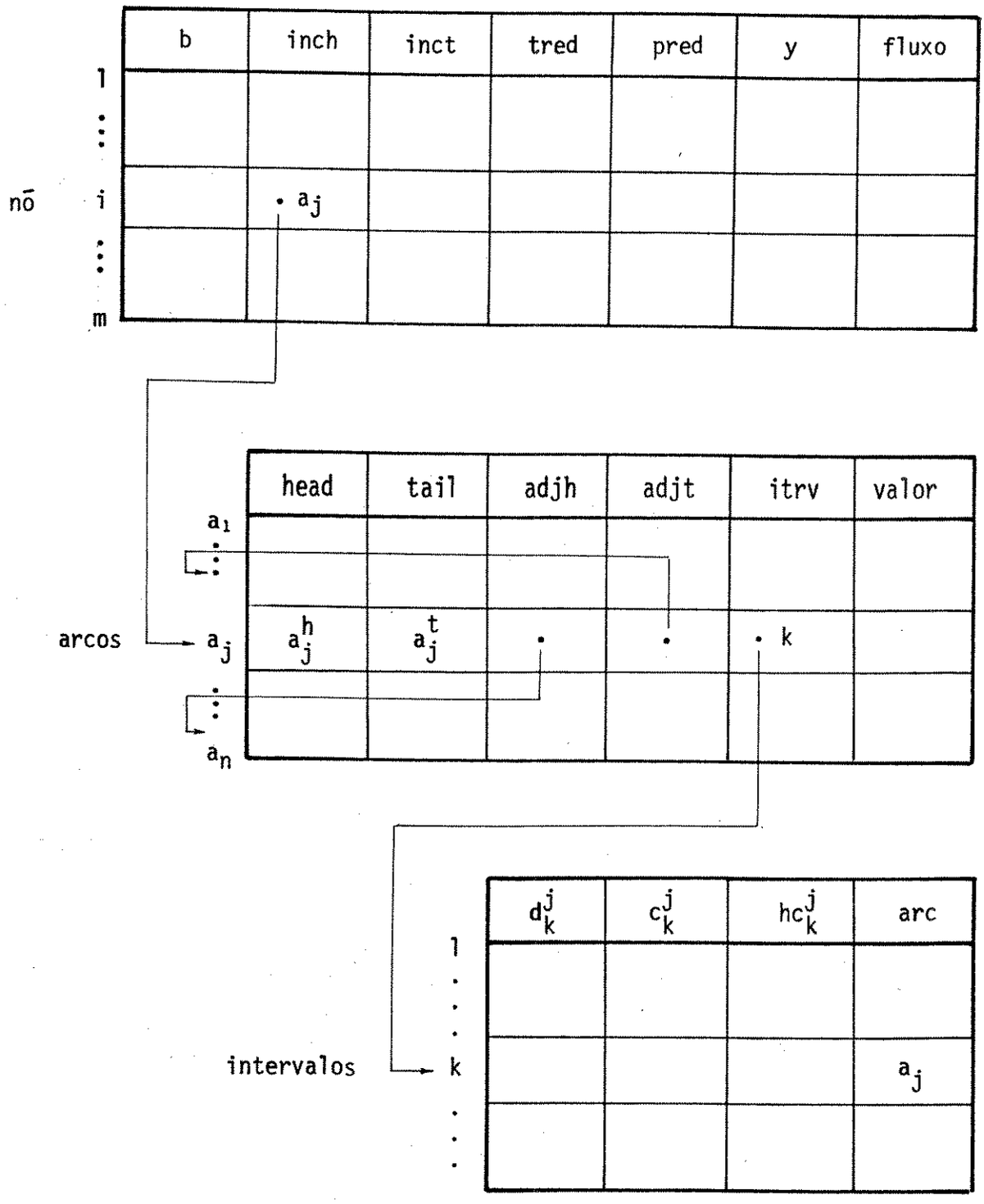


Figura 3.1. Exemplo da Estrutura de Dados Utilizada no MSFV

As regras de entrada escolhidas para serem programadas foram: Regra Dantzig, Regra Bland, Regra Cíclica, Regra Nó-Cíclico, Regra Nó-Cíclica Continuada e Regra LRB. Desta forma pôde-se testar regras de uso geral e regras específicas para redes, em conjunto com o MSFV. Estes programas funcionam como procedimentos externos ao programa MSFV, devendo um deles ser conectado ("linked") ao MSFV antes da execução deste. Deve-se salientar que a inicialização necessária para cada uma das regras é feita no MSFV independentemente da regra escolhida para o teste.

3.3. IMPLEMENTAÇÃO DOS GERADORES DE REDES

Foram desenvolvidos diversos geradores de redes para a preparação de testes computacionais. Estes geradores estão classificados em categorias.

- ***Categoria 1***

Foram implementados 4(quatro) geradores de redes pseudo-aleatórias que permitissem aferir o desempenho do MSFV para várias configurações de redes lineares por partes. Estes geradores são basicamente idênticos, diferindo apenas pelo tipo de dado (inteiro ou real) gerado para a rede; todos fazem uso de uma rotina interna do VAX para geração de números pseudo-aleatórios, a partir de um número dado denominado *semente*, que é a FUNCTION MTH$RANDOM.

Na geração dos problemas, solicita-se do usuário as seguintes informações acerca da rede:

* valor da semente;
* número de nós (m) e arcos (n);
* valor da probabilidade ($0 \leq p_0 \leq 1$) que algum nó seja de transbordo;
* valor dos limitantes inferior (B_{inf}) e superior (B_{sup}) do intervalo onde serão geradas as demandas (b) nos nós;
* valor dos limitantes inferior ($NINT_{inf} \geq 2$) e superior ($NINT_{sup}$) do intervalo onde será gerado o número de intervalos de fluxo (k) de cada arco;
* valor dos limitantes inferior (D_{inf}) e superior (D_{sup}) do intervalo onde serão gerados os limitantes dos intervalos de fluxo (d) de cada arco;
* valor da probabilidade ($0 \leq p_\infty \leq 1$) que um arco tenha seu fluxo canalizado;
* valor dos limitantes inferior (C_{inf}) e superior (C_{sup}) do intervalo onde serão gerados os custos (C) associados aos intervalos de fluxo de cada arco.

Deve-se observar que:

$p_0 = 0 \Rightarrow$ não há nós de transbordo;

$p_0 = 1 \Rightarrow$ todos nós são de transbordo;

$p_\infty = 0 \Rightarrow$ não há arcos com fluxo canalizado;

$p_\infty = 1 \Rightarrow$ todos arcos tem fluxo canalizado.

Além disto, exige-se que $NINT_{inf} \geq 2$ tendo em vista a inclusão nas redes geradas de intervalos adicionais, permitindo va

lores de fluxo (x_j) nos arcos fora da canalização ($x_j \leq d_0^j$), com isto évita-se o uso, em separado, de uma Fase I no MSFV.

- Categoria 2

Foram implementados 2(dois) geradores semelhantes aos que acabamos de descrever. A diferença é que, ao invés de gerarem uma rede para cada conjunto de dados, agora são geradas 3(três) redes associadas.

A primeira é uma Rede Linear por Partes (RLP) com dados inteiros ou reais, dependendo do gerador usado, onde cada arco a_j tem os limitantes de intervalos de fluxo $d_0^j, d_1^j, \ldots d_{k_j}^j$, com os custos $c_1^j, c_2^j, \ldots, c_{k_j}^j$. A segunda é uma rede linear que mantém os mesmos arcos a_j da RLP, porém tendo apenas o intervalo de fluxo $[d_0^j, d_{k_{j-1}}^j]$ e os custos $-\infty$ se o fluxo $x_j < d_0^j$, $+\infty$ se $x_j > d_{k_{j-1}}^j$ e $\bar{c}_j$ se $x_j \in [d_0^j, d_{k_{j-1}}^j]$, onde $\bar{c}_j$ é a média aritmética de $c_1^j, \ldots, c_{k_{j-1}}^j$. Esta rede é denominada Rede Linear Aproximada (RLA). Já a última rede gerada é do mesmo tipo da RLA, mas nela cada arco a_j da RLP dá origem a tantos arcos quantos são seus intervalos de fluxo (viáveis). Isto é, correspondentes a a_j há nesta rede os arcos $a_{j_1}, a_{j_2}, \ldots, a_{j_{k_j}}$, com fluxos x_{j_i} canalizados, respectivamente em $[d_0^j, (d_1^j - d_0^j)]$, $[d_0^j, (d_2^j - d_1^j)]$, $\ldots [d_0^j, (d_{k_{j-1}} - d_{k_{j-2}}^j)]$ e com custos dados por $-\infty$ se $x_{j_i} < d_0^j$, $+\infty$ se $x_{j_i} > (d_i^j - d_{i-1}^j)$ e c_i^j se $x_{j_i} \in [d_0^j, (d_i^j - d_{i-1}^j)]$. Esta rede é chamada Rede Linear Equivalente (RLE).

Na figura 3.2 apresenta-se um exemplo de ilustração.

