RELAXAÇÃO LAGRANGEANA COM GERAÇÃO DE DESIGUALDADES VÁLIDAS APLICADA AO PROBLEMA DE ROTEAMENTO DE VEÍCULOS

Carlos Alberto de Jesus Martinhon

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Nelson Maculan D. Sc
(Presidente/Orientador)

Prof. Abílio Pereira de Lucena, Ph.D
(Orientador)

Prof. Adilson Elias Xavier, D. Sc.

Prof. Cid Carvalho de Souza, Ph.D

Prof. Luiz Satoru Ochi, D.Sc

Prof. Marcus V. Poggi de Aragão, Ph.D

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO DE 1998

MARTINHON, CARLOS ALBERTO DE JESUS

Relaxação Lagrangeana com Geração de Desigualdades Válidas Aplicada ao Problema de Roteamento de Veículos

IX, 188p. 29,7cm

(COPPE/UFRJ, D.Sc PROGRAMA DE ENGENHARIA DE SISTEMAS - OTIMIZAÇÃO 1998)

TESE - Universidade Federal do Rio de Janeiro, COPPE.

1. Roteamento de Veículos

2. Relaxação Lagrangeana

3. Teoria Poliédrica

I COPPE/UFRJ. II. Título (série)

Dedico este trabalho a meus pais
e a minha esposa.

AGRADECIMENTOS

Gostaria de expressar minha sincera gratidão a algumas pessoas que contribuíram, direta ou indiretamente, para realização deste trabalho. Gostaria de agradecer, em especial, a meus orientadores Nelson Maculan e Abílio Lucena, pela amizade, apoio e incentivo dado nas diversas etapas deste trabalho.

Quero agradecer ao amigo Otton pelas lições de linguagem C, tão valiosas no início de minha implementação. Gostaria de agradecer aos amigos Marco, Nivaldo, Orizon, Dante, Rosa, Satoru, Márcia... pelos diversos momentos de descontração e "bate-papo" . Gostaria de agradecer também a meus sogros Cláudio e Martha por todo apoio dado.

Sou realmente muito grato a minha família, meus pais, minhas irmãs, sobrinhos e, em especial minha esposa Priscila pelo seu grande incetivo.

Gostaria de estender meus agradecimentos à CAPES e a todo o Programa de Engenharia de Sistemas da COPPE/UFRJ.

Atenciosamente,

Carlos Alberto de Jesus Martinhon.

Resumo da tese apresentada à COPPE/UFRJ, como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc).

RELAXAÇÃO LAGRANGEANA COM GERAÇÃO DE DESIGUALDADES VÁLIDAS APLICADA AO PROBLEMA DE ROTEAMENTO DE VEÍCULOS

Carlos Alberto de Jesus Martinhon

Setembro, 1998.

Orientadores: Nelson Maculan

Abílio Pereira de Lucena

Departamento: Programa de Engenharia de Sistemas e Computação

RESUMO

Trabalhamos na construção de um método exato que busca a combinação de relaxação lagrangeana com resultados de combinatória poliédrica, direcionados, especificamente, para o problema clássico de roteamento de veículos. Neste problema, dispomos de uma frota de K veículos idênticos e desejamos atender um conjunto de n clientes, cada um com uma demanda específica. Todos os veículos devem partir e retornar a uma mesma origem (depósito) e cada cliente deve ser visitado uma única vez. O objetivo será minimizar o "custo total" de transporte no atendimento aos clientes sem violar a capacidade de cada veículo.

Dado um grafo G com n+1 vértices, uma K-árvore, é definida como sendo um sub-grafo conexo de G com n+1 vértices e n+K arestas. Pode-se mostrar facilmente (Fisher[94.b]), que o problema de roteamento de veículos pode ser modelado como o problema da K-árvore mínima com 2K arestas incidentes ao depósito, juntamente com restrições adicionais de capacidade (imposta aos veículos) e restrições que indiquem que cada cliente deva ser visitado uma única vez. Estas restrições adicionais são dualizadas e

um problema lagrangeano é obtido para geração de limites inferiores para o problema de roteamento.

Fazemos uma análise da abordagem utilizada por Fisher[94.b] que emprega K-árvores mínimas com a geração de desigualdades válidas (particularmente sub-rotas violadas). Este trabalho de Fisher[94.b], nos serviu de base para a adoção de novas estratégias na identificação de restrições violadas. Introduzimos as desigualdades *comb* e *multistars* (formulações de Cornuejouls e Harche[93] e Araque et al.[90] respectivamente) e desenvolvemos um procedimento para fixação de variáveis.

Uma nova heurística lagrangeana é também apresentada na determinação dos limites superiores para o valor de uma solução ótima do problema de roteamento.

São apresentados resultados computacionais para instâncias de até 199 clientes (na geração dos limites inferiores), e de até 100 clientes, na busca em árvore implementada. Fazemos comparações de nossa abordagem com as abordagens apresentadas por Fisher[94.b] (K-árvores mínimas) e Miller[95] (*b-matching*) na geração dos limites inferiores.

Abstract of thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc)

LAGRANGEAN RELAXATION WITH VALID INEQUALITIES GENERATION FOR VEHICLE ROUTING

Carlos Alberto de Jesus Martinhon

September, 1998

Thesis Superviors: Nelson Maculan
Abílio Pereira de Lucena
Department: System Engineering Program

ABSTRACT

We consider an exact approach combining Lagrangean relaxation and polyhedral combinatorics dedicated to the Vehicle Routing Problem (VRP). In the VRP, we want optimaly scheduling a fleet of K vehicles to make deliveries to n customers subject to vehicle capacity constraints.

Given a graph with n+1 nodes, a K-tree is defined to be a set of n+K edges that span the graph. We show that the vehicle routing problem can be modeled as the problem of finding a minimum cost K-tree with two K edges incident on the depot and subject to some side constraints that impose vehicle capacity and the requirement that each customer be visited exactly once. The side constraints are dualized to obtain a Lagrangean problem that provides lower bounds in a branch-and-bound algorithm.

We investigate, with the same framework proposed by Fisher[94.b], new identifications procedures to obtain violated inequalities (specially violated subtours) from the Lagrangean problem. We also deals with combs and multistars (like proposed by Cornuejouls and Harche[93] and Araque at al.[90]) and we present a new scheme for fixing variables based on the K-tree relaxation.

A new lagrangean heuristic is also presented to produce upper bounds for the optimal solution associated with the VRP.

We present computational results for problems involving 199 customers (in the lower bounds generation procedure) and problems involving 100 customers in the branch-and-bound algorithm. We compare our approach with the approaches proposed by Fisher[94.b] (minimum K-trees relaxation) and Miller[95] (b-matching relaxation).

Introdução

No problema clássico de roteamento dispomos de uma frota de K veículos idênticos e desejamos atender um conjunto de n clientes, cada um com uma demanda específica. Todos os veículos devem partir e retornar a uma mesma origem (depósito) e cada cliente deve ser visitado uma única vez. O objetivo será minimizar o "custo total" de transporte no atendimento aos clientes sem violar a capacidade de cada veículo. Iremos nos referir também ao problema clássico de roteamento como o problema de roteamento de veículos com restrições de capacidade (PRVRC).

Trabalhamos na construção de um método exato que busca a combinação de relaxação lagrangeana com resultados de combinatória poliédrica, direcionados, especificamente, para o problema de roteamento de veículos (PRVRC). Fazemos uma análise da abordagem utilizada por Fisher[94.b] que emprega K-árvores mínimas com a geração de desigualdades válidas (particularmente sub-rotas violadas). Este trabalho de Fisher[94.b], nos serviu de base para a adoção de novas estratégias na identificação de restrições violadas. Introduzimos as desigualdades *comb* e *multistars* (formulações de Cornuejols e Harche[93] e Araque et al.[90] respectivamente) e desenvolvemos um procedimento para fixação de variáveis.

Uma nova heurística lagrangeana é também apresentada na determinação dos limites superiores para o valor de uma solução ótima do problema de roteamento.

No capítulo I, fazemos uma breve explanação sobre algumas das variações do problema de roteamento decorrentes do acréscimo de novas restrições. Apresentamos algumas estratégias envolvendo métodos heurísticos e exatos, utilizadas na solução do problema de roteamento. Uma extensa bibliografia utilizando estas duas abordagens é citada.

Os principais conceitos e definições necessárias à aplicação da combinatória poliédrica, são apresentados no capítulo II. Como veremos, estes conceitos serão fundamentais para uma melhor compreensão dos capítulos posteriores.

No capítulo III, fazemos uma exposição do método que combina relaxação lagrangeana com a geração de desigualdades válidas. Este método difere da abordagem tradicional que determina restrições violadas a partir da relaxação linear de uma formulação para o problema original *(branch-and-cut* - vide Junger et al.[95]).

O capítulo IV, trata da relaxação lagrangeana com a geração de sub-rotas violadas utillizada por Fisher[94.b]. Nesta relaxação, Fisher trabalha basicamente na determinação de uma K-árvore mínima com 2K arestas incidentes ao depósito (Fisher[94.a]). Discutimos em seguida (seção IV.3), os principais aspectos, presentes em nossa implementação, na obtenção da K-árvore mínima. A geração das restrições de eliminação de sub-rotas e as heurísticas lagrangeanas utilizadas por Fisher são também apresentadas.

No capítulo V, apresentamos algumas classes de desigualdades válidas existentes na literatura para o problema de roteamento e que foram utilizadas em nosso trabalho. Tratamos especificamente das restrições de eliminação de sub-rotas, *combs* e *multistars* respectivamente.

O capítulo VI trata da identificação de restrições violadas a partir da solução do problema lagrangeano (K-árvore mínima com 2K arestas incidentes ao depósito). Discutimos novos procedimentos na determinação de sub-rotas, *combs* e *multistars* violadas. Em especial, apresentamos um algoritmo exato na identificação das restrições de eliminação de sub-rotas. Como veremos posteriormente, resolvemos de maneira mais eficiente o problema de identificação de sub-rotas violadas. Isto irá possibilitar a geração de limites inferiores melhores que os apresentados em Fisher[94b]. Os principais aspectos relativos à estrutura de dados utilizada em nossa implementação são também considerados.

No capítulo VII, apresentamos critérios que permitam fixar variáveis em *um* ou *zero*, conforme as arestas correspondentes apareçam ou não em uma solução ótima do problema original. Esta fixação de variáveis deverá ser feita sempre a partir da K-árvore mínima obtida na solução do problema lagrangeano. Como veremos, estes critérios irão propiciar condições para uma melhora na qualidade dos limites inferiores e superiores gerados no transcorrer do método subgradiente.

No capítulo VIII, fazemos uma síntese das principais etapas presentes na solução do problema dual lagrangeano (através do método subgradiente) e apresentamos uma nova heurística lagrangeana, baseada no algoritmo Clarke e Wright[64], para determinação dos limites superiores. Em seguida, fazemos uma exposição da estratégia de ramificação (*branching*) e a busca em árvore (*branch-and-bound*) implementada em nosso trabalho.

Finalmente, no capítulo IX, discutimos alguns dos resultados computacionais obtidos em nossa implementação. Fazemos uma comparação dos limites inferiores obtidos em nossa abordagem com as abordagens apresentadas em Fisher[94b] e Miller[95] respectivamente. Apresentamos uma tabela relacionando o número de variáveis fixadas em *zero* e *um* após a utilização de nosso procedimento que fixa variáveis. Mostramos ainda alguns resultados computacionais obtidos pela nova heurística lagrangeana (C&W lagrangeano) bem como o tempo de processamento e o número de nós gerados na árvore de busca.

Capítulo I

Roteamento de Veículos

I.1 - Introdução:

Roteamento de veículos é uma designação genérica utilizada para uma grande quantidade de problemas de otimização combinatória, envolvendo basicamente, a distribuição e coleta de mercadorias e serviços.

Trabalharemos especificamente com o problema de roteamento de veículos com restrições de capacidade (PRVRC). Deveremos atender um conjunto (fixo) de clientes, cada um com uma demanda específica, utilizando para isso, uma frota de veículos com capacidades idênticas. Todos os veículos devem partir e retornar a uma mesma origem (depósito) e cada cliente deve ser visitado uma única vez. Nosso objetivo será minimizar a distância total percorrida pelos veículos atendendo a demanda dos clientes sem violar a capacidade de cada veículo.

Muitas aplicações são extensões diretas deste problema clássico de roteamento (c/ restrições de capacidade). Entre as restrições adicionais associadas ao problema podemos citar:

1) a utilização de vários depósitos;
2) a utilização de percursos que possam se iniciar em um depósito e terminar em um depósito distinto;
3) atendimento aos clientes em janelas de tempo (*time-windows*) previamente estabelecidas;
4) a utilização de um tempo total T de percurso para todos os veículos;
5) a utilização de uma distância máxima D para o percurso de cada veículo;
6) veículos que percorrem duas ou mais rotas distintas em um mesmo dia;
7) restrições de precedência na visita aos clientes;
8) a utilização de outros objetivos, como por exemplo, a minimização do número de veículos utilizados;
9) veículos com capacidades distintas, etc.

O problema clássico de roteamento de veículos pode ser definido em um grafo simétrico $G = (N_0, E)$ sendo que N_0={0,1,2,..,n} representa um conjunto de n+1 vértices (o vértice 0 representa o depósito e os demais vértices os clientes). O conjunto de arestas $E = \{(i,j) / i,j \in N_0 \ e \ i \neq j\}$ define as conexões entre cada par de vértices. A cada uma das arestas (i,j) de E, representamos por c_{ij}, o custo de transporte associado ao se viajar do vértice **i** para o vértice **j**.

Podemos encontrar uma grande quantidade de trabalhos na literatura abordando as inúmeras variações do problema clássico de roteamento de veículos (Laporte e Osman[94]). Entre os "estados da arte" sobre o assunto podemos citar os trabalhos de Bodin et al. [83], Christofides[85], Laporte e Nobert[87], Laporte[92.b] entre outros.