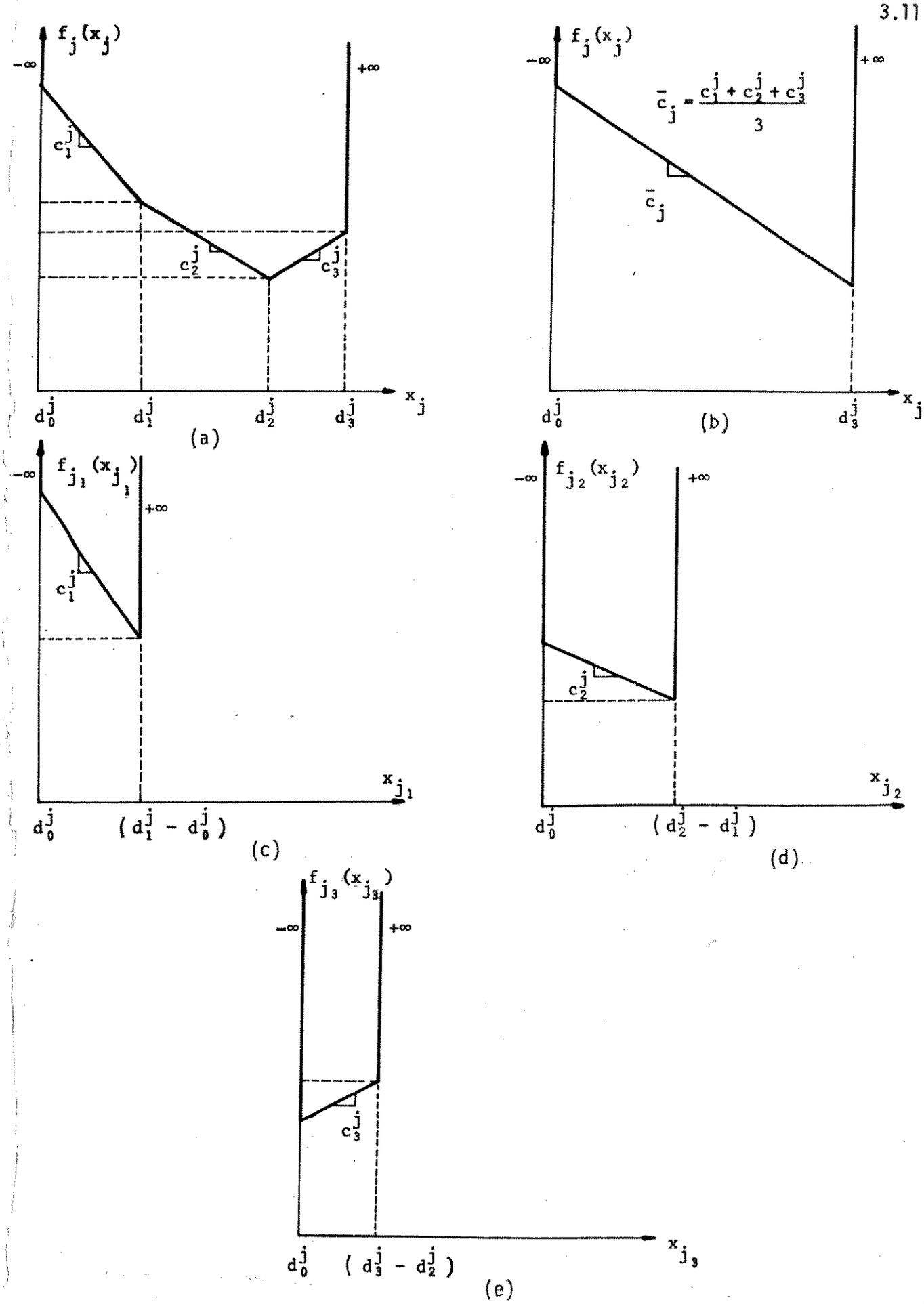


Figura 3.2. Exemplo de Obtenção de Arcos na RLA (b) e RLE (c, d, e) a Partir de um Arco a_j na RLP (a)

- *Categoria 3*

Foram desenvolvidos 2(dois) geradores que permitem reproduzir duas famílias de redes de particular importância para este trabalho. Uma delas é a família de redes proposta por Cunningham /30/, que apresenta o fenômeno de empacamento para o método simplex tradicional, mostra-se nos testes o MSFV evitando esta ocorrência. A outra família foi apresentada por Zadeh /42/, tem uma configuração que prejudica o comportamento do método simplex, mostra-se que o MSFV (com e sem a Técnica de Mudança de Escala) se desempenha muito bem na resolução destas redes.

Nestas gerações solicita-se do usuário apenas o valor do parâmetro (p) associado às dimensões da rede. No caso das redes de Cunningham tem-se $p \geq 2$ e para as redes de Zadeh tem-se $p \geq 3$.

- *Categoria 4*

Finalmente, foi implementado um gerador de redes pseudo-aleatórias do tipo proposto por Ikura e Nemhauser /38/. Estas redes representam problemas de transporte simples (em redes bipartites) com todos seus dados inteiros; na referência citada os autores obtêm um ótimo desempenho do algoritmo ali proposto (com a Técnica de Mudança de Escala). Procura-se aqui verificar se o MSFV com este refinamento tem seu comportamento melhorado no enfoque destas redes.

Solicita-se do usuário as informações:

* valor da semente;

* número de nós origens (m_1), número de nós destinos (m_2) e número de arcos (n);
* valor da soma das demandas positivas, ou negativas (DT);
* valores de ($NINT_{inf}$, $NINT_{sup}$);
* valores de (D_{inf}, D_{sup});
* valor de p_∞;
* valores de (C_{inf}, C_{sup}).

3.4. *DESCRIÇÃO DOS TESTES COMPUTACIONAIS*

Com os programas anteriormente descritos foram efetuados testes envolvendo o MSFV e as regras de entrada para diversos tipos de problemas. Estes testes podem ser classificados, com relação aos seus objetivos, em:

- *categoria 1:* comparação de desempenho entre as regras de entrada para redes lineares por partes pseudo-aleatórias, usando o MSFV.

  (a) redes lineares por partes com dados (demandas, custos e limitantes de intervalos de fluxo) sendo números reais, abrangendo situações em que há nós de transbordo (demanda nula) e/ou arcos com fluxo canalizado (há limitante superior para o fluxo permitido no arco);

(b) redes lineares por partes com dados inteiros versus redes lineares por partes (análogas) com dados reais;

(c) redes lineares por partes com demandas e limitantes de intervalos de fluxo inteiros versus redes lineares por partes (análogas) com dados reais;

(d) redes lineares por partes com custos inteiros versus redes lineares por partes (análogas) com dados reais;

Nestes testes são utilizados os geradores da categoria 1;

- *categoria 2*: comparação de desempenho entre o MSFV aplicado a redes lineares por partes pseudo-aleatórias e o MSFV aplicado a redes lineares equivalentes.

Nestes testes são utilizados os geradores da categoria 2;

- *categoria 3*: aplicação do MSFV a redes especiais, conhecidas da literatura, que apresentam dificuldades ao uso do método simplex tradicional.

São exemplos propostos por Cunningham /30/ e Zadeh /42/; nestes testes são utilizados os geradores da categoria 3;

- *categoria 4*: aplicação do MSFV com a Técnica de Mudança de Escala a redes lineares por partes pseudo-aleatórias e especiais.

São exemplos propostos por Ikura e Nemhauser /38/ e Zadeh /42/; nestes testes são utilizados os geradores das categorias 3 e 4.

Passa-se agora a apresentar os resultados dos diversos experimentos realizados, onde adotou-se a seguinte notação para os dados e resultados:

~(ℓ, u), indica distribuição uniforme no intervalo (ℓ, u);

~[ℓ.. u], indica que um dos valores ℓ, ℓ+1,..., u-1, u é escolhido com probabilidade $\frac{1}{u-\ell+1}$;

nitt, é o número de pivoteamentos do método simplex tradicional na rede linear equivalente; isto é, o número de execuções do Passo 3 do algoritmo;

nits, é o número de atualizações da base; isto é, o número de execuções do Passo 2 do algoritmo;

nitp, é o número de iterações do MSFV na rede linear por partes; isto é, o número de execuções do Passo 1;

nitd, é o número de pivoteamentos degenerados do MSFV na rede linear equivalente;

cpu, é o tempo de processamento (em segundos), não inclui o tempo para entrada de dados e saída de resultados.

3.5. TESTES DA CATEGORIA 1

Esta seção pode ser dividida em 4 (quatro) sub-seções que passa-se a descrever.

3.5.1. Redes Lineares por Partes com Dados sendo Números Reais com Nós de Transbordo e/ou Arcos com Fluxo Canalizado

Optou-se por aplicar o MSFV em redes que representassem os casos limites de PRLP gerais, através da alteração de alguns parâmetros de entrada no gerador. Estes parâmetros foram:

* probabilidade de haver algum nó de transbordo

  $p_0 = 0$, não há transbordo

  $p_0 = 1$, todas demandas são nulas

* probabilidade de haver arcos com fluxo canalizado

  $p_\infty = 0$, nenhum arco tem fluxo canalizado

  $p_\infty = 1$, todos os arcos tem fluxo canalizado

* custos nos intervalos de fluxo

  $c \sim (0,20)$, função objetivo linear por partes convexa estritamente crescente

  $c \sim (-5,15)$, função objetivo linear por partes convexa ("U-shaped")

Assim, nas tabelas 3.2 e 3.3 estão condensados resultados médios (para vários valores de sementes) para os seguintes tipos de redes:

Rede	Parâmetros de entrada no gerador								Características	
	m	n	p_0	b	k	d	p_∞	c		
A	100	1000	0	~(-100,100)	~[3..7]	~(0,70)	0	~(0,20)	sem transbordo e sem canalização	função objetivo convexa estritamente crescente
B	100	1000	0	~(-100,100)	~[3..7]	~(0,70)	1	~(0,20)	sem transbordo e com canalização	
C	100	1000	1	~(-100,100)	~[3..7]	~(0,70)	0	~(0,20)	com transbordo e sem canalização	
D	100	1000	1	~(-100,100)	~[3..7]	~(0,70)	1	~(0,20)	com transbordo e com canalização	
E	100	1000	0	~(-100,100)	~[3..7]	~(0,70)	0	~(-5,15)	sem transbordo e sem canalização	função objetivo "U-Shaped"
F	100	1000	0	~(-100,100)	~[3..7]	~(0,70)	1	~(-5,15)	sem transbordo e com canalização	
G	100	1000	1	~(-100,100)	~[3..7]	~(0,70)	0	~(-5,15)	com transbordo e sem canalização	
H	100	1000	1	~(-100,100)	~[3..7]	~(0,70)	1	~(-5,15)	com transbordo e com canalização	

Tabela 3.1. Redes para Comparações Gerais entre Regras de Entrada para PRLP

Percebe-se que, com relação ao número de iterações (nitp), o comportamento relativo das regras de entrada se mantêm praticamente inalterado (a menos das regras Bland e LRB, na rede D) podendo ser colocadas por ordem de melhor desempenho da forma: Dantzig, Nõ-Cíclica Continuada, Nõ-Cíclica, Cíclica, LRB e Bland (havendo uma alternância de posição entre a segunda e a terceira colocadas).

Já analisando os valores médios de cpu, nota-se uma vantagem das regras Cíclica, Nõ-Ciclica Continuada e Nõ-Cíclica sobre as demais, tendo a primeira se sobressaído como a melhor em 6 das 8 classes de redes investigadas. Quanto as outras regras, neste critério, no caso da função objetivo "U-Shaped" a ordenação é Dantzig, LRB e Bland (bem inferior as demais) e no caso da função objetivo estritamente crescente, esta ordenação se mantêm para as classes A e B (não degenerados) mas nas classes C e D (degenerados) a regra Dantzig tem seu desempenho prejudicado, ficando abaixo da LRB e Bland.

Uma justificativa para este desempenho diferenciado da regra Dantzig nas classes degeneradas C e D versus G e H, é que nas primeiras todas as iterações são degeneradas, não valendo a pena o esforço de escolher como candidato a entrar na base o arco com custo relativo (à esquerda ou à direita) que proporcione a maior taxa de decréscimo da função objetivo. Para as classes G e H nem todas as iterações são degeneradas (em média 5% do total de iterações para G e 9% para H) de modo que este esforço é válido no sentido de diminuir o total de iterações.

Regra \ Rede	cpu				nitp			
	A	B	C	D	A	B	C	D
Bland	80.07	77.53	5.58	5.13	5066	5253	317	285
Cíclica	7.75	10.43	0.93	0.92	1453	2015	258	253
Dantzig	25.43	32.41	7.68	7.32	436	570	151	143
Nó-Ciclica	8.19	9.82	1.24	1.16	1118	1437	171	157
Nó-Ciclica Continuada	8.28	9.68	1.22	1.17	1055	1387	169	157
LRB	73.92	73.69	5.57	5.43	3349	4059	300	295

Tabela 3.2. Desempenhos das Regras de Entrada em PRLP com Função Objetivo Convexa Estritamente Crescente

Regra \ Rede	cpu				nitp			
	E	F	G	H	E	F	G	H
Bland	328.07	130.90	286.02	189.64	23634	9228	20793	12838
Cíclica	13.50	13.29	12.00	8.96	2869	2794	2771	2203
Dantzig	58.85	54.22	58.38	45.88	1014	946	1003	814
Nó-Cíclica	14.02	12.94	13.43	10.65	2144	1940	2193	1642
Nó-Cíclica Continuada	14.95	12.82	13.22	10.64	2188	1952	2113	1640
LRB	81.23	91.67	83.10	83.56	5154	5419	5374	3924

Tabela 3.3. Desempenhos das Regras de Entrada em PRLP com Função Objetivo Convexa ("U-Shaped")

Percebe-se ainda que as regras Nó-Cíclica e Nó-Cíclica Continuada suplantam a Cíclica apenas para os problemas das classes B e F, onde não há transbordo e os arcos são canalizados.

Pode-se afirmar ainda que os problemas mais fáceis de resolver são os das classes C e D, onde ocorre degeneração total e a função objetivo é estritamente crescente.

Com relação a função objetivo, os problemas das classes "U-Shaped" são mais difíceis de resolver (maior cpu e nitp); os problemas canalizados tendem a ser mais difíceis que os não canalizados, o mesmo ocorrendo com os problemas sem transbordo em relação aos problemas com transbordo.

3.5.2. *Redes Lineares por Partes com Dados Inteiros versus Redes Lineares por Partes (análogas) com Dados Reais*

Esta seção tem por objetivo observar o comportamento das regras de entrada quando são utilizados dados inteiros ao invés de dados reais, o que implica no aumento do nível de degeneração do problema.

Apresenta-se na tabela 3.5 resultados médios (nitp, nitd, cpu) obtidos para vários valores de sementes, para uma classe de redes com os seguintes parâmetros:

Rede	Parâmetros de entrada no gerador								Características
	m	n	p_0	b	k	d	p_∞	c	sem transbordo
I	100	1000	0	~[-100..100]	~[3..7]	~[0..70]	1	~[0..20]	e com
B	100	1000	0	~(-100,100)	~[3..7]	~(0,70)	1	~(0,20)	canalização

Tabela 3.4. Redes para Comparações entre Regras de Entrada em PRLP : Dados Inteiros x Dados Reais

Como nos problemas da seção 3.5.1 com relação ao critério cpu, nota-se na classe de redes I uma vantagem flagrante das regras Cíclica, Nó-Cíclica e Nó-Cíclica Continuada sobre as demais, com a regra LRB na última posição. Esta semelhança se mantém também quanto ao número de iterações (nitp), apenas há uma inversão de posicionamento entre as regras Bland e LRB, ficando esta por último.

Regra \ Rede	nitp		nitd		cpu	
	B	I	B	I	B	I
Bland	5253	3760	0	1729	77.53	49.74
Cíclica	2015	1934	0	901	10.43	9.99
Dantzig	570	595	0	642	32.41	33.88
Nó-Cíclica	1437	1333	0	631	9.82	9.92
Nó-Cíclica Continuada	1387	1319	0	600	9.68	9.26
LRB	4059	3831	0	1744	73.69	62.05

Tabela 3.5. Desempenhos das Regras de Entrada em PRLP: Dados Inteiros x Dados Reais

Comparando as classes B (real) e I (inteira), percebe-se que nesta última ocorrem mais pivoteamentos degenerados (nitd) implicando em um decréscimo sistemático nos valores de cpu e nitp para todas as regras, a menos da regra Dantzig que novamente se comporta mal perante degeneração.

Deve-se citar que para outras classes de redes (mais densas, com m = 50 nós e n = 2000 arcos) este quadro se repetiu.

3.5.3. *Redes Lineares por Partes com Demandas e Limitantes de Intervalos de Fluxos Inteiros versus Redes Lineares por Partes (análogas) com Dados Reais*

Objetivou-se verificar a influência do número de algarismos significativos, dos valores das demandas e dos limitantes de intervalos de fluxo, no desempenho do MSFV quando combinado com as regras de entrada. Isto é útil para se inferir acerca do desempenho do MSFV com a Técnica de Mudança de Escala (TME), que pode atuar nestes casos.

Assim foram computados resultados médios (ver tabela 3.7), para várias sementes, relativos as seguintes classes de redes:

Rede	Parâmetros de entrada no gerador								Características
	m	n	p_0	b	k	d	p_∞	c	
J	100	1000	0	~[-10..10]	~[3..7]	~[0..7]	1	~(0,20)	sem transbordo e com canalização
K	100	1000	0	~[-100..100]	~[3..7]	~[0..70]	1	~(0,20)	
L	100	1000	0	~[-1000..1000]	~[3..7]	~[0..700]	1	~(0,20)	
B	100	1000	0	~(-100, 100)	~[3..7]	~(0,70)	1	~(0,20)	

Tabela 3.6. Redes para Comparação entre Regras de Entrada em PRLP: Demandas Inteiras x Demandas Reais

Para que seja viável o uso da TME, isto é, que o rendimento do MSFV seja melhorado, há necessidade de que nitp e/ou cpu cresçam a partir das redes J para as redes L e sejam inferiores ao valor de nitp e/ou cpu das redes B (reais). Ou seja, a TME é indicada a casos em que quanto mais degenerado o problema for, mais rápida é a sua resolução.

Percebe-se, consultando a tabela 3.7, que a grande dificuldade é determinar a partir de qual precisão (número de algarismos significativos) deve-se iniciar este processo de mudança de escala, nas demandas e limitantes de intervalos de fluxo, para que a TME seja útil.

Em nenhum caso nitp (cpu) da classe J foi inferior ao nitp da classe B indicando que esta precisão é prejudicial ao MSFV.

Além disto, apenas uma situação para o critério cpu (Regra Bland) e duas para o critério nitp (Regras Bland e Nó-Cíclica) indicam a possibilidade do uso da TME, a partir da precisão adotada para a classe k, para melhorar o desempenho do MSFV.

Para o caso de redes mais densas (m=50 nós e n=2000 arcos), o resultado obtido foi ainda mais desapontador, apenas uma situação para o critério cpu forneceu uma indicação de melhoria pelo uso da TME (também a partir da precisão da classe k).

Regra	nitp				cpu			
	J	K	L	B	J	K	L	B
Bland	5830	4526	4535	5253	85.77	63.84	66.92	77.53
Cíclica	2774	1986	1901	2015	11.29	10.48	10.32	10.53
Dantzig	857	610	563	570	46.94	35.11	33.68	32.41
Nó-Cíclica	1552	1316	1431	1437	10.28	9.42	10.54	9.82
Nó-Cíclica Continuada	1508	1376	1410	1387	9.83	8.44	10.18	9.68
LRB	5497	4457	4265	4059	118.39	77.76	81.77	73.69

Tabela 3.7. Desempenhos das Regras de Entrada em PRLP : Demandas Inteiras x Demandas Reais

Estes resultados levam a conclusão que é extremamente difícil que a TME, atuando sobre as demandas, auxilie o MSFV para problemas que possuam dados sendo números reais.

3.5.4. Redes Lineares por Partes com Custos Inteiros versus Redes Lineares por Partes com Dados Reais

Aqui investigou-se o comportamento das regras de entrada quando se modifica o número de algarismos significativos dos

custos nos intervalos de fluxo, como se houvesse algum tipo de precisão atuando nestes dados.

Para tanto, foram estudadas as seguintes classes de redes:

Redes	Parâmetros de entrada no gerador								Características
	m	n	p_0	b	k	d	p_∞	c	
M	100	1000	0	~(-100,100)	~[3..7]	~(0,70)	1	~[0..20]	sem transbordo e com canalização
N	100	1000	0	~(-100,100)	~[3..7]	~(0,70)	1	~[0..200]	
O	100	1000	0	~(-100,100)	~[3..7]	~(0,70)	1	~[0..2000]	
B	100	1000	0	~(-100,100)	~[3..7]	~(0,70)	1	~(0,20)	

Tabela 3.8. Redes para Comparação entre Regras de Entrada em PRLP: Custos Inteiros x Dados Reais

Quanto ao critério cpu, percebe-se que as regras Cíclica, Nó-Cíclica e Nó-Cíclica Continuada se destacam em todas as classes e à medida que se aumenta a precisão nos custos (classes N e O), as duas últimas melhoram seu desempenho suplantando a Cíclica, que é superior na classe M. O mesmo ocorre para as regras Bland e LRB (que são as de pior desempenho) com a LRB melhorando com o aumento da precisão.

Já quanto ao número de iterações, a menos da classe M onde há uma inversão de posição entre a LRB e Bland, a ordenação da melhor para a pior é: Dantzig, Nó-Cíclica Continuada, Nó-Cíclica, Cíclica, LRB e Bland.

Regra	nitp				cpu			
	M	N	O	B	M	N	O	B
Bland	3557	9049	15860	5253	49.34	108.0	201.11	77.53
Cíclica	1837	2372	2641	2015	10.04	12.18	13.19	10.43
Dantzig	594	517	515	570	36.04	31.25	30.99	32.41
Nó-Cíclica	1303	1525	1692	1437	10.33	10.07	11.02	9.82
Nó-Cíclica Continuada	1352	1562	1672	1387	10.29	10.32	11.16	9.68
LRB	3668	5682	6599	4059	62.47	76.19	92.03	73.69

Tabela 3.9. Desempenhos das Regras de Entrada em PRLP: Custos Inteiros x Dados Reais

Novamente, como no caso anterior, não parece interessante aplicar a TME nos custos para melhorar o desempenho do MSFV em problemas com dados reais (B) pois já para a classe de redes N o valor de nitp, para todas as regras (com exceção do Dantzig) supera aquele valor obtido para as redes reais.

Contudo, há indícios que este tipo de enfoque trabalhe bem em se tratando de redes com dados totalmente inteiros, pois observa-se que todas as regras (menos Dantzig) tem cpu e nitp aumentando com a melhoria na precisão dos custos (de M para O).

Uma justificativa para este comportamento da regra Dantzig é que com a melhoria da precisão, há um maior destaque para os arcos que realmente são candidatos a melhorar mais profundamente a solução, o que fica mascarado quando do uso de uma precisão menos acurada.

3.6. TESTES DA CATEGORIA 2

Nesta seção objetiva-se principalmente mostrar a vantagem de se possuir um algoritmo que possa ser utilizado diretamente numa rede linear por partes, ao invés de obter-se uma rede linear equivalente e nela aplicar o método simplex.

Para tanto, foi utilizado o gerador de redes pseudo-aleatórias que fornece, a partir de uma entrada de dados, três redes associadas: a rede linear por partes (RLP), a rede linear aproximada (RLA) e a rede linear equivalente (RLE), conforme exposto na figura 3.2.