O problema de roteamento de veículos c/ restrições de capacidade (PRVRC) pode ser visto como uma extensão do problema do m-caixeiro viajante. No m-caixeiro viajante desejamos atender a um conjunto de clientes utilizando uma frota de **m** veículos com capacidade ilimitada, iniciando e finalizando cada um dos percursos em um vértice pré-determinado (depósito). Analogamente ao problema de roteamento, cada cliente deve ser percorrido uma única vez e o objetivo será minimizar o "custo total" de transporte. Os

problemas de roteamento descritos acima podem ser classificados como NP-árduos por serem uma extensão do problema do m-caixeiro viajante, reconhecidamente NP-árduo (vide Garey & Johnson[79] e Lenstra & Rinnooy [81]).

Fazemos a seguir, uma breve exposição de algumas das principais técnicas utilizadas na solução do problema de roteamento de veículos com restrições de capacidade (PRVRC):

I.2 - Métodos Heurísticos:

Em função da elevada complexidade dos problemas de roteamento em geral (problemas NP-Árduos), muitos dos algoritmos propostos para solução destes problemas, não buscam diretamente um mínimo global. Nestes casos, uma solução de baixo custo computacional (em tempo polinomial) e idealmente "próxima" da solução ótima procurada é obtida. Os métodos heurísticos podem ser divididos basicamente em duas fases: determinação de uma "boa" solução viável e busca local para melhoria desta solução.

A determinação de soluções viáveis para o problema de roteamento é frequentemente derivada de procedimentos para o caixeiro viajante (vide Laporte [92.a], [92.b]). O método do vizinho mais próximo, algoritmo de inserção e melhoramento de rotas (*tour improvement*) podem ser aplicados ao problema de roteamento quase sem modificações. A única preocupação nestes casos, é satisfazer as restrições adicionais presentes no problema de roteamento.

Uma das heurísticas mais conhecidas para o problema de roteamento (c/ restrições de capacidade) é o algoritmo de Clarke & Wright [64]. Neste caso, o número de veículos utilizados não é fixado a priori. O método se inicia com a construção de rotas simples (depósito-cliente-depósito) e, a cada passo, busca-se a formação de novas rotas viáveis a partir das rotas já obtidas. A justificativa para o procedimento é que o custo associado a duas rotas distintas é sempre maior que o custo associado a uma única rota que atenda a esses clientes. Desta forma, tem-se uma diminuição no "custo total" de transporte sempre que duas novas rotas puderem ser combinadas para formação de uma nova rota, sem violar obviamente, a capacidade do veículo que atende a esses clientes.

Esta situação é representada esquematicamente na figura I.1 a seguir:

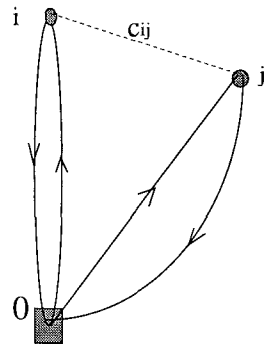


Figura I.1: Construção da rota 0-i-j-0

Note que $s_{ij} = 2c_{0i} + 2c_{0j} - (c_{0i} + c_{ij} - c_{0j}) = c_{0i} + c_{0j} - c_{ij}$ representa a economia de custo obtida após a substituição das rotas 0-i-0-j-0 pela rota 0-i-j-0.

Descrevemos a seguir, as principais etapas presentes no algoritmo de Clarke& Wright[64]. Seja n, o número total de clientes e c_{ij} o custo de transporte associado ao percurso direto do nó **i** para o nó **j** (ou vice-versa).

ALGORITMO: (Clarke & Wright [64])

Passo 0) Constrói n rotas simples (depósito-cliente-depósito);

Passo 1) calcula as economias $s_{ij} = c_{i0} + c_{0j} - c_{ij}$, onde i≠j (o vértice 0 representa o depósito e os demais vértices representam os clientes);

Passo 2) ordena (em ordem decrescente) as economias s_{ij} ;

Passo 3) seleciona a maior economia s_{ij} ;

Passo4) se não for violada a restrição de capacidade, **i** e **j** não pertencerem a mesma rota e **i** (ou **j**) não forem interiores a nenhuma das rotas já obtidas (isto é, se **i** (ou j) não estiverem "ligados" ao depósito) então conectamos os vértices **i** e **j** e atualizamos a capacidade da nova rota;

Passo 5) atualiza a ordenação das economias;

Passo 6) retorna ao passo 3.

Figura I.2: Algoritmo Clarke&Wright

As economias $s_{ij}>0$ indicam a redução obtida no custo de transporte ao percorrermos a aresta (i,j). O algoritmo acima termina quando a lista de economias s_{ij} se tornar vazia. Este procedimento pode ser executado em $O(n^2 \log n)$ passos mas sua complexidade pode ser reduzida utilizando-se estruturas de dados apropriadas (vide p. ex. Golden et al.[77], Nelson et al. [85]). Paessens [88], propõe ainda algumas variantes deste método.

Em Laporte [92.b] pode-se encontrar outras heurísticas bastantes utilizadas para o problema clássico de roteamento (PRVRC).

Uma das técnicas mais eficientes na busca local desenvolvida para o problema do caixeiro viajante é o algoritmo de Lin & Kernighan[73] (k-optimal). Esta técnica pode ser utilizada também para o problema de roteamento após determinarmos uma solução viável para o problema. Neste caso, cada uma das rotas definidas pela solução heurística no roteamento, determina uma rota viável para o problema do caixeiro viajante. Como discutido em Savage et al.[76], os métodos de busca local aplicados a vizinhanças de tamanho polinomial nunca garantem otimalidade, ou seja, sempre podemos construir instâncias aonde a busca local se mostra bastante ineficiente (Papadimitriou & Steiglitz[78]). Logo, não temos garantida o obtenção do ótimo global do problema utilizando-se apenas métodos heurísticos.

Especialmente na última década, outras técnicas conhecidas como metaheurísticas tiveram uma grande aceitação na resolução de problemas combinatórios em geral (vide Reeves[93]). Entre elas podemos citar o *simulated annealing*, busca tabu, método GRASP, algoritmos genéticos entre outras. Estes métodos geram soluções viáveis e fazem busca local como nos métodos heurísticos clássicos, mas permitem também, um aumento no valor da função objetivo na intenção de escapar dos mínimos locais (considerando obviamente que o problema original seja de minimização).

Em Gendreau, Laporte e Potvin[94], podemos destacar a aplicação de várias metaheurísticas distintas na resolução do problema de roteamento de veículos.

I.3 - Métodos Exatos:

Os métodos exatos trabalham, em sua maioria, com esquemas de enumeração conhecidos como "busca em árvore" (*branch-and-bound*) e tem por objetivo à determinação de um ótimo global para o problema. Nestes métodos, visamos basicamente a determinação de limites inferiores (e superiores) para o custo de um ótimo global. Estes limites podem ser obtidos por procedimentos distintos, podendo-se destacar por exemplo, a relaxação lagrangeana (Geoffrion[74]) ou relaxação linear (Dantzig[51]).

Métodos de busca em árvore geram esquemas de enumeração a partir do domínio do problema original. Este domínio é dividido (idealmente particionado) sempre que o limite inferior obtido não atinja a uma solução ótima do respectivo domínio. Pesquisamos somente aquele nós (sub-problemas) da árvore de busca onde uma solução ótima pode ser encontrada. Vários nós poderão ser descartados implicitamente utilizando-se os melhores resultados parciais obtidos ao longo do processo de busca (limites inferiores e superiores). Quanto melhor forem os limites inferiores associados a cada nó, menor será o tamanho (número de nós) de nossa árvore de busca (Murty[76]).

Para ilustrar a geração de limites inferiores nos métodos de busca em árvore (*branch-and-bound*), vejamos por exemplo, a estratégia utilizada por Miller[95] na resolução do problema de roteamento de veículos (PRVRC). Miller trabalha basicamente na determinação do b-emparelhamento (*b-matching*) de custo mínimo (Miller & Pekny[94]).

Seja G(V,E) um grafo completo com um conjunto V, de n+1 vértices, e um conjunto E de arestas. A determinação de um b-emparelhamento mínimo em G(V,E) é realizada buscando-se um subgrafo $M(V,\overline{E})$ tal que, cada vértice i de V seja adjacente em exatamente b_i arestas de E. Cabe ressaltar ainda que uma aresta **e** pertencente a E, pode ser computada d_e vezes. Como veremos a seguir, o problema do b-emparelhamento escolhendo-se convenientemente os valores de b_i para cada $i \in V$, e d_e para cada $e \in E$, pode ser visto como uma relaxação para o problema de roteamento de veículos c/ restrições de capacidade.

Os custos e a capacidade de cada aresta **e** pertencente a E são representados por c_e e d_e respectivamente. Para cada subconjunto $S \subset V$, representamos por $\sigma(S)$ o conjunto das

arestas de E com ambas as extremidades em S, e $\delta(S)$ o conjunto das arestas pertencentes a E com apenas uma das extremidades em S. Seja x_e, a variável que representa o número de vezes que computamos a aresta e∈E. Temos então a seguinte relaxação para o PRVRC:

$$min \sum_{e \in E} c_e x_e \qquad (1)$$

s.a $$\sum_{e \in \delta(i)} x_e = b_i, \quad \forall i \in V \qquad (2)$$

$$0 \leq x_e \leq d_e \qquad (3)$$

$$x_e, d_e \in Z^+, \quad \forall\, e \in E \qquad (4)$$

onde:

$b_0 = 2K$ é grau do depósito;

$b_i = 2$ p/ i∈{1,..,n};

$d_e = 2$, ∀ e∈ $\delta(\{0\})$;

$d_e = 1$, ∀ e∉ $\delta(\{0\})$.

Note que o conjunto de restrições (2), indica a existência de K rotas (já que temos 2K arestas incidentes ao depósito). Além disso, cada cliente deve ser visitado uma única vez. O conjunto de restrições (3) e (4) indicam que apenas as arestas incidentes ao depósito podem ser duplicadas, possibilitando a existência de rotas simples do tipo (depósito-cliente-depósito).

Note, na formulação (1)-(4) acima, que temos uma relaxação (limite inferior) para o problema de roteamento já que as restrições de capacidade de cada veículo e conexidade do grafo são desprezadas.

Representamos na figura I.3 uma solução viável para o b-emparelhamento com 3 veículos e 11 vértices:

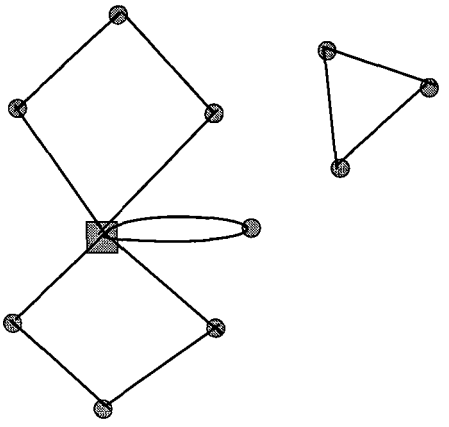


Figura I.3: Solução viável p/ o problema do b-emparelhamento

Seja q_i, a demanda associada a cada cliente e C, a capacidade de cada veículo. Uma formulação para o PRVRC pode ser apresentada se introduzimos o seguinte conjunto de restrições (de eliminação de sub-rotas) às restrições que definem o b-emparelhamento:

$$\sum_{e\in\sigma(S)} x_e \leq |S| - \left\lceil \frac{a(S)}{C} \right\rceil \quad \text{onde } a(S) = \sum_{i\in S} q_i \text{ , } S\subset V\backslash 0 \text{ e } |S| \geq 3 \qquad (5)$$

Seja Ψ, o conjunto de todos os subconjuntos S que satisfazem (5). Se dualizamos o conjunto (exponencial) de restrições (5) e associamos multiplicadores $\lambda_S \geq 0$, a cada uma destas restrições, teremos o seguinte problema lagrangeano:

$$L(x,\lambda) = min\left\{ \sum_e \bar{c}_e x_e - \sum_{S\in\Psi} \lambda_S \left(|S| - \left\lceil \frac{a(S)}{C} \right\rceil \right) \right\}$$

$$s.a \qquad (2)-(4)$$

onde $\bar{c}_e = c_e + \sum_{\{S|e\in\sigma(S)\}} \lambda_S$.

Como temos um número exponencial de restrições de eliminação de sub-rotas são dualizadas apenas aquelas que venham a violar a solução de um problema lagrangeano (Miller[95]).

Desejamos ajustar os multiplicadores de Lagrange de maneira a obter o maior limite inferior possível. Para isto, podemos resolver o seguinte problema dual lagrangeano pelo método subgradiente (Geoffrion[74], Fisher[81]):

$$\begin{aligned} &max\ L(x,\lambda) \\ &s.a \quad \lambda \geq 0 \end{aligned} \qquad \text{(PD)}$$

No método subgradiente, basicamente, determinamos um conjunto de multiplicadores λ inicial e resolvemos o problema lagrangeano associado. Em seguida, esses multiplicadores são atualizados e o processo é repetido até que algum critério de parada tenha sido atendido. Maiores detalhes podem ser encontrados em Geoffrion[74] ou Fisher[81].

Assim, para cada conjunto de multiplicadores λ dado, temos o problema do b-emparelhamento, que pode ser resolvido em tempo polinomial (vide Miller e Pekny[94]). Em seu trabalho, Miller[95] implementa ainda uma pequena variação do método subgradiente apresentado em Geoffrion[74].

Entre os métodos de resolução para o problema de roteamento de veículos utilizando programação linear inteira podemos citar o trabalho de Balinski e Quandt[64]. Nesta abordagem, eles utilizam uma formulação que trabalha com a partição de conjuntos (*set partitioning*) e geração de colunas.