As redes geradas, com dados reais, possuem as seguintes características:

Redes	Parâmetros de entrada no gerador								Características
	m	n	p_0	b	k	d	p_∞	c	
RLP	100	1000	0	~(-100,100)	~[7..7]	~[0,80]	1	~(0,20)	rede linear por partes sem transbordo e com canalização
RLA	100	1000	0	~(-100,100)	~[3..3]	~[0,80]	1	~(0,20)	rede linear sem transbordo e com canalização
RLE	100	5000	0	~(-100,100)	~[3..3]	~[0,80]	1	~(0,20)	rede linear sem transbordo e com canalização

Tabela 3.10. Redes para Comparações RLP x RLA x RLE

Deve ser observado que a RLP possue todos seus arcos com fluxo canalizado (pois $p_\infty=1$), cinco intervalos de fluxo viáveis (desconsiderando dos sete gerados aqueles dois fora da canalização) para cada arco e nenhum nó é de transbordo (pois $p_0=0$). A RLA possue o mesmo número de arcos canalizados que a RLP (n=1000), apenas um intervalo viável para cada arco, cujo custo é a média aritmética dos custos gerados para o arco associado na RLP. Quanto a RLE, ela tem n=5000 arcos (um arco associado a cada intervalo de fluxo viável da RLP), com um intervalo de fluxo viável cada, mantendo o custo do intervalo de fluxo associado (ver figura 3.2).

Utilizando-se valores diferentes para a semente, foram computadas informações médias acerca do nitt, nitp e cpu.

Todas as redes foram resolvidas pelo MSFV combinado com cada uma das regras de entrada implementadas; na tabela 3.11 apresenta-se os valores médios obtidos para três sementes distintas.

Medidas / Regra	RLE		RLA		RLP			
	nitp	cpu	nitp	cpu	nitt	nits	nitp	cpu
Bland	11897	609.39	1570	20.16	11849	11402	8355	119.06
Cíclica	4248	22.73	1091	4.45	5303	5106	2509	16.93
Dantzig	1858	263.61	360	19.52	2765	2629	616	40.03
Nó-Cíclica Continuada	3329	42.62	710	4.46	3700	3412	1616	14.71
Nó-Cíclica	3340	44.16	724	4.72	3582	3311	1510	13.60
LRB	18904	1720.21	2234	36.66	10531	9930	6259	110.65

Tabela 3.11. Desempenhos das Regras de Entrada: RLP x RLA x RLE

Observe-se que para as redes lineares (RLE e RLA) tem-se nitt = nits = nitp.

Inicialmente deve-se comentar que com respeito a RLP tem-se, para todas as regras de entrada, que nitt > nitp e nits > nitp significando que a característica (básica e/ou não-básica) continuada do MSFV não é aproveitada em se tratando da rede linear equivalente, ocasionando um número substancialmente maior de iterações e atualizações da árvore básica nesta última.

Como era de se esperar numa comparação entre RLP x RLA, a vantagem é toda para a rede linear aproximada, já que as duas tem a mesma dimensão (arcos e nós) mas a RLA possue apenas três intervalos de fluxo contra sete da RLP e além disto o custo no intervalo de fluxo viável da RLA é um valor médio dos custos (viáveis) da RLP.

Comparando-se os resultados da RLP contra os da RLE, fica evidente a inconveniência de se tratar uma rede linear por partes através da rede linear equivalente, bastando observar a discrepância entre os valores de nitp e cpu para cada tipo de problema. Ou seja, o número de iterações (e tempo de processamento) é muito sensível ao aumento do número de arcos (1000 para RLP e 5000 para RLE).

Quanto as regras de entrada percebe-se que os seus desempenhos relativos mantiveram-se praticamente os mesmos em todos os problemas. Quanto ao número de iterações a que melhor se comportou foi a Dantzig, seguida das regras Nó-Cíclica e Nó-Cíclica Continuada, que se alternaram em melhor desempenho, com resultados um pouco mais defasados aparece a regra Cíclica e com comporta

mento bastante ruim têm-se as regras Bland e LRB.

Já com respeito ao tempo de processamento, a ordenação da melhor para pior tem alterações. Aparecem com destaque as regras Nó-Cíclica, Nó-Cíclica Continuada (para a RLP) e Cíclica (para RLE e RLA), seguidas (já de longe) pela regra Dantzig e por último as regras Bland e LRB, novamente com desempenho fraco neste critério.

Pode-se comentar ainda que, as regras Nó-Cíclica e Nó-Cíclica Continuada são mais sensíveis que a regra Cíclica ao aumento no grau (externo e interno) dos nós da rede, tornando-se bem mais lentas (em torno de 50% contra apenas 24% para Cíclica) com o aumento de arcos incidentes em cada nó (isto pode ser observado comparando-se os desempenhos relativos destas regras com respeito as redes RLE e RLA).

Já com relação ao número de intervalos de fluxo para cada arco, a situação se inverte, com a regra Cíclica perdendo em torno de 40% de seu desempenho contra 31% e 28%, respectivamente para Nó-Cíclica e Nó-Cíclica Continuada (basta comparar os desempenhos relativos destas regras com respeito as redes RLP e RLA).

3.7. TESTES DA CATEGORIA 3

Aqui a preocupação é mostrar o desempenho do MSFV com regras de entrada que evitam o empacamento (ver Teorema 4, capítu

lo 2) comparativamente com o uso de regras que não tem esta propriedade, em redes especiais conhecidas da literatura (ver exemplo 2.4, capítulo 2), que apresentam tal fenômeno.

Além disto, utiliza-se o MSFV em redes que se caracterizam por ocasionarem um desempenho ruim para o método simplex tradicional (ver exemplo 2.5, capítulo 2), nestes problemas ocorre, para as regras que não evitam empacamento (Bland, Dantzig), um fenômeno com características parecidas com o de empacamento (onde há somente pivoteamentos degenerados) só que envolvendo pivoteamentos não-degenerados. Na seção 3.6 volta-se a enfocar estes problemas com a Técnica de Mudança de Escala.

Como foi visto na seção 2.3, para identificação de ocorrência de empacamento, deve-se apresentar resultados do algoritmo em estudo quando aplicados a uma família de problemas. Assim sendo, foram geradas redes do tipo proposto por Cunningham, como as do exemplo 2.4, para vários valores consecutivos do parâmetro p, que está ligado à dimensão do problema (número de nós e arcos). Na tabela 3.12 apresenta-se os valores de nitp e cpu para estas redes quando resolvidas através do MSFV combinado com várias regras de entrada.

Para cada valor de p tem-se uma rede com número de nós $m = 2p$ e número de arcos $n = 4p-2$. Pode-se observar que a Regra Bland, que não evita empacamento, se comporta exatamente como previsto por Cunningham /30/, com respeito ao tamanho da sequência de empacamento, ou seja, para o problema p se tem uma sequência (exponencial) com $3 \cdot 2^p - 2p - 2$ árvores básicas degeneradas.

Já as regras Cíclica e Nó-Cíclica, bem como a Nó-Cícli

ca Continuada e a LRB que tem comportamento idêntico e não são expostas na tabela 3.12, tem um desempenho (polinomial) muito bom tanto no número de iterações necessárias para a resolução (nitp=2p-1) quanto no tempo de processamento. A regra de Dantzig também repete este desempenho.

Regra de Entrada / parâmetro	Bland		Cíclica		Nó-Cíclica		Tamanho da Sequência de Empacamento
	nitp	cpu	nitp	cpu	nitp	cpu	
2	5	0.02	3	0.01	3	0.01	6
3	15	0.03	5	0.01	5	0.02	16
4	37	0.06	7	0.02	7	0.02	38
5	83	0.13	9	0.02	9	0.02	84
6	177	0.29	11	0.02	11	0.02	178
7	367	0.63	13	0.03	13	0.04	368
8	749	1.33	15	0.03	15	0.04	750
9	1515	2.75	17	0.04	17	0.05	1516
10	3049	5.66	19	0.05	19	0.06	3050
15	98271	196.09	29	0.08	29	0.08	98272

Tabela 3.12. Desempenhos das Regras de Entrada Bland, Cíclica e Nó-Cíclica em Redes com Empacamento

Na tabela 3.13 estão colocados os resultados relativos ao uso do MSFV e das seis regras de entrada na resolução de vários problemas da família proposta por Zadeh /42/; compara-se com os resultados citados por este autor para o método simplex tradicional.

Regras de Entrada / parâmetro p	Bland		Cíclica		Dantzig		Nõ-Cíclica		Nõ-Cíclica Continuada		LRB		Simplex tradicional
	nitp	cpu	nitp	cpu	nitp	cpu	nitp	cpu	nitp	cpu	nitp	cpu	nitp
3	10	0.03	9	0.03	10	0.04	5	0.02	8	0.03	5	0.02	8
4	22	0.07	15	0.04	22	0.09	7	0.03	13	0.05	7	0.02	18
5	46	0.16	23	0.04	46	0.26	9	0.04	19	0.06	9	0.03	38
6	94	0.36	33	0.07	94	0.64	11	0.05	26	0.10	11	0.04	78
7	190	0.73	45	0.10	190	1.66	13	0.06	34	0.11	13	0.05	158
8	382	1.53	59	0.12	382	4.07	15	0.06	43	0.14	15	0.05	318
expressões gerais para nitp	$3 \cdot 2^{p-1} - 2$		$p^2 - p + 3$		$3 \cdot 2^{p-1} - 2$		$2p - 1$		$\frac{p^2 + 3p - 2}{2}$		$2p - 1$		$2^p + 2^{p-2} - 2$

Tabela 3.13. Desempenhos das Regras de Entrada em Redes de Zadeh

Pode-se constatar que, o MSFV com as regras de Bland e Dantzig tem nitp dado por expressões que crescem exponencialmente com o aumento do parâmetro p e de forma a ser pior que o método simplex tradicional; já as demais regras de entrada (que evitam empacamento) tem este número de iterações como função polinomial de p.

Deve-se comentar que na geração das redes os nós e arcos são ordenados de forma a prejudicar as regras Cíclica e Nó-Cíclica Continuada, com conseqüente vantagem para a Nó-Cíclica e LRB. Assim têm-se limitantes, para o melhor e pior caso de ordenação dos arcos e nós, com respeito ao número de iterações necessárias para resolver uma rede de Zadeh /42/.

3.8. *APLICAÇÃO DO MSFV COM A TÉCNICA DE MUDANÇA DE ESCALA A REDES LINEARES POR PARTES PSEUDO-ALEATÓRIAS E ESPECIAIS*

Apresenta-se, nas próximas sub-seções, os resultados obtidos para implementações do MSFV com a Técnica de Mudança de Escala, para as duas situações analisadas.

3.8.1. *Redes Lineares por Partes Pseudo-Aleatórias*

Adotou-se a sistemática de gerar redes com diversas configurações, aplicar o MSFV com e sem a Técnica de Mudança de

Escala (TME) atuando nas demandas (e limitantes de intervalos), e comparar os resultados. Nesta etapa foram utilizados os mesmos geradores da seção 3.4, mas os resultados não foram animadores, pois o MSFV sem TME apresentou resultados superiores, mas não consistentes. Este fato já era esperado, conforme destacado na seção 3.5.3.

A seguir, tendo em vista a referência Ikura e Nemhauser /38/, passou-se a enfocar redes com as características descritas por aqueles autores, ou seja, redes bipartites inteiras.

Numa primeira etapa, estudou-se o escalonamento das demandas (e limitantes de intervalos), conforme exposto no capítulo 2. Novamente os resultados foram insatisfatórios, pois não se conseguiu, para nenhuma das regras implementadas, melhorar o comportamento do MSFV através do uso da TME.

Numa segunda etapa, aplicou-se a TME nos custos dos intervalos obtendo-se alguns resultados interessantes, que passa-se a comentar.

Os dados referentes a redes geradas estão colocados na tabela 3.14, devendo ser observado que as classes A1, A2, A3 são as mesmas usadas na referência /38/; isto é, são redes lineares; e as classes A1LP, A2LP, A3LP são as suas correspondentes na Programação Linear por Partes.

Foi constatado que, para todos os tipos de redes geradas, a única regra que teve seu desempenho melhorado pela TME foi a Regra Cíclica. Os seus resultados estão expostos na tabela 3.15.

Classes de Redes	Parâmetros de entrada no gerador								Características
	m_1	m_2	n	DT	k	d	p_∞	c	
A1	100	100	2000	5000	~[2..2]	~[1..70]	0	~[1..100]	redes bipartites lineares, sem canalização
A2	200	200	4000	10000	~[2..2]	~[1..70]	0	~[1..100]	
A3	400	400	6000	10000	~[2..2]	~[1..70]	0	~[1..100]	
A1LP	100	100	2000	5000	~[3..5]	~[1..70]	0	~[1..100]	redes bipartites lineares por partes sem canalização
A2LP	200	200	4000	10000	~[3..5]	~[1..70]	0	~[1..100]	
A3LP	400	400	6000	10000	~[3..5]	~[1..70]	0	~[1..100]	

Tabela 3.14. Redes Bipartites Aleatórias para Uso da Regra Cíclica com o MSFV/TME (custos)

Numa avaliação global em termos de nitp, percebe-se que para as classes de redes lineares A1, A2, A3, apenas para a primeira (A1) conseguiu-se melhorias em todos os problemas gerados; sendo que nas classes A2 e A3 as dificuldades aumentaram e os resultados não foram satisfatórios, pois somente 2 problemas (em 6 gerados) e 1 problema (em 5 gerados) apresentaram vantagem do uso da TME, respectivamente para as classes A2 e A3.

Já com respeito as classes de redes lineares por partes, a situação foi ainda pior, com somente a classe A1LP mantendo a característica de melhorar o desempenho da Regra Cíclica (num nível abaixo da sua correspondente A1); na classe A2LP apenas 1 problema (em 6 gerados) mostrou vantagem no uso da TME e na classe A3LP, nenhum dos 5 problemas gerados apresentou condições de melhoria.

Deve-se ser comentado que, para todas as classes de redes analisadas, houve um acréscimo no tempo de cpu quando se utilizou a TME. É possível, num estudo posterior, tornar a implementação do MSFV com este refinamento mais eficiente, pois foi adotada a idéia básica de se trabalhar com uma estrutura de dados flexível que permitisse a incorporação da análise de pós-otimalidade e também permitisse o acoplamento ao MSFV de diversas regras de entrada.

Rede	A1		A2		A3		A1LP		A2LP		A3LP	
versão do algoritmo	nitp	cpu	nitp	cpu	nitp	cpu	nitp	cpu	nitp	cpu	nitp	cpu
MSFV/TME	1667	27.99	4720	101.18	10694	321.48	3523	59.32	9307	221.46	15583	540.89
MSFV	1778	18.31	4712	76.12	10478	272.87	3646	44.72	8836	178.67	14486	443.01
número de redes geradas	5		6		5		5		6		5	
número de redes com vantagem para a TME	5		2		1		5		1		0	
porcentagem melhoria no nitp, com a TME	6%		0%		-2%		3.5%		-5%		-8%	

Tabela 3.15. Resultados da Regra Cíclica com o MSFV/TME (custos) para Redes Bipartites

3.8.2. Redes Especiais

O MSFV aliado à Técnica de Mudança de Escala nas demandas e limitantes de intervalos foi aplicado, também, as redes de Zadeh /42/ a título de investigação. Os resultados estão na tabela 3.16, colocados comparativamente com o desempenho do algoritmo de Edmonds e Karp /36/ com a TME e também o método simplex tradicional.

Percebe-se, consultando a tabela 3.13 referente ao uso do MSFV sem a TME a estas mesmas redes, que neste caso a TME melhora sensivelmente o desempenho das regras Bland, Cíclica, Dantzig e Nõ-Cíclica Continuada e não prejudica as regras Nõ-Cíclica e LRB. Mais ainda, estes desempenhos são superiores aos do método Edmonds e Karp/TME e do método simplex tradicional, justificando assim a incorporação deste refinamento no MSFV neste caso.

Aparentemente, destas redes pode-se inferir que o MSFV/TME é útil para situações em que as regras de entrada não estão se desempenhando bem, para cada tipo de regra de entrada considerada. Isto justifica, em parte, os resultados insatisfatórios obtidos na seção 3.8.1.

Regra de Entrada / Parâmetro p	Bland		Cíclica		Dantzig		Nó-Cíclica		Nó-Cíclica Continuada		LRB		Edmonds e Karp	Simplex
	nitp	cpu	nitp	cpu	nitp	cpu	nitp	cpu	nitp	cpu	nitp	cpu		
3	7	0.04	7	0.05	7	0.05	5	0.03	7	0.06	5	0.03	5	8
4	11	0.08	10	0.07	11	0.11	7	0.08	10	0.08	7	0.07	21	18
5	16	0.15	13	0.11	16	0.16	9	0.12	13	0.16	9	0.10	28	38
6	22	0.24	16	0.16	22	0.28	11	0.16	16	0.17	11	0.16	35	78
7	29	0.34	19	0.21	29	0.44	13	0.23	19	0.21	13	0.21	42	158
8	37	0.48	22	0.30	37	0.59	15	0.28	22	0.30	15	0.27	49	318
expressões gerais para nitp	$\frac{p^2+p+2}{2}$		$3p-2$		$\frac{p^2+p+2}{2}$		$2p-1$		$3p-2$		$2p-1$		$4p-7$	$2^p+2^{p-2}-2$

Tabela 3.16. Comparações nas Redes de Zadeh : MSFV/TME (demandas) x Edmonds e Karp/TME x Simplex

CAPITULO 4

CONCLUSÕES FINAIS

Neste capítulo, inicialmente são feitos comentários gerais abordando vários aspectos e dificuldades encontradas no desenvolvimento deste trabalho, bem como uma avaliação do que se conseguiu tendo em vista os objetivos estabelecidos como meta. Na seqüência, são discutidos alguns tópicos que podem ser trabalhados tendo como base esta dissertação.

4.1. COMENTÁRIOS GERAIS E VERIFICAÇÃO DOS OBJETIVOS

Cabe fazer-se algumas observações finais acerca do que foi feito neste trabalho, principalmente sobre as implementações realizadas.

Um primeiro comentário é com relação a estrutura de dados utilizada. Um interessante estudo feito por Ali et alli /34/ descreve as diversas estruturas de dados usadas em implementações do método simplex em redes. A estrutura escolhida é das mais simples, tem sido utilizada por outros pesquisadores, e possibilita facilmente a incorporação de procedimentos de reotimização (vide seção 4.2).

Outro detalhe, é a definição do valor do parâmetro inft($=10^8$), usada na atribuição do primeiro valor da tolerância (equivalente à Fase I do Simplex) para a primeira versão do MSFV. Idealmente, este valor deve ser atribuído pelo usuário. Entretanto, uma escolha indevida deste parâmetro provoca sérios problemas numéricos que comprometem a aplicação do algoritmo, apesar da preocupação que se teve em minimizar estes efeitos nas diversas operações realizadas pelo MSFV. Por exemplo, a ordem em que são realizadas operações envolvendo várias parcelas, aquelas que podem envolver parcelas de maior ordem de grandeza são efetuadas antes, de forma a evitar a proliferação de erros de arredondamento.

Com relação aos resultados computacionais e a sua análise são feitas as seguintes observações.

Considerando a impossibilidade física de executar todos os problemas ao mesmo tempo, o nível crescente de saturação que o sistema VAX da UNICAMP vem apresentando e a ordem de grandeza dos tempos de cpu observados, é conveniente desconsiderar diferenças nestes tempos inferiores a 1 (um) segundo.

Numa análise global, com respeito as regras de entrada implementadas, deve-se destacar o bom comportamento (homogêneo) da regra Cíclica em todos os experimentos realizados, inclusive alguns com a TME; as regras Nó-Cíclica e Nó-Cíclica Continuada também tiveram bom desempenho, a menos nos casos de TME aplicada a redes bipartites. Quanto as regras Bland e LRB pode-se afirmar que não é recomendável o seu uso, pois foram sistematicamente as que mais tempo de execução e número de iterações demandaram. Já a regra Dantzig, ofereceu a melhor opção em termos de número de

iterações (a menos nos casos em que a TME foi usada) mas no tempo de execução foi superada por outras regras. Um detalhe a ser observado é que esta regra cai de desempenho à medida que aumenta o grau de degeneração dos problemas enfocados.

Quanto aos experimentos abordando as redes propostas por Cunningham (que apresentam empacamento para a regra Bland), tem-se a dizer que apesar da regra Dantzig, neste caso, ter um comportamento idêntico ao das regras que satisfazem o Teorema 4 (isto é, evitam empacamento), não se tem uma demonstração formal de que isto se repita para redes gerais, mesmo aquelas que sejam apenas lineares. Uma explicação para o caso em questão, é que os custos dos arcos, das redes propostas por Cunningham, estão ordenados de forma a favorecer a aplicação deste regra, no sentido que um arco que entra na base ali permanece até o final de execução do MSFV. Já para as redes de Zadeh percebe-se que esta regra não se comporta adequadamente, se comparando à regra Bland.

Deve-se ressaltar, que com respeito as implementações utilizando a TME, os testes levados a efeito visam abrir uma nova linha de pesquisa em Programação Linear por Partes, e não objetivavam esgotar esta abordagem o que necessitaria de um estudo mais amplo. Apesar disto, cumpre dizer que obteve-se sucesso quanto a obtenção de limitantes polinomiais para o número de iterações, ao se usar o MSFV/TME. Isto pode ser constatado no caso das redes propostas por Zadeh, onde a TME melhorou sensivelmente o desempenho das regras Bland e Dantzig, passando-se de limitantes exponenciais ($3.2^{p-1}-2$) para polinomiais ($\frac{p^2 + p+2}{2}$).