Seja J, o conjunto de todas as possíveis rotas **j** viáveis e a_{ij} um conjunto de coeficientes binários 0-1, representando 1 se e somente se o vértice i>0 aparece na rota **j** e 0 em caso contrário. Representaremos por c_j^*, o custo ótimo associado a rota **j** e por x_j, a variável binária igual a 1 se e somente se a rota **j** aparece na solução ótima. Se não fizermos restrições quanto ao número de veículos utilizados o problema de roteamento poderá ser formulado da seguinte maneira:

$$min \sum_{j \in J} c_j^* x_j \qquad (6)$$

$$\text{s.a.} \quad \sum_{j \in J} a_{ij} x_j = 1, \quad \forall\ i \in V \setminus \{0\} \qquad (7)$$

$$x_j \in \{0,1\}, \quad \forall\ j \in J \qquad (8)$$

Observe que o conjunto de restrições (7) indica que cada cliente deve pertencer a uma única rota. Podemos destacar 2 dificuldades principais associadas a esta formulação:

(**i**) o número extremamente grande de variáveis binárias x_j;

(**ii**) a grande dificuldade no cálculo dos custos c_j^*.

Note (item (ii)) que no problema de roteamento (PRVRC), cada rota **j** corresponde a um conjunto de vértices S_j satisfazendo a capacidade de cada veículo. O valor c_j^* é então obtido resolvendo-se o problema do caixeiro viajante em S_j. Se considerarmos que o número de variáveis envolvidas seja razoavelmente pequeno (não mais que alguns milhares de variáveis), qualquer algoritmo desenvolvido especificamente para o problema de partição pode ser utilizado (vide Balas & Padberg[79], Marstens[74]).

Uma abordagem natural para se contornar tais dificuldades é utilizarmos o método de geração de colunas na obtenção da relaxação linear de (6)-(8) (isto é, (8) é substituído por $0 \le x_j \le 1, \forall\ j \in J$). Esta técnica foi aplicada inicialmente ao problema de roteamento de veículos por Rao e Zionts[68], Foster e Ryan[76], Orloff[76] entre outros.

Na geração de colunas são construídos, a cada iteração, problemas reduzidos contendo um subconjunto de todas as possíveis colunas (variáveis). A relaxação linear do problema reduzido gera variáveis duais ótimas λ. Para checarmos a otimalidade da solução, calculamos uma coluna **s** (associada a uma variável x_s) e satisfazendo a:

$$c_s^* - \lambda y_s = \min_{j \in J} \{c_j^* - \lambda y_j\} \qquad (9)$$

onde y_j é o vetor coluna associado à variável x_j. Se o custo reduzido associado à variável x_s for positivo, a solução corrente será ótima. Caso contrário, x_s entra na base e o problema é reotimizado. Como no problema de roteamento de veículos as soluções viáveis devem ser inteiras, este procedimento deve ser utilizado conjuntamente com o algoritmo *branch-and-bound* como forma a garantir soluções ótimas. Desrosiers, Soumis e Desrochers[84] resolvem (9) utilizando um algoritmo para o problema do caminho mínimo com as mesmas restrições que o problema original de roteamento.

A técnica de programação dinâmica foi originalmente proposta para o problema de roteamento de veículos por Eilon, Watson-Gandy e Christofides[71]. Outras técnicas geram limites inferiores utilizando programação dinâmica. Na programação dinâmica, a principal etapa é a determinação de uma fórmula de recorrência. A solução ótima do problema original ou de uma relaxação para o problema original é então obtida iterativamente a partir da base do processo recursivo. Entre os principais trabalhos que utilizam programação dinâmica podemos citar Christofides et al.[81.a], a relaxação de espaços de estado de Christofides, Mingozzi e Toth[81.b] e mais recentemente, Hadjiconstantinou, Christofides e Mingozzi[95].

Em Lucena[86], é proposto um algoritmo exato para o problema de roteamento (PRVRC). O procedimento elimina inicialmente várias rotas sub-ótimas através do uso de limitantes e condições de dominância derivadas da relaxação de espaços de estados. Um conjunto de rotas contendo uma solução ótima é identificado e um problema de partição (*Set partitioning*) é resolvido. Uma abordagem envolvendo uma formulação de dois fluxos é também apresentada.

Em Araque, Kudva, Morin e Pekny[94]; Hill[95] e Augerat[95] são propostos algoritmos do tipo *branch-and-cut,* na solução do problema clássico de roteamento. Em todas as abordagens apresentadas são resolvidas instâncias que variam de 50 a 150 clientes. Novas classes de restrições válidas para o problema são também apresentadas.

Capítulo II

Teoria Poliédrica

II.1 - Introdução:

Pode-se dizer que os primeiros resultados sobre a teoria de poliedros convexos tiveram início com os tratados de Euclides na época dos gregos . Euclides estudava por exemplo, a fórmula de alguns poliedros em 2 e 3 dimensões. A primeira contribuição considerável após este período é devida provavelmente a Euler, quando estabeleceu a relação entre o número de vértices, faces e arestas de politopos em 3 dimensões (Grümbaum[67]). Alguns destes resultados entretanto, já eram conhecidos 100 anos antes por Descartes . Outras contribuições se seguiram, Schläfli[1901], Minkowski[1911] entre outras, até que em 1934 Steinitz e Rademacher (nova edição em 1976), publicaram o primeiro trabalho "estado-da-arte" sobre o assunto.

Como discutido em Pulleyblank[83], a análise de problemas combinatórios utilizando-se conceitos e resultados da teoria poliédrica pode ser dividida basicamente em 3 períodos, todos eles demarcados pelos grandes avanços na área de teoria e construção de algoritmos.

O primeiro período teve início na década de 50 com o surgimento do método simplex para programação linear (desenvolvido e aprimorado principalmente pelos

trabalhos de Dantzig, Hoffman, Kantorovich, Koopmans entre outros). Apesar de tratar-se de um problema contínuo, muitos problemas combinatórios (em particular, vários problemas de fluxo em redes) possuem uma matriz de restrições totalmente unimodular. Isto permite que diversos problemas "inteiros" possam ser resolvidos de maneira exata utilizando-se apenas o algoritmo simplex.

O segundo período teve início em meados da década de 60 quando Jack Edmonds mostrou que era possível tratar de problemas inteiros um pouco mais gerais. Através da adição, possivelmente exponencial de restrições ao problema original, ele obtinha uma descrição linear completa do problema, de maneira que o problema dual associado fosse resolvido mesmo quando o algoritmo simplex não pudesse ser aplicado diretamente ao problema. Desta forma, algoritmos de complexidade polinomial podiam ser obtidos utilizando-se alguns dos resultados da teoria de dualidade em programação linear. Esta abordagem se mostrou ineficaz para uma grande quantidade de problemas combinatórios (hoje conhecidos como NP-Árduos). Para estes problemas, uma descrição linear completa não pode ser obtida (a menos que P=NP). Apesar disso, a utilização da teoria de dualidade juntamente com métodos de enumeração tem demonstrado resultados bastante interessantes.

O terceiro período teve início em 1979 com o desenvolvimento do método dos elipsóides em programação linear (Shor[77]] e Kachian [79]). Apesar de tratar-se do primeiro algoritmo polinomial para programação linear, o algoritmo de Kachian não tinha o mesmo desempenho computacional quando comparado ao método simplex (de complexidade exponencial). Além do excelente resultado teórico para programação linear, a estrutura do método dos elipsóides traria também importantes conseqüências para a combinatória poliédrica. Este algoritmo permite responder eficientemente se um dado ponto, pertence ou não a um conjunto definido por um sistema linear (poliedro). Caso o ponto dado não pertença, o algoritmo permite a determinação de uma desigualdade violada (problema de separação). Sempre que o problema de separação for resolvido de maneira eficiente (em tempo polinomial), o problema de otimização também o será. Da mesma forma, pode-se demonstrar que a recíproca também é verdadeira. Assim, para poliedros associados a problemas de otimização NP-Árduos, não podemos esperar que algoritmos

polinomiais sejam encontrados para o problema de separação correspondente (a menos obviamente que P=NP). Maiores detalhes sobre este assunto podem ser encontrados em Grötschel, Lováz e Schrijver [88].

Apresentamos agora uma breve descrição de alguns resultados da teoria poliédrica e álgebra linear necessários ao estudo da combinatória poliédrica (em particular ao estudo do Caixeiro Viajante e Roteamento de Veículos).

II.2 - Definições Básicas:

Dizemos que um conjunto $P \subseteq \Re^n$ define um *poliedro* se $P = \{x \in \Re^n \ / \ Ax \leq b\}$, onde $A \in \Re^{mxn}$ e $b \in \Re^m$ (escrevemos também P=(A,b)). Note que o poliedro é obtido através da interseção finita de semi-espaços de $\Re^n$. Diremos que o poliedro P é *limitado* se pudermos inscrevê-lo totalmente em uma bola de raio r. Caso nosso poliedro P seja limitado, teremos então um *politopo*.

Sejam $x_1,..,x_n \in \Re^n$ e $\lambda_1,..,\lambda_n \in \Re$. Dizemos que o vetor $x = \lambda_1 x_1 + ... + \lambda_n x_n$ é uma *combinação linear* dos vetores $x_1,..,x_n$. Se, além disso tivermos $\lambda_1 + ... + \lambda_n = 1$, dizemos que **x** é uma *combinação linear afim* de $x_1,..,x_n$. Se **x** é combinação linear afim e $\lambda_i \geq 0 \ \ p/ \ i = 1,..,n$; teremos uma *combinação linear convexa* dos vetores $x_1,..,x_n$.

A *envoltória convexa* de um conjunto discreto $X = \{x_1,..,x_k\} \subseteq \Re^n$ é definida como:

$$conv(X) = \left\{ x \in \Re^n \ / \ x = \sum_{i=1}^{k} \lambda_i x_i ; \ \sum_{i=1}^{k} \lambda_i = 1 \ e \ \lambda_i \geq 0 \ \ p/ \ i = 1,..,,k \right\}$$

Como veremos posteriormente, nos problemas combinatórios, estaremos sempre interessados na envoltória convexa de um conjunto de vetores de $\Re^n$ com coordenadas inteiras.

Considere agora o seguinte conjunto de índices $M = \{1,2,\ldots,m\}$ representando cada uma das linha de P=(A,b). Considere também $M^{=} = \{i \in M \ / \ a_i x = b_i, \ \forall x \in P\}$ e

$M^{\leq} = \{i \in M \,/\, a_i x < b_i,\ \textit{para algum}\ \ x \in P\}$ dois subconjuntos de índices. Temos então um conjunto de igualdades $(A^{=}, b^{=})$ e um conjunto de desigualdades $(A^{\leq}, b^{\leq})$ onde:]

$$P = \{x \in \Re^n \,/\, A^{=} x = b^{=},\ A^{\leq} x \leq b^{\leq}\}$$

Note que P continua definindo um poliedro já que cada igualdade pode ser desmembrada em duas desigualdades.

A *dimensão* de um poliedro P é igual a dimensão do menor espaço afim que o contém. Assim, temos dim(P) = dim(aff(P)), onde $aff(P) = \{x \in \Re^n \,/\, A^{=} x = b^{=}\}$. Segue então que dim(P) = n - posto$(A^{=}, b^{=})$. Caso posto$(A^{=}, b^{=}) = 0$, dizemos que $P \subseteq \Re^n$ tem *dimensão plena.*

Na figura seguinte representamos 2 poliedros de $\Re^3$ (de dimensão 2 e 3 respectivamente):

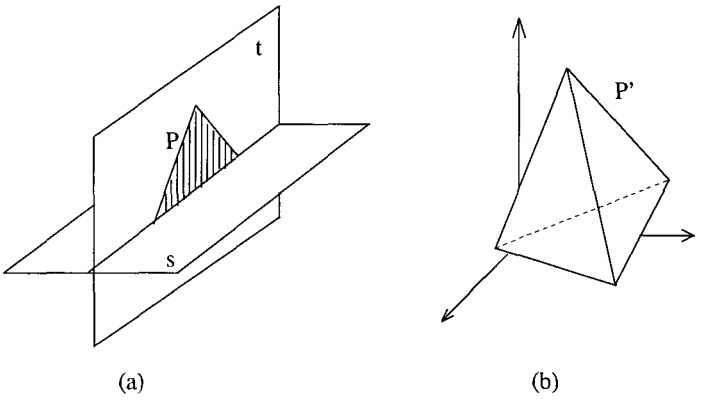


Figura II.1: Poliédros de dimensão 2 e 3

Note que o poliedro P é formado por uma restrição de igualdade e 3 desigualdades, enquanto P' é formado por 4 desigualdades (dimensão plena).

Uma desigualdade $a^T x \leq b$ é *válida* com respeito a um conjunto $P \subseteq \Re^n$, se $P \subseteq \{x \in \Re^n \,/\, a^T x \leq b\}$, ou seja, P está contido no subespaço definido por $a^T x \leq b$. Além disso, esta desigualdade válida será *suporte* para P, se $P \cap \{x \in \Re^n \,/\, a^T x = b\} \neq \varnothing$.

Uma desigualdade válida $a^T x \leq b$ (com respeito a P) será *própria*, se P não estiver contida no hiperplano $\{x \in \Re^n / a^T x = b\}$. Na figura II.1(a) acima, a desigualdade associada ao plano **s** é própria já que P não está contido em **s**.

Um subconjunto $F \neq \varnothing$ de um poliedro P é uma *face* de P se existir uma desigualdade $a^T x \leq a_0$ válida com respeito a P tal que $F = \{x \in P / a^T x = a_0\}$. Dizemos que a desigualdade $a^T x \leq a_0$ define uma face de P. Como $P = \{x \in P / 0^T x = 0\}$, o poliedro P é uma face de si próprio. Além disso, como $\{x \in P / 0^T x = 1\} = \varnothing$ o conjunto vazio também define uma face de P.

Uma face F será *própria* se $\varnothing \neq$ F $\neq$ P. Assim, se $P = \{x \in \Re^n / a_i^T x \leq b_i,\ p/\ i = 1,..,k\}$ é um poliedro e F uma face de P, então existirá um subconjunto não vazio de índices $I \subseteq \{1,2,\ldots,k\}$ tal que $F = \{x \in P / a_i^T x = b_i,\ i \in I\}$.

Sejam $a^T x \leq a_0$ e $b^T x \leq b_0$ duas desigualdades válidas de um poliedro P. Se $\{x \in P / a^T x = a_0\} = \{x \in P / b^T x = b_0\}$, dizemos que $a^T x \leq a_0$ e $b^T x \leq b_0$ são *equivalentes* com respeito a P. Na figura II.2, representamos um exemplo em $\Re^3$ que ilustra a equivalência de duas desigualdades válidas definidas pelos planos **r** e **s**.

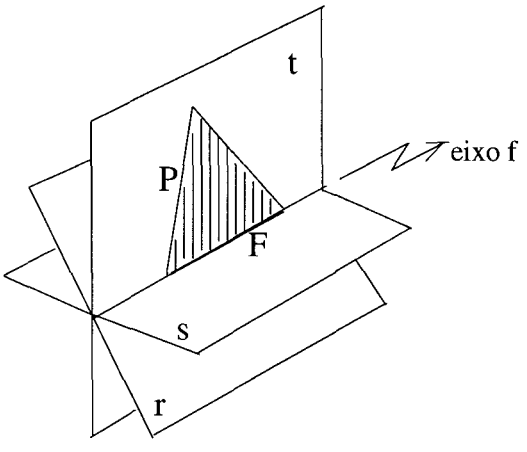


Figura II.2: Os planos r e s definem a mesma face F de P.

Note que **r** e **s** se interceptam no eixo **f**. Ambos definem a mesma face F de P (F é própria já que F ≠ P).

II.3 - Caracterização de Facetas:

Dizemos que uma face própria F de P, define uma *faceta*, se F for maximal com respeito à inclusão de conjuntos. Neste caso, nenhuma outra face própria de P poderá conter F.

Usualmente, nos problemas de otimização combinatória, o poliedro associado é dado como a envoltória convexa de um número finito de pontos. Para se aplicar técnicas de programação matemática que resolvam estes problemas, é necessário descrever esta envoltória convexa como um sistema de desigualdades lineares. O maior problema neste caso, é definir este sistema com o menor número de desigualdades possível. Portanto, a utilização de desigualdades que definam facetas se torna de suma importância no estudo dos problemas combinatórios.

O teorema seguinte estabelece que condições devem ser obedecidas para que uma desigualdade defina uma faceta:

Teorema II.1: (Facetas de um poliedro)

Seja $P = \{x \in \Re^n \ / \ A^{=} x = b^{=},\ A^{\leq} x \leq b^{\leq}\} \subseteq \Re^n$ um poliedro qualquer onde $A^{=} \in \Re^{mxn}$ e $b^{=} \in \Re^m$. Considere ainda aff(P)=$\{ x \in \Re^n \ / \ A^{=} x = b^{=} \}$, o espaço afim que contém P. Uma face F de P será uma *faceta* se:

i) F for uma face maximal própria de P;

ii) dim(F) = dim(P) - 1;

iii) existir uma desigualdade $c^T x \leq c_0$, válida com respeito a P com as seguintes propriedades:

a) $F \subseteq \{x \in P \ / \ c^T x = c_0\}$;

b) $\exists\, \overline{x} \in P$ *tal que* $c^T \overline{x} < c_0$, ou seja, a desigualdade é própria;

c) se qualquer desigualdade $d^T x \leq d_0$, válida com respeito a P satisfizer $F \subseteq \{x \in P \ / \ d^T x = d_0\}$, então existirá um escalar $\alpha \geq 0$ e um vetor $\lambda \in \Re^m$ tal que:

$$d^T = \alpha c^T + \lambda^T A^=$$
$$d_0 = \alpha c_0 + \lambda^T b^=$$

Na figura II.3, a seguir representamos um exemplo que ilustra a condição (iii) do teorema anterior:

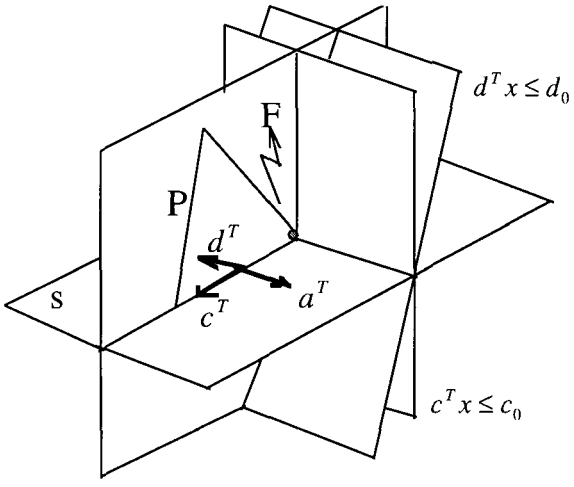


Figura II.3: Verificação da condição (iii).

Suponha que a matriz $A^= x = b^=$, seja representada por uma equação do tipo $a^T x = a_0$ (como na figura acima). Note que o vértice F representado acima não define uma faceta de P já que o vetor d^T (ortogonal a $d^T x \le d_0$) não pode ser escrito como combinação linear dos vetores a^T e c^T, ortogonais a $a^T x = a_0$ e $c^T x = c_0$ respectivamente!

O mesmo não ocorre se tomamos a face F representada na figura II.2. Note que qualquer que seja a desigualdade válida $d^T x \le d_0$ que contenha F, a condição (iii) se verifica. Observe também que $c^T x \le c_0$ e $d^T x \le d_0$ são desigualdades equivalentes!

A caracterização de facetas é de suma importância na determinação de um conjunto mínimo de desigualdades que descrevam um poliedro $P \subseteq \Re^n$. Um sistema de igualdades

$A^{=}x = b^{=}$ *e* $A^{\leq}\ x \leq b^{\leq}$ é dito *completo* com respeito a um poliedro $P \subseteq \Re^n$, se $P = \{x \in \Re^n \ / \ A^{=}x = b^{=},\ A^{\leq}\ x \leq b^{\leq}\}$.

O teorema seguinte apresenta importantes resultados para a caracterização mínima de desigualdades que representam P:

Teorema II.2: (Caracterização de poliedros completos e não redundantes)

Seja $P \subseteq \Re^n$ um poliedro e $A^{=}x = b^{=}$ *e* $A^{\leq}\ x \leq b^{\leq}$ um sistema completo e não redundante para P, onde $A^{=} \in \Re^{mxn}$ e $A^{\leq} \in \Re^{kxn}$. Então temos as seguintes propriedades:

i) $aff(P) = \{x \in \Re^n \ / \ A^{=}x = b^{=}\}$ e posto($A^{=}$) = m;

ii) dim(aff(P)) = dim(P) = n - m;

iii) Toda desigualdade $a_i^T x \leq a_0$ do sistema $A^{\leq}\ x \leq b^{\leq}$ define uma faceta F_i de P, onde $F_i = \{x \in P \ / \ a_i^T x = b_i\}$, *para* $i = 1,\ldots,k$;

iv) Se $\overline{a}_i^T x \leq \overline{b}_i$, $i = 1,\ldots,\overline{k}$ (associado a $\tilde{A}^{\leq}\ x \leq \tilde{b}^{\leq}$) e $\tilde{a}_i^T x = \tilde{b}_i$, $i = 1,\ldots,\overline{m}$ (associado a $\tilde{A}^{=}\ x = \tilde{b}^{=}$) for outra representação completa e não redundante para P então:

d1) $k = \overline{k}$ *e* $m = \overline{m}$;

d2) $\tilde{a}_i^T = (\lambda^i)^T A^{=}$ para algum $\lambda_i \in \Re^m - \{0\}$ onde i=1,..,m;

d3) $\overline{a}_i^T = \alpha_i a_j^T + (\lambda_i)^T A^{=}$ para algum $\alpha_i > 0$, $\lambda^i \in \Re^m$, $j \in \{1,..,k\}$ e i=1,...,$\overline{k}$ •

Note em particular que, para um poliedro P=(A,b) de dimensão plena, temos um *único* sistema de desigualdades completo e não redundante que o descreve. Assim, se $A^{\leq}$ é formado por desigualdades $\overline{a}_i^T x \leq \overline{b}_i$, *para* $i = 1,\ldots,\overline{k}$, deverá satisfazer k = $\overline{k}$ e $\overline{a}_i = \alpha_i a_i$ para algum $\alpha_i > 0$.

A mesma representação única não ocorre se tivermos um sistema de igualdades e desigualdades na representação de P (P não é de dimensão plena).

Alguns problemas combinatórios permitem que seu poliedro associado $P \subseteq \Re^n$ (de dimensão menor que n), seja expandido para um poliedro $\overline{P}$ de dimensão plena. Neste caso, será fundamental uma conversão dos resultados obtidos em $\overline{P}$ para P. Note por exemplo, que o problema da equivalência de desigualdades é facilmente resolvido em um poliedro de dimensão plena!

II.4 - Considerações adicionais:

Em algumas situações, estaremos interessados na obtenção de desigualdades válidas mais fortes, ou até mesmo facetas, a partir das faces ou restrições válidas de um poliedro P.

Considere $\pi x \leq \pi_0$ e $\beta x \leq \beta_0$ (onde $x \in \Re^n_+$), restrições válidas para um poliedro P. Diremos que $\beta x \leq \beta_0$ é um *lifting* (melhoramento) da desigualdade $\pi x \leq \pi_0$ se existir um α>0 tal que $\beta \geq \alpha\pi$ e $\beta_0 \leq \alpha\pi_0$.

Note que se uma desigualdade define uma faceta para P então ela não poderá sofrer o processo de *lifting*.

Seja $S = \{e_1, e_2, .., e_n\}$ um conjunto de n elementos, e U um subconjunto de S. Dizemos que x^U é o *vetor de incidência* associado a U, se x^U pertencente a $\Re^n$ for tal que, a i-ésima componente de x^U é igual a 1 se $e_i \in U$, e 0 em caso contrário.

Representarmos por 2^S o conjunto de todos os subconjuntos de S. Se Γ é um conjunto de subconjuntos de S, diremos que P_Γ é a envoltória convexa de todos os vetores de incidência de elementos de Γ, ou seja:

$$P_\Gamma = conv\left\{x^U \in \Re^n \ / \ U \in \Gamma\right\}$$

É fácil ver que cada vértice de P_Γ corresponde a um conjunto em Γ e vice-versa.

Considere pesos ou distâncias $c_e \in \Re$, associadas a cada um dos elementos de S. Teremos então o seguinte problema de otimização:

$$min\{c^T x \,/\, x \in P_\Gamma\} \qquad (1)$$

Note que, cada um dos vértices do P_Γ tem coordenadas inteiras (vetores de incidência de elementos de Γ). Para resolução de (1) utilizando técnicas de programação linear, necessitamos de uma descrição completa e não redundante das restrições que definem o politopo P_Γ. Acredita-se entretanto, que para problemas NP-completos essa descrição completa não seja possível (a menos que P=NP). Apesar disso, uma descrição poliédrica parcial de P_Γ pode ser interessante se combinarmos técnicas de programação linear com os métodos de busca em árvore (vide Nemhauser&Wolsey[88]).

Capítulo III

Relaxação Lagrangeana com Geração de Restrições:

III.1 - Introdução:

Recentemente, um número considerável de problemas de otimização combinatória vem sendo resolvidos com a utilização de resultados provenientes da teoria poliédrica. Neste capítulo, descrevemos uma técnica que combina relaxação lagrangeana com a geração de desigualdades válidas (idealmente facetas) na geração de limites inferiores para o problema original.

Fazemos inicialmente uma introdução sobre alguns dos conceitos e técnicas utilizadas no decorrer do trabalho e apresentamos em seguida (seções III.3 e III.4) a descrição da técnica que combina relaxação lagrangeana com geração de restrições. Exemplos de aplicações desta técnica podem ser encontrados em Escudero et al. [94], Fisher[94.b], Lucena[96], Miller[95] entre outros.

III.2 - Considerações Iniciais:

Representaremos por $\{0,1\}^n$, o conjunto de todos os vetores de $\Re^n$ com coordenadas inteiras 0 e 1. O problema geral de programação linear inteira 0-1 pode então ser expresso da seguinte forma:

$$\begin{aligned} & min\ c^T x \\ & s.a\ \begin{cases} Ax \le b \\ Bx \le d \end{cases} \qquad (P_0) \\ & \quad x \in \{0,1\}^n \end{aligned}$$

onde $A \in \Re^{mxn}; B \in \Re^{xn}; c,x \in \Re^n; b \in \Re^m$ e $d \in \Re$.

A **região viável** (ou **conjunto viável**) associada a P_0 é definida por $M_0 = \{x \in \{0,1\}^n \ / \ Ax \le b \ e \ Bx \le d\}$, enquanto $z = c^T x$ é a **função objetivo** a ser minimizada (ou maximizada) em M_0.

Como ocorre freqüentemente com os problemas de programação linear inteira, ao invés de minimizarmos a função objetivo diretamente sobre o conjunto viável original, poderá ser mais interessante o particionamento desta região em um número finito de regiões menores. Trabalhamos então na minimização da função objetivo em cada uma destas regiões separadamente. Além disso, este processo deverá ser realizado de maneira que uma solução ótima do problema original pertença a pelo menos uma das regiões particionadas.

Caso os problemas de otimização associados a cada uma das regiões menores, não sejam ainda resolvidos "facilmente" (em tempo polinomial) através das ferramentas disponíveis, particionamos novamente estas regiões até que seja possível resolvê-las.

Na figura III.1, representamos o problema original P_0 e os diversos subproblemas P_k, obtidos após o particionamento sucessivo da região viável M_0 (associada a P_0).

Como o número de subproblemas obtidos pode ser extremamente grande, necessitamos de um esquema de enumeração que permita explicitá-los apenas parcialmente, não descartando obviamente, aqueles subproblemas cuja região viável correspondente contenha uma solução ótima do problema original. Assim, dado um subproblema P_k

qualquer, deveremos ser capazes de decidir por um novo particionamento de P_k, ou então excluí-lo definitivamente por não ter uma região viável associada M_k que contenha uma solução ótima do problema original.