Tem-se a dizer ainda, que foram seguidas as normas propostas por Crowder, Dembo e Mulvey /46/(1978) e recomendadas pela Mathematical Programming Society para a publicação de experimentos computacionais em Programação Matemática.

Finalizando, deve-se destacar uma série de aplicações reais da PRLP que vem sendo desenvolvidas no Departamento de Engenharia de Sistemas da FEE-UNICAMP, principalmente em convênio com a TELEBRÁS, abordando problemas de expansão de redes telefônicas (/47/ 1984, /48/ 1987).

4.2. FUTURAS DIREÇÕES DE PESQUISA

Passa-se a comentar algumas linhas de pesquisa possíveis de serem desenvolvidas em PRLP.

4.2.1. Procedimentos de Reotimização no MSFV

Uma continuação natural deste trabalho consiste na análise de pós-otimalidade de Programas em Redes Lineares por Partes. Mesmo para redes lineares, esta questão não vem tendo a devida atenção, fato este destacado por Ali, Allen, Barr e Kennington /49/(1986).

A análise de pós-otimalidade abrange dois aspectos: Análise de Sensibilidade e Reotimização após mudanças nos dados do problema.

A Análise de Sensibilidade admite incerteza nos dados e investiga a estabilidade da solução ótima com respeito a alterações em coeficientes do problema. O seu objetivo é determinar um intervalo de variação de um determinado coeficiente em estudo que não cause mudanças qualitativas na solução ótima.

Já os procedimentos de reotimização são úteis para se conhecer o impacto causado ao problema, pela alteração de seus dados além do intervalo de variação.

A seguir são apresentadas sugestões para implementação, no MSFV, destes procedimentos de reotimização, no sentido de motivar a pesquisa nesta área.

Cumpre destacar dois procedimentos que fazem parte do MSFV, que serão citados na exposição que se segue.

Procedimento *SOLVE* : Resolve o sistema de restrições (Mx = b) obtendo os valores de fluxo em cada arco, e também calcula o valor dos preços (variáveis duais) em cada nó. É constituído por 3 subprocedimentos : SOLVE 1, SOLVE 2, SOLVE 3.

Observação: tred(j) é o "label" *thread* associado ao nó j;

pred(i) é o "label" *predecessor* associado ao nó i, conforme descrito no capítulo 3.

```
{SOLVE 1}
1.  set   p(1): = raiz;
2.  set   j: = raiz;
3.  for   i: = 2 to m      do     p(i): = tred(j);  j:= p(i) ;
4.  for   nõ: = 1 to m     do     fr(nõ): = 0 ;
5.  for   (cada arco não-básico)  do
          fr(cabeça (arco)): = fr (cabeça (arco)) - fluxo (arco);
          fr(cauda  (arco)): = fr (cauda  (arco)) + fluxo (arco);

{SOLVE 2}
6.  for   j: = m downto  2   do
          arco: = abs (pred(p(j)));
          if  pred (p(j)) < 0
              then  fluxo (p(j)): = -(b(p(j)) + fr(p(j)));
                    fr (cabeça(arco)): = fr (cabeça(arco)) - fluxo (p(j))
              else  fluxo (p(j)): = b(p(j)) + fr(p(j)) ;
                    fr (cauda(arco)): =  fr (cauda(arco))  + fluxo (p(j));
7.  for   (cada arco básico)  do  atualizar intervalo corrente  do
                                  arco de modo a se ter um vetor
                                  básico h fortemente viável;

{SOLVE 3}
8.  set   nõ: = raiz;
9.  set   pi (nõ): = 0;
```

```
10.  repeat   j; = tred(nõ); arco: = abs (pred(j)); itv: = intervalo (arco);
              if  pred(j) > 0
                  then  pi(j): = pi (cauda(arco)) + custo (itv);
                  else  pi(j): = pi (cabeça(arco)) - custo (itv);
              set  nõ: = j ;
     until  tred (nõ): = raiz;
```

Procedimento *ENTRA* : Verifica qual arco é candidato a entrar na base;

Admite-se que se dispõe de uma solução ótima para um PRLP, e há interesse em se fazer alterações (nas demandas, nos limitantes de intervalos de fluxos ou nos coeficientes de custo) na rede original e depois reotimizá-la. Para cada caso, são fornecidas as etapas a serem seguidas para se obter uma nova solução ótima (para o problema modificado) a partir de uma solução ótima disponível usando a primeira versão do MSFV.

(a) Modificação no vetor de demandas (b) nos nós.

Seja $b' = (b'_i)$ o novo vetor de demandas

```
1.  for  i: = 1 to m  do  fr (i): = b'(i);
2.  call  SOLVE 1;
3.  call  SOLVE 2;
4.  call  MSFV;
```

Observação: O MSFV dispõe de uma solução básica inicial não necessariamente viável.

(b) Modificação em coeficiente de custo de intervalo de fluxo (c_j) para um arco a_j

(b_1) Se o arco a_j é básico na solução ótima disponível :

1. call SOLVE 2;
2. call MSFV;

Observação: O MSFV dispõe de uma solução básica fortemente viável inicial.

(b_2) Se o arco a_j é não-básico na solução ótima disponível:

1. call ENTRA;
2. *if* (a_j é candidato) *then* call MSFV;

Observação: O MSFV dispõe de uma solução básica fortemente viável inicial.

(c) Modificação em elemento do vetor de limitantes de intervalos de fluxo (d^j) de um arco a_j

Seja d'^j_k o valor do limitante alterado.

(c_1) Se o arco a_j é básico na solução disponível:

1. Verificar se o fluxo no arco a_j permanece no mesmo intervalo em que estava na solução ótima atual;
2. *if* (fluxo no mesmo intervalo)
   then stop (pois solução permaneceu ótima)
   else atualizar intervalo de modo que h permaneça fortemente viável;
3. call SOLVE 3;
4. call MSFV;

Observação: O MSFV dispõe de uma solução básica inicial não necessariamente viável.

(c_2) Se o arco a_j é não-básico na solução ótima disponível:

1. Verificar se o fluxo no arco a_j é igual ao valor do limitante d_k^j que foi alterado;
2. *if* (fluxo em outro limitante)
   then stop (pois a solução permanece ótima)
   else set fluxo (a_j) := d'^j_k ;
3. call SOLVE;
4. call MSFV.

Observações: Naturalmente as alterações a serem feitas nos dados, devem manter inalterados a propriedade de convexidade da função objetivo (casos (b) e (c)) e a condição $\sum_{i \in N} b(i) = 0$ (caso (a)).

4.2.2. *Um Algoritmo Out-of-Kilter para PRLP*

Um outro método importante para resolução de PRLP, é uma extensão do método Out-of-Kilter, desenvolvido por Fulkerson /50/(1961) para Programas em Redes Lineares.

Justifica-se esta opção, pelo fato de que apesar de várias comparações computacionais terem sido efetuadas com método primal, dual e out-of- kilter para redes lineares (ver Barr, Glover e Klingman /51/(1974), Glover, Karney e Klingman /25/(1974), Glover e Klingman /52/(1978)), ainda não se concluiu sobre qual tipo de algoritmo é superior aos demais.

A apresentação que é feita a seguir baseia-se na versão do método out-of-kilter proposta por Kennington e Helgason/33/.

Conforme exposto no capítulo 2, para a PRLP têm-se as seguintes formulações para os problemas Primal (P), Dual (D) e as Condições de Folgas Complementares (CFC), incorporando a determinação do vetor de custos correntes (inclinações de referência) z:

P : $\min \{f(x) \quad s.a: \quad Mx = b, \quad x \geq 0 \}$

com f(x) linear por partes;

D : $\max \{yb - g(z) \quad s.a: \; y\,M \leq z \;, \quad z \geq c^1\}$

com g(z) linear por partes (conjugada de f(x));

CFC: $d^j_{k-1} < x_j < d^j_k \implies c^j_k = y_{a^h_j} - y_{a^t_j}$;

$c^j_k < y_{a^h_j} - y_{a^t_j} < c^j_{k+1} \implies x_j = d^j_k$. (supor $c^j_k = -\infty$ se não definido).

$z_j = \max \{c^j_1 \;, \; y_{a^h_j} - y_{a^t_j} \}$

O método é inicializado com uma solução qualquer (x,y,z) que satisfaça $Mx = b$, $x \geq 0$, $z_j = \max \{c^j_1 \;, \; y_{a^h_j} - y_{a^t_j} \}$ para todo a_j. Dado um arco a_j, se são satisfeitas as CFC então diz-se que a_j está *In Kilter* (IK); caso contrário diz-se que a_j está *Out-of-Kilter* (OK).

Associa-se a cada arco a_j um valor, denominado *Número de Kilter* (NK_j), que corresponde quanto o fluxo x_j deve ser

alterado para que o arco a_j fique (ou permaneça) IK.

Assim a cada solução (x, y, z) corresponde o *Número de Kilter da Solução* (NK), dado pela soma dos números de kilter de cada arco da rede, isto é, $NK = \sum_{a_j \in A} NK_j$.

A estratégia básica do método out-of-kilter é efetuar alterações em x (fase primal) ou em y (fase dual) de modo a obter uma solução em que NK=0 (ou seja, onde todos os arcos estão IK). Estas alterações são de tal tipo que NK_j (para todo arco a_j) nunca aumenta e em cada iteração algum NK_j é reduzido.

Para a PRLP pode-se estabelecer que:

se $c_k^j < z_j < c_{k+1}^j$ então $h_j = k$ (útil adiante) e $NK_j = \begin{cases} x_j - d_k^j & \text{para } x_j > d_k^j \\ 0 & \text{para } x_j = d_k^j \\ d_k^j - x_j & \text{para } x_j < d_k^j \end{cases}$

se $z_j = c_k^j$ então $h_j = k$ e $NK_j = \begin{cases} d_{k-1}^j - x_j & \text{para } x_j < d_{k-1}^j \\ 0 & \text{para } d_{k-1}^j \leq x_j \leq d_k^j \\ x_j - d_k^j & \text{para } d_k^j < x_j \end{cases}$

e $\bar{c}_j$ (útil adiante) é determinado por

se $d_{k-1}^j < x_j < d_k^j$ então $\bar{c}_j = c_k^j - z_j$

$$\text{se } x_j = d_k^j \quad \text{então} \quad \bar{c}_j = \begin{cases} c_k^j - z_j & \text{para} \quad z_j < c_k^j \\ 0 & \text{para} \quad c_k^j \leq z_j \leq c_{k+1}^j \\ c_{k+1} - z_j & \text{para} \quad c_{k+1}^j < z_j \end{cases}$$

Observação: Um arco a_j está no estado IK se e somente se $\bar{c}_j \cdot NK_j = 0$.

Estas condições podem ser expressas através da esquematização da tabela 4.1.

z_j \ x_j	. . .	$x_j = d_{k-1}^j$	$d_{k-1}^j < x_j < d_k^j$	$x_j = d_k^j$	. . .
⋮	⋮	⋮	⋮	⋮	⋮
$c_{k-1}^j < z_j < c_k^j$	OK	IK (0)	OK ($x_j - d_{k-1}^j$)	OK ($x_j - d_{k-1}^j$)	OK
$z_j = c_k^j$	OK	IK (0)	IK (0)	IK (0)	OK
$c_k^j < z_j < c_{k+1}^j$	OK	OK ($d_k^j - x_j$)	OK ($d_k^j - x_j$)	IK (0)	OK
⋮	⋮	⋮	⋮	⋮	⋮

Tabela 4.1. Esquema para Condições Possíveis do Arco a_j, IK ou OK, e Valores de Número de Kilter, (NK_j)

Passa-se agora a descrever as etapas de aplicação do mētodo.

Fase Primal : Os valores de y são mantidos fixos.

Seja a_s um arco OK, procura-se encontrar um ciclo que inclua a_s e permita um aumento de fluxo, respeitando a tabela 4.1.

0. *set* $T = \emptyset$;

if $\bar{c}_s > 0$ *then* $V \leftarrow \{a_s^t\}$; $\Delta_{a_s} t := NK_s$

else $V \leftarrow \{a_s^h\}$; $\Delta_{a_s} h := NK_s$;

1. *let* $\psi_1 = \{a_r \neq a_s \ , \ \bar{c}_r < 0 \ , \ a_r^t \notin V, \ a_r^h \in V \ , \ x_r < d_{h_r}^r \}$;

$\psi_2 = \{a_r \neq a_s \ , \ \bar{c}_r = 0 \ , \ a_r^t \notin V, \ a_r^h \in V \ , \ x_r < d_{h_r}^r \}$

$\psi_3 = \{a_r \neq a_s \ , \ \bar{c}_r > 0 \ , \ a_r^t \in V, \ a_r^h \notin V \ , \ x_r > d_{h_r}^r \}$;

$\psi_4 = \{a_r \neq a_s \ , \ \bar{c}_r = 0 \ , \ a_r^t \in V, \ a_r^h \notin V \ , \ x_r > d_{h_r-1}^r \}$;

if $(\psi_1 \cup \psi_2 \cup \psi_3 \cup \psi_4 = \emptyset)$ *then* stop (não existe ciclo)

else let $a_r \in \psi_1 \cup \psi_2 \cup \psi_3 \cup \psi_4$;

2. *case of*

$a_r \in \psi_1$: $\Delta_{a_r} t := \min \{\Delta_{a_r} h \ , \ d_{h_r}^r - x_r\}$; $V \leftarrow V \cup \{a_r^t\}$; $T \leftarrow T \cup \{a_r\}$;

$a_r \in \psi_2$: $\Delta_{a_r} t := \min \{\Delta_{a_r} h \ , \ d_{h_r}^r - x_r\}$; $V \leftarrow V \cup \{a_r^t\}$; $T \leftarrow T \cup \{a_r\}$;

$a_r \in \psi_3$: $\Delta_{a_r^h} := \min \{\Delta_{a_r^t}, x_r - d^r_{h_r}\}$; $V \leftarrow V \cup \{a_r^h\}$; $T \leftarrow T \cup \{a_r\}$;

$a_r \in \psi_4$: $\Delta_{a_r^h} := \min \{\Delta_{a_r^t}, x_r - d^r_{h_r - 1}\}$; $V \leftarrow V \cup \{a_r^h\}$; $T \leftarrow T \cup \{a_r\}$;

if $\{a_s^t, a_s^h\} \not\subset V$ *then* go to 1 ;

3. *if* $\bar{c}_s > 0$ *then* aumentar fluxo no ciclo de $\Delta_{a_s^h}$

else aumentar fluxo no ciclo de $\Delta_{a_s^t}$.

Fase Dual : Se a fase primal termina com a conclusão que não há ciclo, tenta-se reduzir NK através do ajuste nas variáveis duais y, fixando os fluxos x. É utilizada a árvore (V, T) encontrada na fase primal.

1. *let* $\psi_1 = \{a_j : a_j^h \in V, a_j^t \notin V \text{ e } \bar{c}_j > 0\}$;

$\psi_2 = \{a_j : a_j^t \notin V, a_j^h \in V \text{ e } \bar{c}_j < 0\}$.

2. *set* $\theta := \min \{|\bar{c}_j| : a_j \in \psi_1 \cup \psi_2\}$;

3. *for* $i \in V$ *do* $y_i := y_i - \theta$;

for $j = 1$ to n *do* $z_j := \max \{y_{a_j^h} - y_{a_j^t} ; c_j^1\}$

Algoritmo Out-of-Kilter para PRLP:

0. *let* (x, y, z) uma solução para PRLP satisfazendo as condições

$Mx = b$, $x \geq 0$ e $z_j = \max \{y_{a_j^h} - y_{a_j^t}, c_j^1\}$ para todo a_j.

1. *if* (todos arcos IK) *then* stop (x, y, z) é uma solução ótima;

   let a_s arco OK;

2. Executar Fase Primal com a_s;

   if (não existe ciclo) *then* go to 3

   else go to 1;

3. Execute a Fase Dual com a árvore (V, T) construída na Fase Primal;

   if (a_s continua OK) *then* go to 2

   else go to 1.

É interessante frizar que, no caso em que todos os dados são inteiros, num número finito de iterações este algoritmo consegue reduzir o número de kilter NK a zero.

De fato, com esta hipótese de integralidade dos dados, cada vez que os fluxos (x) são alterados, pela fase primal, tem-se NK_j reduzido de ao menos uma unidade. Por sua vez, quando se altera os valores das variáveis duais (y) na fase dual, tem-se que ou a árvore a ser construída na próxima fase primal possuirá ao menos um nó (e arco) adicional, ou o número de kilter da solução é diminuído de ao menos uma unidade. De maneira que, não há possibilidade de se aplicar indefinidamente as fases primal e dual para um mesmo arco a_s OK, pois eventualmente se encontrará uma árvore geradora na fase primal, que juntamente com a_s formarão um ciclo, interrompendo a repetição destas duas etapas do

algoritmo. Assim, pode-se concluir que num número finito de passos NK pode ser diminuído até o valor zero.

4.2.3. Considerações acerca de um Método Dual para PRLP

Um algoritmo dual trabalha com soluções básicas dual viáveis (x, y, z), ou seja, as restrições $y\,M \leq z$, $z \geq c_1$, $Mx = b$ e as Condições de Folgas Complementares devem ser satisfeitas (relaxando a restrição $x \geq 0$). Neste caso, após a escolha do arco a_s a sair da base, o método realiza a cada pivoteamento dois testes da razão. O primeiro com respeito aos custos relativos, para escolha de um arco a_r a entrar na base, formando um ciclo com os arcos básicos e que inclue a_s. O segundo teste da razão é relativo a mudança de fluxo neste ciclo, pois os fluxos não podem ultrapassar os limitantes dos intervalos correntes para manter a solução dual viável. Isto implica que um arco básico $a_j \neq a_s$ pode ser, de fato, o arco a sair da base neste pivoteamento. Se isto ocorrer, deve-se reaplicar o algoritmo, ainda para o arco a_s prosseguindo a iteração.

Percebe-se assim a necessidade de um esforço computacional considerável, se comparado com outros métodos, de maneira que, não parece ser vantajoso esta linha de desenvolvimento de algoritmos para a PRLP.

4.2.4. Considerações acerca do Método de Khachiyan e do Método de Karmarkar para PRLP

Duas outras linhas de trabalho na abordagem da PRLP podem ser desenvolvidas, pelo enfoque de redes lineares por partes através dos métodos recentemente propostos, respectivamente, por Khachiyan /53/(1979), /54/(1980) e Karmarkar /55/(1984).

Sobre o primeiro método, também denominado Método do Elipsóide ("Ellipsoid Method"), tem-se a comentar o grande impacto causado quando da sua divulgação, dado que este método, que é uma adaptação de um método para otimização convexa anteriormente desenvolvido por Shor e Zhurbenko /56/(1971), possibilita a obtenção de um algoritmo polinomial (pelo critério da análise do pior caso) para a Programação Linear (PL), fato que não ocorre com o método simplex.

Contudo, um algoritmo que é superior a outros pelo critério do pior caso não necessariamente é melhor que aqueles na abordagem de problemas práticos (mundo real). De fato, isto ocorre com o método de Khachiyan em relação ao simplex, pois os resultados computacionais disponíveis até o momento indicam que este método não é uma alternativa prática para o simplex (ver Murty /57/(1983).

Assim sendo, apesar da aparente motivação para o uso deste método do elipsóide em PRLP em detrimento do simplex, não é aconselhável optar por esta alternativa.

Com relação ao mētodo proposto por Karmarkar, as perspectivas parecem ser melhores. Este procedimento, conhecido por Mētodo da Transformação Projetiva ("Projective Transformation Algorithm"), também possue as caracterīsticas polinomiais do mētodo anterior, na resolução de problemas em Programação Linear. Alguns pesquisadores, como Adler et alli /56/(1987), tem obtido resultados interessantes de implementações de variantes deste algoritmo, usando transformações afins, indicando que este tipo de procedimento pode se tornar um elemento importante na avaliação de problemas em PL e no futuro, possivelmente, para a PRLP.

No momento em que esta dissertação foi iniciada, poucas publicações estavam disponīveis acerca da abordagem, por este algoritmo, de problemas em redes lineares. Particularmente, teve-se acesso ao trabalho de Aronson et alli /57/(1985) em que relatava-se experimentos computacionais do algoritmo de Karmarkar aplicado a Problemas da Designação. Nesta referência foram comparadas resultados de implementações de um algoritmo especializado para redes, de um algoritmo de uso geral em PL e do mētodo de projeção, para problemas da designação (densos), gerados aleatoriamente. Encontrou-se que o mētodo simplex em redes foi 18 (dezoito) vezes mais rāpido que o mētodo simplex geral, que por sua vez foi 14 (quatorze) vezes mais rāpido que a implementação do mētodo Karmarkar. Este resultado justifica em parte a escolha efetuada pelo mētodo simplex, no estudo das redes lineares por partes levado a efeito nesta dissertação.

REFERÊNCIAS BIBLIOGRÁFICAS

/1/ CHARNES, A. e LEMKE, C.E. *Minimization of Non-Linear Separable Convex Functionals*. Naval Research Logistics Quartely 1, 12 páginas, 1954.

/2/ DANTZIG, G.B. *Recent Advances in Linear Programming*. Management Science 2, 14 páginas, 1956.

/3/ DANTZIG, G.B., JOHNSON, S. e WHITE, W. *A Linear Programming Approach to the Chemical Equilibrium Problem*. Management Science 5, 6 páginas, 1958.