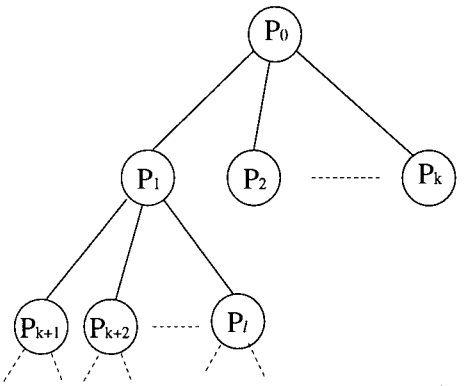


Figura III.1: Árvore de busca

Para que esse esquema de enumeração possa ser implementado na prática, utilizaremos **relaxações** P_k' de cada um dos subproblemas P_k. Mais formalmente:

$$\begin{aligned} &min\ w(x) \\ &s.a\quad x \in R_k \end{aligned} \qquad (P_k')$$

onde $M_k \subseteq R_k$ e $w(x) \leq c^T x,\ \forall\ x \in R_k$. Claramente, uma solução ótima de P_k' nos dará um limite inferior para P_k. Como veremos posteriormente, para que um esquema eficiente de enumeração seja implementado, cada uma das relaxações P_k' deverá ser resolvida "facilmente" (em tempo polinomial).

Para ilustrar essa definição, consideremos novamente o problema original P_0 juntamente com M_0 (seu conjunto viável associado). Caso as restrições de integralidade $x \in \{0,1\}^n$ em M_0 sejam substituídas por $x \in [0,1]^n$ (conjunto de todos os vetores de $\Re^n$ com coordenadas entre 0 e 1) diremos que a nova região $R_0 = \{x \in [0,1]^n \ / \ Ax \leq b \ e \ Bx \leq d\}$

juntamente com a função objetivo $w(x) = c^T x$ define uma relaxação para P_0. Mais particularmente, teremos uma **relaxação linear** para P_0.

Outros tipos de relaxação podem obviamente ser utilizadas, como por exemplo, a **relaxação lagrangeana** (Geoffrion[74], Fisher[81]), discutida a seguir.

Considere z*, o valor da solução ótima do problema original (P_0). Dualizando-se as restrições Bx≤d e associando a elas um vetor de multilicadores λ≥0, teremos o seguinte problema relaxado (ou lagrangeano):

$$\begin{aligned} w_0(x,\lambda) = min\ & c^T x + \lambda^T (Bx - d) \\ s.a\ & Ax \le b \qquad\qquad (P_0') \\ & x \in \{0,1\}^n \end{aligned}$$

onde as componentes de $\lambda \in \Re_+^r$ são denominadas multiplicadores de Lagrange. Note neste caso que $M_0 \subseteq R_0 = \{x \in \{0,1\}^n / Ax \le b\}$ e $w_0(x,\lambda) \le z^*$ qualquer que seja λ≥0 (Geoffrion [74]).

Dizemos que a relaxação lagrangeana (P_0') possui a **propriedade de integralidade**, se a solução de (P_0') (para todos os multiplicadores de Lagrange λ), permanece inalterada após a troca das restrições de integralidade $x \in \{0,1\}^n$ por $x \in [0,1]^n$. Isto significa que o maior limite inferior obtido após a solução de (P_0') é igual ao valor da relaxação linear associada ao problema original. Caso contrário, se a relaxação lagrangeana não tem a propriedade de integralidade, o maior limite inferior obtido para (P_0') é maior ou igual ao valor da relaxação linear associada ao problema original.

Idealmente, desejamos encontrar o melhor limite inferior possível. Para isto, resolvemos o seguinte problema *dual lagrangeano*:

$$L_0^* = \max_{\lambda \ge 0} w_0(x,\lambda) \qquad (D_0)$$

É bem conhecido que w_0 é uma função côncava (afim por partes), mas não necessariamente diferenciável em todos os pontos do domínio. Além disso, $L_0^* \leq z^*$. O valor ótimo em D_0 (obtido para $\lambda = \lambda^*$) nos dará o maior limite inferior possível. Esses multiplicadores podem ser gerados utilizando-se, por exemplo, o método subgradiente com convergência demonstrada por Held, Crowder e Wolfe[74]. Dados multiplicadores λ, devemos encontrar uma direção de "subida" buscando a maximização de $w(x,\lambda)$ (Geoffrion[74], Fisher[81]).

Se $\bar{x}$ é uma solução do problema relaxado, pode-se mostrar facilmente que cada uma das componentes $\sigma_i = b_i\bar{x} - d_i$ (p/ i=1,..,r) são utilizadas na determinação do vetor subgradiente associado à relaxação lagrangeana corrente (Geoffrion[74]).

Os novos multiplicadores de Lagrange (associados às desigualdades dualizadas) são atualizados da seguinte forma:

$$\lambda_i := max\{0,\ \lambda_i + \theta\sigma_i\};\quad i = 1,2,\ldots,r.$$

onde o tamanho do passo θ é usualmente determinado através da expressão:

$$\theta = \frac{\pi(z_{UB} - z_{LB})}{\sum_{i=1}^{r}\sigma_i^2}$$

sendo $z_{LB} = w(\bar{x},\lambda)$ e z_{UB}, limites inferior e superior respectivamente de z*, e π um escalar que assume valores no intervalo (0,2]. Como discutido em Lucena[96], quanto maior for o número de subgradientes obtidos, menor será o tamanho do passo θ e maior o tempo de processamento exigido na solução do problema relaxado.

Assim, para implementarmos eficientemente o esquema de enumeração como descrito anteriormente, devemos trabalhar na solução das relaxações P_k' sempre que uma solução de P_k não puder ser obtida "facilmente" através das ferramentas disponíveis.

Esse esquema de enumeração pode ser ainda incrementado se conhecemos bons limites superiores v* (associado às soluções viáveis) do problema original P_0. Desta forma, se um subproblema P_k qualquer tiver um limite inferior maior que v*, a região viável associada M_k poderá ser descartada definitivamente de nosso processo de busca por não conter uma solução ótima de P_0.

Esse processo de busca em árvore é mais usualmente conhecido na literatura como **branch-and-bound** (dividir e limitar), por se tratar de um processo de partições sucessivas do conjunto viável e pela geração de limites (inferiores e superiores) associados a cada um dos subproblemas gerados. Para maiores detalhes sobre este assunto vide Murty[76], Nemhauser& Wolsey[88].

Através da relaxação linear do modelo associado ao problema original (ou do modelo original acrescido de outras desigualdades válidas geradas previamente), podemos buscar a identificação e a inserção de restrições que são violadas pela solução do problema relaxado corrente. Procedendo desta forma, obtemos uma melhor aproximação da envoltória convexa do conjunto viável original. Caso a solução da relaxação corrente não nos dê uma solução viável para o problema original e a identificação de novas restrições violadas não seja mais "possível", fazemos uma partição (ou *branching*) do conjunto viável original e repetimos o processo para cada um dos subconjuntos gerados. Denominamos **branch-and-cut**, ao processo que combina geração de restrições válidas, a partir de relaxações lineares, com os métodos de busca em árvore (*branch-and-bound*). Alguns exemplos de aplicação desta técnica podem ser encontrados em Padberg e Rinaldi[91] para o problema do caixeiro viajante e Augerat[95] para o problema de roteamento de veículos.

Como discutido no capítulo anterior, a determinação de desigualdades violadas é conhecida como o problema da separação e nem sempre pode ser realizada em tempo polinomial (a menos que P=NP). Neste caso, soluções heurísticas podem ser utilizadas. Como demonstrado em Schrijver[86], sempre que o problema de separação for NP-Completo o problema de otimização associado será NP-Árduo.

Uma das dificuldades presentes nos métodos que utilizam relaxação linear é que a estrutura (ou conjunto de restrições) inerente ao modelo original é destruída sempre que uma nova desigualdade válida é adicionada ao problema. Uma alternativa interessante seria

resolvermos uma seqüência de problemas combinatórios mais simples de mesma estrutura ou conjunto de restrições para obtenção de limites inferiores. Isto pode ocorrer se utilizamos técnicas baseadas em relaxação lagrangeana ao invés de relaxação linear.

Apresentamos neste trabalho uma técnica que combina geração de desigualdades válidas (idealmente facetas) e relaxação lagrangeana. Exemplos de aplicações desta técnica podem ser encontradas em Fischer[94.b], Kohl e Madsen[97], Escudero et al.[94], Miller[95], Lucena[96], entre outros.

Neste método, adicionamos e dualizamos restrições válidas que são violadas por uma solução do problema relaxado (problema lagrangeano). Como veremos, esta técnica permite um maior incremento do limite inferior nos problemas de otimização combinatória que utilizam apenas relaxação lagrangeana.

III.3 - Descrição da Técnica:

Consideremos novamente o seguinte problema geral de programação linear inteira 0-1:

$$\begin{aligned} & min\ c^T x \\ & s.a \begin{cases} Ax \le b \\ Bx \le d \end{cases} \qquad (P_0) \\ & x \in \{0,1\}^n \end{aligned}$$

onde $A \in \Re^{mxn}; B \in \Re^{rxn}; c,x \in \Re^n; b \in \Re^m$ e $d \in \Re^r$.

Seja $\Psi = \{x \in \{0,1\}^n \ / \ Ax \le b \ e \ Bx \le d\}$ o domínio de nosso problema original (P_0) e conv(Ψ) sua envoltória convexa.

Como discutido anteriormente o problema dual lagrangeano associado será:

$$L_0^* = \max_{\lambda \ge 0} \ w_0(x,\lambda) \qquad (D_0)$$

Se gerarmos um conjunto de desigualdades válidas $Hx \le f$ para conv(Ψ) (diferentes das já utilizadas), teremos o seguinte problema (P_1) equivalente ao problema original (P_0):

$$\begin{aligned} & min\ c^T x \\ & s.a \begin{cases} Ax \le b \\ Bx \le d \\ Hx \le f \end{cases} \qquad (P_1) \\ & x \in \{0,1\}^n \end{aligned}$$

onde A, B, b, c, d são como definidos anteriormente e $H \in \Re^{kxn}$ e $f \in \Re^k$. Dualizando também o novo conjunto de restrições teremos o seguinte problema dual lagrangeano associado a (P_1):

$$L_k^* = \max_{\substack{\lambda \ge 0 \\ \mu \ge 0}} w_k(x, \lambda, \mu) \qquad (D_1)$$

onde:

$$\begin{aligned} w_k(x,\lambda,\mu) = min\ & c^T x + \lambda^T (Bx - d) + \mu^T (Hx - f) \\ s.a\ & Ax \le b \qquad (P_1') \\ & x \in \{0,1\}^n \end{aligned}$$

A seguinte proposição, apresentada em Aboudi et al.[91], garante que o novo limite inferior obtido é maior ou igual ao anterior:

Teorema III.1 - Sejam L_0^* e L_k^* o valor da solução ótima dos problema duais (D_0) e (D_1) respectivamente. Então $L_0^* \le L_k^*$.

Demonstração:

Considere λ^* a solução ótima para o problema dual (D_0) e o par $\bar{\lambda}, \bar{\mu}$ os multiplicadores ótimos para o problema dual (D_1). É fácil ver que:

$$w_0(x,\lambda) = w_k(x,\lambda,0) \qquad \forall\ \lambda \ge 0.$$

Segue então que:

$$L_0^* = w_0(x^*, \lambda^*) = w_k(x^*, \lambda^*, 0) \le w_k(\bar{x}, \bar{\lambda}, \bar{\mu}) = L_k^*$$

onde x* é uma solução ótima de w_0 associada a λ^* e $\overline{\mathbf{x}}$ é uma solução ótima de w_k associada a $\overline{\lambda}$ *e* $\overline{\mu}$. •

Note que a introdução de desigualdades válidas para o problema original na função lagrangeana pode melhorar o limite inferior para o valor da solução ótima z*. Caso o número de restrições em Hx≤f seja pequeno, dualizá-los de forma lagrangeana não representa uma dificuldade. No entanto, quando este número é grande (possivelmente uma função exponencial de n) a utilização de todas as restrições se torna difícil. Quando isto ocorre, pode-se utilizar um número pequeno daquele conjunto de restrições ou permitir que todas as restrições do conjunto possam ser candidatas à dualização, utilizando, no entanto, apenas um número pequeno destas. Caso a segunda opção seja escolhida surge a questão de como implementá-la. Uma sugestão natural, seria tentar adaptar o procedimento utilizado em relaxação linear, ou seja, à cada problema lagrangeano resolvido, procura-se identificar desigualdades de Hx≤f que sejam violadas. Tais desigualdades seriam então dualizadas. Caso este procedimento seja seguido, algumas questões se colocam:

i) Dada uma solução do problema lagrangeano, como obter desigualdades válidas (idealmente facetas) violadas por esta solução?
ii) Que valor deve ser associado aos novos multiplicadores?
iii) Se uma desigualdade válida é obtida, que condições devem ser obedecidas para se garantir um crescimento estrito do limite inferior?

Na primeira questão, a identificação de desigualdades violadas depende fundamentalmente das características presentes em cada problema. No caso de uma relaxação linear por exemplo, a solução obtida satisfaz todas as restrições do problema original, à exceção das restrições de integralidade (ou seja, as coordenadas associadas à relaxação linear assumem valores fracionários). Assim, um problema não-trivial de separação deve ser "resolvido" buscando a determinação de desigualdades violadas. O mesmo ocorre no problema lagrangeano. A única diferença é que a solução obtida é inteira e satisfaz apenas as restrições do tipo Ax≤b. Como veremos, poderá ser mais "fácil"

resolver a identificação de restrições violadas tomando-se uma solução do problema lagrangeano do que o problema de separação quando tomamos uma solução obtida através de relaxação linear. Isto ocorre basicamente por preservamos a estrutura (ou conjunto de restrições) do problema relaxado além de se tratar de soluções com coordenadas inteiras.