/4/ HO, J.K. *Relations Among Linear Formulation of Separable Convex Piecewise Linear Programs*. Mathematical Programming Studies 24, 20 páginas, 1985.

/5/ SNYDER, R.D. *Programming with Piecewise-Linear Objective Functions*. Working paper 9/81, Department of Econometrics and Operations Research, Monash University, Clayton, Victoria, Austrália, 1981.

/6/ SNYDER, R.D. *Linear Programming with Special Ordered Sets*. Journal of the Operational Research Society 35, 6 páginas, 1984.

/7/ PREMOLI, A. *Piecewise-Linear Programming: The Compact (CPLP) Algorithm.* Mathematical Programming 36, 18 páginas, 1986.

/8/ ROCKAFELLAR, R.T. *Network Flows and Monotropic Optimizations.* John Wiley & Sons, New York, 1984.

/9/ CONN, A.R. *Linear Programming via a Nondifferentiable Penalty Function.* SIAM Journal on Numerical Analysis 13, 10 páginas, 1976.

/10/ BARTELS, R.H. *A Penalty Linear Programming Method using Reduced-Gradiente Basis-Exchange Techniques.* Linear Algebra and its Applications 29, 16 páginas, 1980.

/11/ GOLSTEIN, E.G. *A Certain Class of Nonlinear Extremum Problems.* Soviet Mathematics 1, 4 páginas, 1960.

/12/ ORDEN, A. e NALBANDIAN, V. *A Bidirectional Simplex Algorithm.* Journal of the Association of Computing Machinery 15, 15 páginas, 1968.

/13/ ROSVANY, G.I.N. *Concave Programming in Structural Optimization.* International Journal of Mechanical Sciences 12, nº 2, 12 páginas, fevereiro, 1970.

/14/ ROSVANY, G.I.N. *Concave Programming and Piece-wise Linear Programming.* International Journal for Numerical Methods in Engineering 3, 14 páginas, 1971.

/15/ CHARRET, D.E. *Plastic Analysis and Optimum Design of Slab e Frames*. Special Report S2/1970, Civil Engineering, Monash University, Clayton, Vitória, Austrália, 264 páginas, 1970.

/16/ GOLSTEIN, E.G. e YOUDINE, D. *Problèmes Particuliers de la Programmation Lineaire*. Editions Mir, Moscou, 560 páginas, 1973.

/17/ SOUZA, C.R. *Aplicação de Programação Linear por Partes a Sistemas de Potência*. Tese de Mestrado. Faculdade de Engenharia Elétrica, UNICAMP, Campinas, São Paulo, Brasil, 71 páginas, 1977.

/18/ GARCIA, A.S. *Método Dual-Simplex para Problemas com Critério Linear por Partes*. Tese de Mestrado. Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, Campinas, São Paulo, Brasil, 68 páginas, 1978.

/19/ FERREIRA, E.P. *Programação Linear por Partes: Método Primal Dual*. Tese de Mestrado. Faculdade de Engenharia Elétrica, UNICAMP, Campinas, São Paulo, Brasil, 86 páginas, 1979.

/20/ RIBEIRO, R.V. *Estudos em Programação Linear*. Tese de Doutorado. Faculdade de Engenharia Elétrica, UNICAMP, Campinas, São Paulo, Brasil, 109 páginas, 1980.

/21/ FERNANDES, J.F.R. *Programação Linear por Partes: Resolução por Decomposição de um Problema de Grande Porte*. Tese de

Doutorado, Faculdade de Engenharia Elétrica, UNICAMP, Campinas, São Paulo, Brasil, 164 páginas, 1979.

/22/ FOURER, R. *A Simplex Algorithm for Piecewise-Linear Programming I: Derivation and Proof.* Mathematical Programming 33, 30 páginas, 1985.

/23/ FOURER, R. *A Simplex Algorithm for Piecewise-Linear Programming II: Finitess, Feasibility and Degeneracy.* Technical Report 85-03, Department of Industrial Engineering and Mabagement Sciences, Northwestern University, Evanston, Illinois, EUA, 46 páginas, 1985.

/24/ FOURER, R. *A Simplex Algorithm for Piecewise-Linear Programming III: Computational Analysis and Applications.* Technical Report 86-03, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, EUA, 65 páginas, 1986.

/25/ GLOVER, F., KARNEY, D. e KLINGMAN, D. *Implementation and Computation Comparasions of Primal, Dual and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems.* Networks 4, 22 páginas, 1974.

/26/ BARR, R., GLOVER, F. e KLINGMAN, D. *The Alternating Basis Algorithm for Assignment Problems.* Mathematical Programming 13, 13 páginas, 1977.

/27/ BARR, R., ELAM, J., GLOVER, F. e KLINGMAN, D. *A Network Augmenting Path Algorithm for Transshipment Problems*. Working Paper 77-10-04, Wharton School, University of Penn, Philadelphia, Pa., EUA, 1977.

/28/ BRADLEY, G.H., BROWN, G.G. e GRAVES, G.N. *Design and Implementation of Large Scale Primal Transshipment Algorithms*. Management Science 24 nº 1, setembro, 1977.

/29/ CUNNINGHAM, W.H. *A Network Simplex Method*. Mathematical Programming 11, 12 páginas, 1976.

/30/ CUNNINGHAM, W.H. *Theoretical Properties of the Network Simplex Method*. Mathematics of Operations Research 4 nº 2, maio, 1979.

/31/ CUNNINGHAM, W.H. e KLINCEWICZ, J.G. *On Cycling in the Network Simplex Method*. Mathematical Programming 26, 8 páginas, 1983.

/32/ BLAND, R.G. *New Finite Pivoting Rules for the Simplex Method*. Mathematics of Operations Research 2 nº 2, maio, 1977.

/33/ KENNINGTON, J.L. e HELGASON, R.V. *Algorithms for Network Programming*. John Wiley & Sons, New York, 291 páginas, 1980.

/34/ ALI, A.I., HELGASON, R.V., KENNINGTON, J.L. e LALL, H.S. *Primal Simplex Network Codes: State-of-the-Art Implementation Technology*. Networks 8, 25 páginas, 1978.

/35/ EDMONDS, J. e KARP, R.M. *Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems.* Em: Combinatorial Structures and Their Applications páginas 93-96, Gordon e Breach, New York, 1970.

/36/ EDMONDS, J. e KARP, R.M. *Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems.* Journal of Association for Computing Machinery 19, 17 páginas, 1972.

/37/ GABOW, H.N. *Scaling Algorithms for Network Problems.* Journal of Computer and System Sciences 31, 21 páginas, 1985.

/38/ IKURA, Y. e NEMHAUSER, G.L. *Computational Experience with a Polynomial-Time Dual Simplex Algorithm for the Transportation Problem.* Discrete Applied Mathematics 13, 20 páginas, 1986.

/39/ KLEE, V. e MINTY, G.J. *How Good is the Simplex Algorithm?* Em: Inequalities III, páginas 159-175, O.Shicha (ed.), Academic Press, New York, 1972.

/40/ GOLDFARB, D. e SIT, W.Y. *Worst Case Behavior of the Steepest Edge Simplex Method.* Discrete Applied Mathematics 1, 10 páginas, 1979.

/41/ GASSNER, B.J. *Cycling in the Transportation Problem.* Naval Research Logistics Quartely 11, 16 páginas, 1964.

/42/ ZADEH, N. *A Bad Network Problem for the Simplex Method and Other Minimum Cost Flow Algorithms*. Mathematical Programming 5, 17 páginas, 1973.

/43/ HUNG, M.S. *A Polynomial Simplex Method for the Assignment Problem*. Operations Research 31 nº 3, maio-junho, 1983.

/44/ CROWDER, H. e HATTINGH, J.M. *Partially Normalized Pivot Section in Linear Programming*. Mathematical Programming Study 4, 14 páginas, 1975.

/45/ GLOVER, F., KLINGMAN, D. e STUTZ, J. *Augmented Threaded Index Method for Network Optimization*. Infor 12 nº 3, 6 páginas, outubro, 1974.

/46/ CROWDER, H.P., DEMBO, R.S. e MULVEY, J.M. *Reporting Computational Experiments in Methematical Programming*. Mathematical Programming 15, 14 páginas, 1978.

/47/ NAKAGAWA, J.M. *Planejamento de Sistemas Telefônicos : Alocação de Centros de Fios*. Tese de Mestrado. Faculdade de Engenharia Elétrica, UNICAMP, Campinas, São Paulo, Brasil, 1984.

/48/ FRANÇA, P.M., FERNANDES, J.F.R. e TAVARES, H.M.F. *Expansão de Redes Telefônicas*. SBA: Controle & Automação, vol.1, nº 3, julho, 1987.

/49/ ALI, A.I., ALLEN, E.P., BARR, R.S. e KENNINGTON, J.L. *Reoptimization Procedures for Bounded Variable Primal Simplex Network Algorithms*. European Journal of Operational Research 23, 8 páginas, 1986.

/50/ FULKERSON, D.R. *An Out-of-Kilter Method for Minimal-Cost Flow Problems*. Journal of the Society of Industrial and Applied Mathematics 9, volume 1, 10 páginas, 1961.

/51/ BARR, R.S. GLOVER, F. e KLINGMAN, D. *A Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes*. Mathematical Programming 7, volume 1, 28 páginas, 1974.

/52/ GLOVER, F. e KLINGMAN, D. *Some Recent Practical Misconceptions about the State-of-the-Art of Network Algorithms*. Operations Research 2, 10 páginas, 1978.

/53/ HAČIJAN, L.G. *A Polynomial Algorithm in Linear Programming*. Soviet Math-Doklady 20, volume 1, 5 páginas, 1979.

/54/ KHACHIYAN, L.G. *Polynomial Algorithms in Linear Programming*. Zhurnal Vichislitel'noi Matēmatiki i Matematicheskoi Fiziki 20, volume 1, 18 páginas, 1980.

/55/ KARMARKAR, N. *A New Polynomial Time Algorithm for Linear Programming*. Combinatorica 4, 23 páginas, 1984.

/56/ SHOR, N.Z. e ZHURBENKO, N.G. *A Minimization Method Utilizing the Operation of Space Expansion in the Direction of the Difference of Two Successive Gradientes*. Cybernetics 7, volume 3, 9 pāginas, 1971.

/57/ MURTY, K.G. *Linear Programming*. John Wiley & Sons, New York, EUA, 482 pāginas, 1983.

/58/ ADLER, I., RESENDE, M.G.C. e VEIGA, G. *An Implementation of Karmarkar's Algorithm for Linear Programming*. Working Paper 86-8 Dept. of Industrial Engineering and Operations Research, University of California, Berkeley, EUA, 1986.

/59/ ARONSON, J., BARR, R., HELGASON, R., KENNINGTON, J., LOH, A. e ZAKI, H. *The Projective Transformation Algorithms by Karmarkar : A Computational Experiment with Assignment Problems*. Technical Report 85-OR-3, Department of Operations Research, Southern Methodist University, Dallas, Texas, 10 pāginas, 1985.