Para a segunda pergunta assumimos um valor zero para os multiplicadores de Lagrange e os atualizamos utilizando alguma das fórmulas para atualização de multiplicadores (p. ex. Geoffrion[74]). Nada indica entretanto, ser esta a maneira ideal de proceder, embora, como veremos mais tarde, tem funcionado muito bem para os problemas pesquisados.

Na terceira questão, não são conhecidas ainda condições suficientes que garantam um incremento estrito do limite inferior. O trabalho desenvolvido por Aboudi et al. [91], foi provavelmente o primeiro artigo a tratar esse assunto. Obviamente, nem todas as desigualdades válidas que são violadas pela solução do problema lagrangeano permitem um incremento estrito do limite inferior.

Em Aboudi et al. [91] são apresentadas duas condições necessárias para esse crescimento estrito.

Suponha inicialmente que **k** restrições já tenham sido dualizadas e que $X = \{x \in \{0,1\}^n \,/\, Ax \leq b\}$ seja o conjunto de restrições associadas ao problema lagrangeano. Suponha ainda que $h_{k+1}x \leq f_{k+1}$ seja a próxima restrição a ser dualizada.

Temos então a seguinte proposição (Aboudi et al.[91]):

Teorema III.2: Se $h_{k+1}x \leq f_{k+1}$, $\forall\ x \in X$ então $L_k^* = L_{k+1}^*$.

Demonstração:

Da proposição anterior temos que $L_k^* \leq L_{k+1}^*$ (demonstração análoga).

Considere agora dois vetores λ e μ tais que $\lambda \in \Re^r$ *e* $\mu \in \Re^k$. Como $h_{k+1}x \leq f_{k+1}$, $\forall\ x \in X$ e $\mu_{k+1} \geq 0$ temos que $\mu_{k+1}(h_{k+1}x - f_{k+1}) \leq 0$.

Assim:

$$w_{k+1}(x,\lambda,\mu,\mu_{k+1}) \leq w_{k+1}(x,\lambda,\mu,0) \qquad \forall\ x \in X,\ \lambda \geq 0;\ \mu \geq 0\ e\ \mu_{k+1} \geq 0.$$

Logo existirão valores $\overline{\lambda}$ *e* $\overline{\mu}$ tais que: $L^*_{k+1} = w_{k+1}(\overline{x}, \overline{\lambda}, \overline{\mu}, 0)$ sendo $\overline{x}$ a solução associada a $\overline{\lambda}$, $\overline{\mu}$ *e* $\mu_{k+1} = 0$.

Portanto:

$$L^*_{k+1} = w_{k+1}(\overline{x}, \overline{\lambda}, \overline{\mu}, 0) = w_k(\overline{x}, \overline{\lambda}, \overline{\mu}) \leq L^*_k$$

Assim: $L^*_k \geq L^*_{k+1}$. Como $L^*_k \leq L^*_{k+1}$, segue então que $L^*_k = L^*_{k+1}$. •

Note portanto, que uma condição necessária para o incremento estrito do limite inferior é que exista pelo menos um $x \in X$ que viole a restrição $h_{k+1}x \leq f_{k+1}$!

Uma outra condição necessária pode ser apresentada: Considere inicialmente que os sistemas de desigualdades Bx≤d e Hx≤f sejam representados respectivamente por $\{b_i x \leq d_i$ p/ i=1,..,r$\}$ e $\{h_j x \leq d_j$ p/ j=1,..,k$\}$. Assim:

Teorema III.3: Se $h_{k+1}x \leq f_{k+1}$ for obtida através de uma combinação linear com coeficientes não-negativos das desigualdades $b_i x - d_i \leq 0$ para i=1,2,..,m e $h_j x - d_j \leq 0$ para j=1,...,k então $L^*_k = L^*_{k+1}$.

Demonstração:

Se $h_{k+1}x - f_{k+1} \leq 0$ puder ser obtida como uma combinação linear de coeficientes não-negativos teremos:

$$h_{k+1}x - f_{k+1} = \sum_{i=1}^{r} \lambda_i (b_i x - d_i) + \sum_{j=1}^{k} \mu_j (h_j x - f_j); \qquad \forall\, x \in X$$

Segue diretamente que $h_{k+1}x - f_{k+1} \leq 0 \quad \forall\, x \in X$. Logo, da proposição anterior teremos $L^*_k = L^*_{k+1}$. •

Assim, se a restrição $h_{k+1}x \leq f_{k+1}$ não for obtida através de uma combinação linear de coeficientes não-negativos das desigualdades $b_i x - d_i \leq 0$ para i=1,2,..,m e $h_j x - d_j \leq 0$ para j=1,...,k; teremos uma condição necessária para o incremento estrito do limite inferior.

Note entretanto que, mesmo que a desigualdade introduzida $h_{k+1}x \leq f_{k+1}$, satisfaça as duas condições necessárias acima (conseqüência dos teoremas III.2 e III.3 respectivamente) não teremos uma garantia do crescimento estrito do limite inferior!

Sempre que uma desigualdade violada pela solução do problema lagrangeano não puder ser obtida pela rotina de identificação devemos retornar ao *branch-and-bound*.

Vejamos inicialmente um procedimento (apresentado em Aboudi et al.[91]) que combina geração de restrições com relaxação lagrangeana. Como discutido anteriormente, um limite inferior para uma solução ótima z* é obtido após a resolução do seguinte problema dual lagrangeano:

$$w_k(x,\lambda,\mu) = min\ c^T x + \sum_{i=1}^{r} \lambda_i (b_i x - d_i) + \sum_{j=1}^{k} \mu_j (h_j x - f_j)$$

$$s.a \quad x \in X$$

onde $X = \left\{ x \in \{0,1\}^n \,/\, a_l x \leq b_l,\ l = 1,..,m \right\}$.

A idéia é adicionarmos restrições $h_j x \leq f_j$, $(j = 1,..,k)$ à medida que forem sendo violadas pela solução do problema lagrangeano, solução esta obtida pelo método subgradiente (Geoffrion[74]). Temos então o seguinte procedimento:

PROCEDIMENTO III.1: Relax. Lagrang. c/ Geração de Restrições (1^a versão)

Início

k:=0;

$\lambda_i := 0$; $(para\ \ i = 1,..,r)$; {inicializa multip. associados a $b_i x - d_i$ }

$\mu_0 = 0$;

Repita

- Calcula: $L_k^* = \max_{\substack{\lambda \geq 0 \\ \mu \geq 0}} w_k(x,\lambda,\mu)$ {método subgradiente}

- **Se** ($L_k^* < z^*$) **então**

- Sejam $\bar{\lambda}, \bar{\mu}$ variáveis duais ótimas associadas ao prob. dual lagrangeano e $\bar{x}$ a solução do problema relaxado correspondente;

- Obter restrição violada $h_{k+1}x \leq f_{k+1}$; {caso exista, deveremos ter $h_{k+1}\bar{x} > f_{k+1}$}

- **Se** (exitir restrição violada) **então**

$\mu_{k+1} := 0$;

k:=k+1;

senão

- retornar ao *branch-and-bound*;

fim;

Até que ("não existam" restrições violadas) ou (k > máximo de rest. dualizadas);

fim.

Figura III.2: Relaxação Lagrangeana c/ Geração de Restrições

Note que deverão ser definidas 3 condições de parada para o algoritmo acima. Na primeira situação, o procedimento termina quando a solução ótima do problema dual lagrangeano (obtida pelo método subgradiente) for superior ou igual a uma solução ótima do problema original. Neste caso, devemos retornar ao *branch-and-bound*. Como não conhecemos z*, podemos utilizar um limite superior z_{UB} para o valor da solução ótima do problema original.

Na segunda situação, finalizamos a geração de restrições caso "não existam" restrições violadas pela solução do problema lagrangeano. Observe que em muitos casos, o problema de identificação de restrições pode ser NP-completo. Logo, a utilização de uma heurística particular poderá não responder satisfatoriamente ao problema da existência de restrições violadas.

Finalmente, na terceira situação, como o número de restrições geradas pode ser extremamente grande (exponencial), estipulamos um número máximo de restrições a serem dualizadas.

Observe que, sempre que uma restrição $h_{k+1}x \leq f_{k+1}$ for violada por uma solução do problema dual lagrangeano, adicionamos esta desigualdade ao nosso conjunto de restrições violadas atendendo uma condição necessária para o crescimento estrito do limite inferior (teorema III.2).

Se a relaxação lagrangeana possui a propriedade de integralidade, o valor da solução do problema dual lagrangeano será igual à relaxação linear do problema original (Reeves[95]). Assim, o algoritmo da figura III.2 gera, a cada iteração, uma solução com mesmo valor que a relaxação linear do problema original. A partir desta solução, geramos uma nova desigualdade violada e repetimos o processo.

Embora o procedimento apresentado por Aboudi et al.[91] permita a geração de novas restrições violadas a cada solução do problema lagrangeano, as condições necessárias para o crescimento estrito do limite inferior não são implementadas de forma satisfatória. Note que algumas das restrições violadas em passos anteriores poderão ser verificadas por todos os pontos de X a medida que inserimos novas restrições!

Devemos portanto, desenvolver mecanismos que permitam uma atualização do conjunto de restrições dualizadas, identificando e eliminando as restrições que não contribuem mais para o incremento estrito do limite inferior.

Na seção seguinte apresentamos outra alternativa que combina, de maneira mais eficiente, o método subgradiente com a geração de restrições.

III.4 - Procedimento Alternativo:

Vamos agora considerar um procedimento alternativo que tem funcionado extremamente bem na prática, embora não exista ainda uma prova de convergência associada. Este procedimento procura adaptar os ingredientes básicos de um algoritmo de planos de cortes faciais para o contexto lagrangeano, aplicando o método subgradiente diretamente ao problema de programação linear inteira considerado. Isto é feito no entanto, considerando-se explicitamente, apenas um número "pequeno" das restrições candidatas à dualização. Para facilitar a notação, representamos por $P=\{ x \in \{0,1\}^n / Bx \leq d \text{ e } Hx \leq f\}$ o

conjunto das restrições candidatas à dualização, e por Q, o conjunto das restrições já dualizadas. Observe inicialmente que temos Q=∅.

O procedimento é iniciado, como feito usualmente, associando-se multiplicadores de valor 0 a cada restrição de P. Após a obtenção da solução do problema lagrangeano, identificamos um subconjunto de restrições pertencentes a P, violadas pela solução do problema relaxado corrente. Estas restrições são dualizadas e adicionadas ao conjunto Q. É evidente, ao final desta primeira iteração, que todos os multiplicadores de Lagrange associados às desigualdades não-dualizadas permaneçam nulos, à excessão daqueles associados às restrições dualizadas (conjunto Q), que se tornarão positivos após a atualização dos multiplicadores.

Na atualização dos multiplicadores, utilizamos uma alternativa onde unicamente as componentes do vetor subgradiente associadas às desigualdades de Q são consideradas. Evita-se assim, a tarefa impraticável de se calcular as componentes do subgradiente para cada uma das restrições candidatas à dualização. Vale aqui ressaltar que as demais componentes do subgradiente (não consideradas) afetariam unicamente o tamanho do passo e desta forma o valor dos novos multiplicadores associados às componentes de Q.

Como atualizar entretanto, o conjunto Q das restrições dualizadas? Observe que este conjunto pode se tornar bastante grande se não nos preocuparmos em "eliminar" de Q, aquelas desigualdades que não contribuem mais para o incremento estrito do limite inferior. Para responder a essa pergunta introduz-se o conceito de **desigualdades ativas** como sendo aquelas com multiplicadores não nulos ou que sejam violadas pela solução do problema lagrangeano. Desta forma, o conjunto Q das restrições dualizadas será composto unicamente por aquelas desigualdades que são ativas .

Nas fórmulas de atualização dos multiplicadores, considera-se apenas as restrições ativas. Vale aqui ressaltar mais uma vez que os multiplicadores associados às restrições não-ativas devem permanecer, de qualquer forma, iguais a zero, ao final de cada iteração. Observe ainda que um multiplicador associado a uma restrição ativa pode tornar-se igual a *zero*. Neste caso, tal desigualdade seria "eliminada" e se tornaria novamente ativa caso viesse a ser violada pela solução de um problema lagrangeano futuro. Como pode ser verificado por resultados computacionais (vide Fisher[94], Miller[95], Lucena[96] entre

outros), o número de desigualdades ativas tende a ser pequeno, tornando computacionalmente tratável a atualização de multiplicadores no método subgradiente.

Analogamente ao procedimento III.1, busca-se a inserção no conjunto ativo daquelas restrições que violam a solução $\bar{x}$ do problema relaxado. Neste caso, teremos $\sigma_i = b_i\bar{x} - d_i > 0$ ou $\varphi_j = h_j\bar{x} - f_j > 0$ para um subconjunto de índices i$\in$ {1,..,r} e j$\in$ {1,..,k} respectivamente. Obviamente se $\sigma_i \leq 0$ ou $\varphi_j \leq 0$, as restrições correspondentes não serão violadas por $\bar{x}$. Retiramos uma restrição σ_i $(ou\ \varphi_j)$ do conjunto ativo, caso o respectivo multiplicador λ_i $(ou\ \mu_j)$ seja nulo.

Resumindo todos os passos descritos acima, vejamos outra versão, computacionalmente mais interessante, combinando geração de restrições com relaxação lagrangena. Seja z* o valor da solução ótima associado ao problema original:

PROCEDIMENTO III.2: Relax. Lagrang. c/ Geração de Restrições (2^a versão);
Início

k:=0;

$\lambda = 0$; {inicializa multip. associados a Bx≤d}

$\mu = 0$; {inicializa multip. associados a Hx≤f}

Repita

- Calcula: $z_{LB} = w_k(x, \lambda, \mu)$; {retorna $\bar{x}$ - sol. do prob. lagrangeano}
- **Se** (z_{LB} < z*) **então**
  - Adiciona restrições violadas por $\bar{x}$ ao conjunto Q;
  - Calcula θ; {tamanho do passo}
  - Atualiza multiplicadores λ, μ associados às desig. de Q;
  - Elimina todas as restrições de Q c/ multiplicadores nulos;
  - Atualiza custos lagrangeanos;

senão

- retornar ao *branch-and-bound*;

fim; {fim do se-senão}

Até que (condição de parada);

fim.

Figura III.3: Relaxação Lagrangeana c/ Geração de Restrições

Para evitar que o número de restrições presentes no conjunto de ativo Q seja muito grande, podemos estipular um limite para o número máximo de restrições a serem dualizadas.

Em relação à condição de parada utilizada, poderíamos repetir o algoritmo acima até que o tamanho do passo $\pi \in (0,2]$, seja menor ou igual a ε (suficientemente pequeno). Este critério entretanto pode resultar em um elevado número de iterações. Outra possibilidade mais plausível, é determinarmos de antemão, o número máximo de iterações presentes no subgradiente.

Obviamente, algumas variações do procedimento III.2 podem ser apresentadas. Miller[95] propõe ainda alterações no tamanho do passo θ , em função da variação e da freqüência de atualizações dos limites inferiores. Volgenant e Jonker[82], apresentam uma direção de busca gerada através de uma combinação linear dos subgradientes atuais e os subgradientes obtidos em iterações anteriores. Atribuindo pesos distintos a cada um dos subgradientes gerados nas últimas iterações do subgradiente busca-se uma nova direção de maximização da função dual lagrangeana.

Capítulo IV

Roteamento de Veículos e a utilização de K-árvores Mínimas

IV.1 - Introdução:

Apresentamos neste capítulo, o trabalho desenvolvido por Fisher[94.b] que utiliza o método de relaxação lagrangeana com geração de restrições (capítulo anterior) aplicado ao problema clássico de roteamento de veículos. Como discutido anteriormente, uma frota de K veículos deve atender a um conjunto de n clientes cada um com uma demanda específica. O atendimento deve ser realizado de modo a não violar as restrições de capacidade de cada veículo. Além disso, cada cliente é visitado por apenas um veículo e nenhum veículo poderá atender mais de uma rota. O objetivo será atribuir a esses veículos uma seqüência de visitas aos clientes que minimize o "custo total" de transporte.

Em seu artigo, Fisher[94.b] utiliza uma relaxação lagrangeana onde cada subproblema considerado se resume na obtenção de uma K-árvore mínima (definida adiante) com 2K arestas incidentes ao depósito (vértice 0). Este subproblema é resolvido em $O(n^3)$ passos e gera limites inferiores melhores que aqueles obtidos por Christofides et

al.[81.a], que utilizam uma abordagem parecida para a obtenção de limites inferiores. Neste artigo, Christofides et al.[81.a] constróem uma 0-árvore com um vértice fixo (depósito) de grau k (onde $K \leq k \leq 2K$).

Na seção seguinte apresentamos a formulação e a relaxação lagrangeana utilizadas por Fisher [94.b] na solução do problema de roteamento. Na seção IV.3, estudamos como obter uma K-árvore mínima com um número fixo de arestas incidentes ao depósito. Na seção IV.4, apresentamos a abordagem utilizada por Fisher na identificação de restrições violadas pela solução do problema lagrangeano. Finalmente, na seção IV.5, discutimos algumas estratégias para a determinação de limites superiores (heurísticas lagrangeanas) sugeridas por Fisher[94.b].

IV.2 - Formulação utilizando K-árvores Mínimas:

Suponha que cada um dos K veículos a serem utilizados tenha capacidade **b** e cada cliente demanda d_i (onde i=1,..,n). Consideramos ainda $d_0 = 0$, como sendo a demanda associada ao depósito. Os custos entre as cidades **i** e **j** serão simétricos, ou seja, $c_{ij} = c_{ji}$.

Seja N={1,..,n} o conjunto de clientes e $N_0 = N \cup \{0\}$, o conjunto N com a inclusão do depósito. Utilizaremos uma variável binária x_{ij} para indicar se a aresta (i,j) foi ou não utilizada na solução do problema. Se $E = \{(i,j) / i,j \in N_0 \ e \ i < j\}$ é o conjunto de arestas associados a nosso problema, diremos que um subgrafo T_K de um grafo $G(N_0,E)$ define uma K-árvore de G, se e somente se T_K for conexo, e tiver n+1 vértices e n+K arestas.

Como as arestas (i,j) de E são não direcionadas, para simplificar a notação, assumiremos que o par de índices *ij* em x_{ij} estão associados a um par não ordenado. Assim, x_{ij} e x_{ji} denotam a mesma variável. Teremos portanto, n(n+1)/2 variáveis x_{ij} associadas a pares não-ordenados de $N_0 x N_0$. Definimos ainda:

$$x = (x_{01}, x_{02}, .., x_{0n}, x_{12}, .., x_{n-1,n}) \text{ e } X = \left\{ x / x \in \{0,1\}^{|E|} \text{ define uma K - arvore e } \sum_{i=1}^{n} x_{0i} = 2K \right\}$$

Note x $\in$X é um vetor de incidência (ou característico) associado à K-árvore com depósito de grau 2K.

Seja d_i, a demanda associada ao cliente i$\in$N. Para cada S $\subseteq$ N, define-se $\overline{S} = N_0 \setminus S$, $d(S) = \sum_{i \in S} d_i$ e $r(S) = \left\lceil d(S)/b \right\rceil$. Como veremos adiante, r(S) representa um limite inferior para o número de veículos necessários para atender aos clientes do conjunto S. Para $S \subseteq N_0$, seja E(S) o conjunto de todos os pares não-ordenados *ij* onde i, j$\in$S e i$\neq$j. Assim, temos a seguinte formulação para o Problema de Roteamento de Veículos com restrições de capacidade:

$$z^* = min \sum_{ij \in E(N_0)} c_{ij} x_{ij} \qquad (1)$$

$$s.a. \begin{cases} \sum_{\substack{j \in N_0 \\ i \neq j}} x_{ij} = 2; & \forall\, i \in N \qquad (2) \\ \sum_{i \in S} \sum_{j \in \overline{S}} x_{ij} \geq 2r(S); & \forall\, S \subseteq N \;\; com \;\; |S| \geq 2 \qquad (3) \end{cases}$$

$$x \in X \qquad (4)$$

As desigualdades (3) são análogas às restrições de eliminação de sub-rotas utilizadas na formulação do problema do caixeiro viajante proposta por Dantzig, Fulkerson e Johnson[54]. Nobert[82] e Laporte et al.[85] mostraram que as restrições de capacidade (3) podem ser representadas equivalentemente por:

$$\sum_{ij \in E(S)} x_{ij} \leq |S| - r(S); \qquad \forall\, S \subseteq N \;\; com \;\; |S| \geq 2 \qquad (3')$$

Para ilustrar a condição (3') vejamos o exemplo da figura IV.1 onde consideramos 3 veículos de capacidade 15 e duas partições S e S'. Ao lado de cada nó estão representadas as demandas de cada cliente.

Na partição S temos $r(S)=\lceil 12/15 \rceil = 1$. Segue então de (3') que $\sum_{ij \in E(S)} x_{ij} \leq 3-1=2.$

Logo, a restrição se verifica para esta partição já que temos apenas 2 arestas com ambas as extremidades em S. Entretanto, para a partição S' temos $r(S')=\lceil 16/15 \rceil = 2$, assim $\sum_{ij \in E(S')} x_{ij} \leq 3-2=1$, e a restrição não é satisfeita. Observamos portanto que a restrição associada à partição S' é violada. Assim, temos uma solução não viável para o problema de roteamento de veículos com restrições de capacidade.

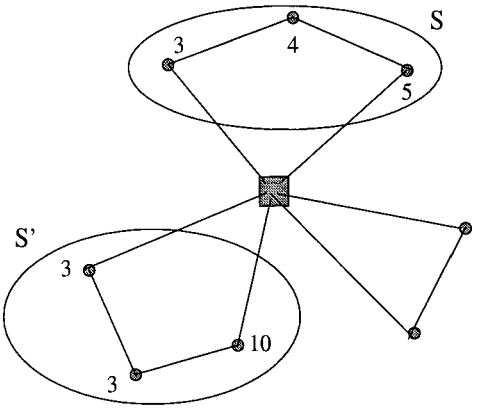


Figura IV.1:Solução que não satisfaz as restrições de capacidade

A viabilidade é garantida se a solução acima satisfizer a seguinte condição apresentada em Fisher[94.b]:

Teorema IV.2.1:

Uma solução $x \in X$ é viável para o problema de roteamento de veículos se e somente se satisfizer (2) e (3).

Prova:

($\leftarrow$) Note que para x pertencente a X e que satisfaça a condição (2) devemos ter K ciclos que começam e terminam no depósito (não temos outra possibilidade). Seja S_k um destes ciclos onde $k \in \{1,..,K\}$. Se $d(S_k) > b$ então teremos $r(S_k)=\left\lceil \frac{d(S_k)}{b} \right\rceil > 1$. Segue-se

então de (3) que $\sum_{i\in S_k}\sum_{j\in\bar{S}_k} x_{ij} \geq 4$. Isto é um absurdo já que $\sum_{i\in S_k}\sum_{j\in\bar{S}_k} x_{ij} = 2,\ \forall\, k \in \{1,..,K\}$. Devemos ter portanto, $d(S_k) \leq b,\ \forall\, k \in \{1,..,K\}$ (solução viável).

($\rightarrow$) Se $x \in X$ é viável para o problema de roteamento, devemos ter K ciclos que começam e terminam no depósito. Segue portanto que a condição (2) é satisfeita. Seja r(S) o número mínimo de veículos necessários para atender os clientes de S. Desta forma, como cada veículo deve entrar e sair de S, teremos pelo menos 2r(S) arestas entre os conjuntos S e $\bar{S}$. Logo (3) se verifica. •

Em seu trabalho, Fisher [94.b] apresenta inicialmente resultados para rotas com no mínimo 2 clientes, ou seja, não são permitidas soluções viáveis com a presença de rotas simples. Em um problema de roteamento de veículos, rotas simples são inviáveis quando:

$$\sum_{\substack{i=1\\ i\neq j}}^{n} d_i - d_j > (K-1)b, \qquad \forall\ j \in \{1,..,n\}$$

Note que a soma das demandas de todos os outros clientes (diferentes de **j**) não pode ser atendida pelos (K-1) veículos restantes. Sendo assim, não podemos ter soluções viáveis para o problema de roteamento com rotas simples e utilizando K veículos. Como discutido em Fisher [94.b], sua abordagem pode ser facilmente modificada de maneira a permitir a presença de rotas simples.

Associando-se multiplicadores $u_i \in \Re\ (p/\ i \in N)$ às restrições (2) e $v_S \in \Re^+$ $(\forall\, S \subseteq N\ c/\ |S| \geq 2)$ às restrições (3) e dualizando-as de forma lagrangeana temos:

$$\sum_{ij\in E(N_0)} c_{ij}x_{ij} + \sum_{i=1}^{n} u_i\left(2 - \sum_{j\in N_0\ e\ i\neq j} x_{ij}\right) + \sum_{S\subseteq N} v_S\left(2r(S) - \sum_{i\in S}\sum_{j\in\bar{S}} x_{ij}\right) \quad \text{(i)}$$

ou ainda:

$$\sum_{ij\in E(N_0)} c_{ij}x_{ij} - \sum_{i=1}^{n} u_i \sum_{j\in N_0 \ e \ i\neq j} x_{ij} - \sum_{S\subseteq N} v_S \sum_{i\in S}\sum_{j\in \overline{S}} x_{ij} + 2\sum_{i=1}^{n} u_i + 2\sum_{S\subseteq N} v_S r(S) \qquad \text{(ii)}$$

Para simplificar essa expressão vejamos o seguinte exemplo onde temos duas partições S_0 *e* S_1 como representadas na figura IV.2.

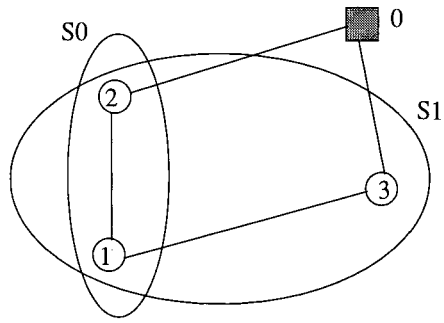


Figura IV.2: Obtençao da função lagrangeana

Desenvolvendo as parcelas não nulas de nossa expressão e lembrando que $x_{ji} = x_{ij}$ (sempre que i>j) temos:

$$c_{02}x_{02} + c_{03}x_{03} + c_{12}x_{12} + c_{13}x_{13} - u_1(x_{13} + x_{12}) - u_2(x_{02} + x_{12}) - u_3(x_{03} + x_{13}) -$$
$$-v_{S_0}(x_{13} + x_{02}) - v_{S_1}(x_{03} + x_{02}) + 2(u_1 + u_2 + u_3) + 2v_{S_0}r(S_0) + 2v_{S_1}r(S_1)$$

Note que, como não temos rotas simples não temos arestas duplicadas na função objetivo. Colocando as variáveis em evidência obtemos:

$$(c_{02} - u_0 - u_2 - v_{S_0} - v_{S_1})x_{02} + (c_{03} - u_0 - u_3 - v_{S_1})x_{03} + (c_{12} - u_1 - u_2)x_{12} + (c_{13} - u_2 - u_3 - v_{S_0})x_{13} +$$
$$+2(u_1 + u_2 + u_3) + 2v_{S_0}r(S_0) + 2v_{S_1}r(S_1)$$

Generalizando este procedimento para a expressão (ii) chegamos à seguinte relaxação lagrangeana:

$$z_D(u,v) = min \sum_{ij \in E(N_0)} \bar{c}_{ij} x_{ij} + 2\sum_{i=1}^{n} u_i + 2\sum_{S \subseteq N} v_S r(S) \qquad (5)$$

$$s.a. \quad x \in X$$

onde:

$$u_0 = 0 \quad e \quad \bar{c}_{ij} = c_{ij} - u_i - u_j - \sum_{\substack{i \in S, j \in \bar{S} \\ ou \\ i \in \bar{S}, j \in S}} v_S$$

Temos que $z_D(u,v) \leq z^*$ onde z* é o valor da solução ótima do problema original (Geoffrion [74]). Utilizamos o método subgradiente (Held, Crowder e Wolfe[74]) para "resolver" o seguinte problema dual lagrangeano:

$$\begin{aligned} &max\ z_D(u,v) \\ &s.a.\ u \in \Re^n \\ &\quad v \geq 0 \end{aligned} \qquad \text{(P.D)}$$

obtendo assim o melhor limite inferior possível neste caso.

Como temos um número exponencial de restrições em (3), não é viável a dualização de todas essas restrições explicitamente. Sendo assim, devemos desenvolver mecanismos que gerem dinamicamente essas restrições à medida que forem sendo violadas pela solução do problema lagrangeano. Todas as outras restrições serão ignoradas fazendo-se $v_S = 0$. Note que uma restrição ignorada só irá efetivamente contribuir para a função de custo lagrangeana quando seu multiplicador associado for não nulo. No método subgradiente, todos os multiplicadores são inicialmente tomados iguais a *zero*. Desta forma, uma restrição só é efetivamente considerada quando for violada pela solução do problema lagrangeano, já que isto levaria a um multiplicador não-nulo para a restrição.

Como discutido anteriormente, seria impraticável a geração explícita de todas as restrições apresentadas em (3) ou (3'). Fisher[94.b] trabalha com uma quantidade de restrições bem inferior às 2^n restrições geradas por todas as possíveis partições do conjunto

N. Segundo Fisher[94.b], testes computacionais indicam que a utilização das restrições (3) geram limites inferiores mais interessantes quando comparados às restrições (3').

Fisher[94.b] propõe ainda a substituição de (3) ou (3') por um novo conjunto de restrições que levem a uma formulação mais forte (no sentido de uma relaxação linear) para o problema.

Note que o produto br(S) define a capacidade mínima exigida para todos os veículos que circulam pelos vértices de S. Na ilustração abaixo apresentamos um exemplo bem simples onde temos uma partição S com 2 nós. Considerando r(S)=1 (segue-se de (3) que $\sum_{i\in S}\sum_{j\in\bar{S}} x_{ij} \geq 2$, e a restrição se verifica). Observe que temos 4 arestas com um de seus extremos em S e outro em $\bar{S}$.

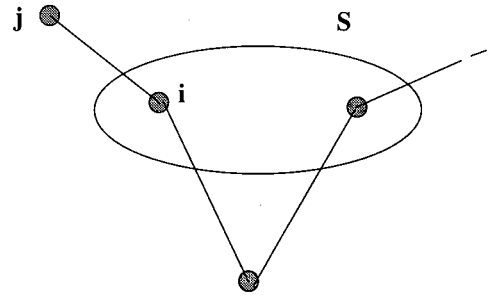


Figura IV.3: Análise das restrições (3)

Note que, se $x_{ij} = 1$ no lado esquerdo de (3) (para algum cliente $j \in \bar{S}$), teremos **j** sendo atendido por algum veículo de S. Dessa forma, a capacidade mínima br(S) destinada aos veículos que percorrem S, deverá ser suficiente para atender também a demanda de clientes de $\bar{S}$ tais que $x_{ij} = 1$ para algum **i** pertencente a S. Logo, a capacidade mínima de todos os veículos que entram e saem de S deverá ser igual a d(S) mais a demanda de todos os clientes **j** de $\bar{S}$ com $x_{ij} = 1$. Teremos portanto a seguinte modificação:

$$\sum_{i\in S}\sum_{j\in\bar{S}} x_{ij} \geq 2\left\lceil \frac{d(S)+\sum_{\substack{i\in S\\ j\in\bar{S}}} d_j x_{ij}}{b} \right\rceil \qquad (6)$$

A desigualdade (6) não pode ser utilizada diretamente já que seu lado direito é uma função não linear de x. Isto não impede entretanto, que a utilizemos como ferramenta para geração de outras restrições.

Seja S⊆N, um subconjunto qualquer de N. Fazendo $S' = \left\{ j \in \overline{S} / \; j \geq 1 \; e \; d_j > br(S) - d(S) \right\}$ definimos:

$$e_j = \begin{cases} 0, & j \in S \\ 0, & j \in S' \; e \; |S'| \leq 2 \\ \dfrac{r(S)}{r(S)+1}, & j \in S' \; e \; |S'| > 2 \\ 1, & j \in \overline{S} - S' \end{cases} \qquad (7)$$

Note que S' representa os clientes de $\overline{S}$ que não podem ser atendidos pela capacidade "ociosa" após o atendimento aos clientes de S. Ou seja, a demanda de **j** é superior à capacidade restante nos veículos que percorrem S.

A idéia é reduzir o máximo possível alguns dos coeficientes de (3) sem destruir a viabilidade do problema original, ou seja, sem violar as restrições de capacidade. Desta forma estaremos obtendo um novo conjunto de restrições (apresentadas abaixo) que dominarão as restrições apresentadas em (3). Assim:

$$\sum_{j=0}^{n} e_j \sum_{i \in S} x_{ij} \geq 2r(S), \quad \forall \; S \subseteq N \; para \; |S| \geq 2 \qquad (8)$$

Note em (7) que se $|S'| \leq 2$ teremos (8) apenas com coeficientes 0´s e 1´s. Entretanto, se $|S'| > 2$ aqueles coeficientes associados a S' serão alterados para r(S)/(r(S)+1) (menor que 1). Obtemos portanto uma sensível melhora em relação às restrições (3).

O seguinte teorema (apresentado em Fisher [94.b]) garante a substituição das restrições (3) pelas restrições (8) apresentadas acima:

Teorema IV.2: Uma solução $x \in X$ será viável para o problema de roteamento de veículos se e somente se satisfizer as restrições (2) e (8).

Prova:

($\leftarrow$) Este caso segue diretamente já que qualquer solução $x \in X$ satisfazendo (2) e (8) também irá satisfazer (2) e (3). Note que (3) e (8) são quase idênticos, excetuando-se apenas aqueles coeficientes que são menores em (8). Logo, do teorema IV.1, temos que $x \in X$ será viável para o problema de roteamento.

($\rightarrow$) Mostraremos agora que se $x \in X$ satisfaz (2) e (3) então x também satisfaz a (8). Consideremos inicialmente $|S'| = 0$. Neste caso, teremos de (7) que $e_j = 1, \forall\ j \in \bar{S}$. Assim, (3) e (8) serão idênticos.

Vejamos agora a situação onde temos $|S'| \geq 1$ e $x_{kj^*} = 1$ para algum $j^* \in S'$ e $k \in S$. Dois casos deverão ser analisados para mostrar que (8) realmente se verifica:

Caso 1: ($|S'| \leq 2$)

Seja $S'' = \{j \in S' / \sum_{i \in S} x_{ij} > 0\}$. Se $|S''| = 1$, então o cliente j^* de S' e um ou mais clientes de S estarão em uma mesma rota **r** (isto se deve ao fato de termos $x \in X$ e que satisfaz (2)). Como $d_{j^*} > br(S) - d(S)$ ou ainda $d_{j^*} + d(S) > br(S)$, serão utilizados pelo menos r(S)+1 veículos para atender os clientes de $S \cup \{j^*\}$. Dessa forma, pelo menos r(S) veículos serão necessários para atender os clientes de S que não estão na rota **r**. Assim, teremos pelo menos 2r(S) arestas incidentes em S de $\bar{S} - S'$. Segue portanto, que toda solução que satisfaz (3) com $|S''| = 1$ também irá satisfazer a (8). Assim, poderemos anular e_{j^*} (coeficiente de $x_{ij^*}\ p/\ j^* \in S'$), sem perder a viabilidade de nosso problema.

Vejamos agora a situação onde temos $|S''| = 2$, ou seja, dois clientes de S' estão ligados a clientes de S. Se os dois clientes de S" estiverem em uma mesma rota, analogamente à situação anterior teremos no mínimo r(S) veículos para atender os demais clientes de S. Logo, teremos pelo menos 2r(S) arestas incidentes a S de $\bar{S} - S'$ e (8) se verifica.

Se os dois clientes de S" estiverem em duas rotas distintas, serão necessários pelo menos r(S)-1 veículos para atender os demais clientes de S (que não estão nestas duas

rotas). Segue que, no mínimo 2r(S)-2 arestas serão incidentes a S. Note entretanto que, para cada uma das duas rotas temos uma aresta de $\bar{S} - S'$ incidente a S. Assim, teremos no mínimo 2r(S) arestas incidentes a S de $\bar{S} - S'$. Concluímos então que mesmo fazendo $e_j = 0$, toda solução que satisfaz a (3) com $|S''| = 2$ também irá satisfazer a (8).

Caso 2: ($|S''| > 2$)

Como $d_{j*} > br(S) - d(S)$ e $x_{kj*} = 1$ para k∈S, j∈S', serão necessários pelo menos r(S)+1 veículos para atender os clientes de S. Logo, teremos no mínimo 2r(S)+2 arestas incidentes a S. Segue: $\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geq 2r(S) + 2$. Multiplicando ambos os lados da desigualdade por r(S)/(r(S)+1) teremos: $\sum_{i \in S} \frac{r(S)}{r(S)+1} \sum_{j \in \bar{S}} x_{ij} \geq 2r(S)$.

Portanto, toda solução x que satisfaz a desigualdade acima também satisfaz (8) •

Apesar da melhoria introduzida acima, o conjunto de restrições (8) não define facetas. Para ilustrar essa situação vejamos o seguinte exemplo onde temos S={1,2,3}, $d_1 = d_2 = d_3 = 3$, $d_j = 1$ para j≥4 e b=5.

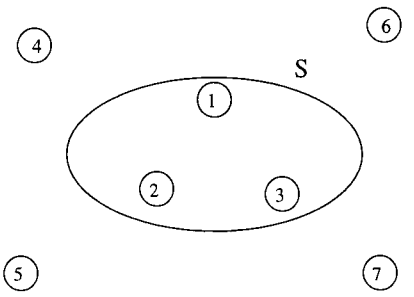


Figura IV.4: Análise das restrições (3) e (8)

Note que, como S'=∅ temos (3) e (8) duas desigualdades idênticas. Assim, $\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geq 4$, já que $\lceil 9/5 \rceil = 2$. É fácil ver que $\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geq 6$ é válida e domina (8). Como discutido no capítulo II, uma faceta não pode ser dominada por nenhuma outra restrição válida para o problema .

Antes de tratarmos propriamente da solução do problema dual lagrangeano vejamos como obter uma K-árvore mínima com 2K arestas incidentes ao depósito. Assim, definido os custos $\bar{c}_{ij}$ para cada par i,j $\in N_0$, trataremos primeiramente da relaxação lagrangeana (apresentada em (5)).

IV.3 - Obtenção de uma K-árvore Mínima c/ um número fixo de arestas incidentes ao depósito:

Seja dado um grafo G=(N_0,E) com um conjunto de n+1 vértices onde N_0={0,1,..,n}, um conjunto E de arestas e um inteiro K≥0. Como discutido na seção anterior, chamaremos T_K (subgrafo de G) de K-árvore se T_K for um grafo conexo com n+1 vértices e n+K arestas. Note que se K=0, teremos uma árvore geradora de G (0-árvore). Mostraremos que para se obter uma K-árvore de menor custo, bastará calcular uma árvore geradora mínima T adicionando-se em seguida as K menores arestas de E-T.

Para resolvermos a relaxação lagrangeana associada ao problema de roteamento de veículos (como descrito anteriormente), precisamos calcular uma K-árvore mínima com 2K arestas incidentes ao depósito (vértice 0). Chamaremos este problema de P(2K). Como veremos adiante, este problema pode ser resolvido em tempo polinomial ($O(n^3)$ iterações). Se impomos uma restrição adicional exigindo que cada um dos vértices i de N_0 tenham um grau no máximo g_i, estipulado a priori, teremos o problema da árvore geradora mínima com restrições de grau. Como demonstrado em Garey e Johnson[79], trata-se de um problema NP-Árduo! Em nosso caso entretanto, apenas o depósito tem grau fixo.

Fisher[94.a] faz uma extensão dos trabalhos desenvolvidos por Glover, Klingman [74 e 84], onde buscavam uma árvore geradora mínima (0-árvore) com um número fixo de arestas em um dado vértice.

Se a K-árvore mínima T_K obtida inicialmente não possuir grau 2K no depósito (número de arestas incidentes), fazemos trocas sucessivas entre as arestas de T_K, obtendo uma nova árvore T_K' incrementando ou decrementando o número de arestas incidentes ao depósito conforme o caso. Supondo por exemplo, que o grau do depósito seja superior a

2K, adicionamos uma aresta não-incidente ao depósito e não pertencente a T_K e retiramos uma das arestas de T_K incidentes ao depósito. Repetimos o processo até que se tenha o grau desejado. Analogamente, se o grau do depósito for inferior a 2K adicionamos uma aresta não pertencente a T_K e incidente ao depósito e retiramos uma aresta de T_K não incidente ao depósito. Da mesma forma, o processo é repetido até que se tenha o grau desejado.

Resumindo, teremos o seguinte procedimento. O valor GRAU(0) abaixo representa o número de arestas incidentes ao depósito.

PROCEDIMENTO: K-árvore mínima com depósito de grau 2K;
Início
- Calcula Árvore Geradora Mínima {árvore T}
- Adiciona as K menores arestas de E-T; {obtém T_K inicial}
- Calcula GRAU(0);
- **Se** GRAU(0) > 2K **então**
  - decrementa GRAU(0) até que GRAU(0)=2K

  senão
  - **se** GRAU(0) ≠ 2K **então**
    - incrementa GRAU(0) até que GRAU(0)=2K;

fim; {se}
- Calcula custo de TK;

fim.

Figura IV.5: K-árvore mínima c/ 2K arestas incidentes ao depósito

Para mostrar a consistência do procedimento acima vejamos alguns resultados demonstrados em Fisher [94.a] que estabelecem as condições de otimalidade e resultados que permitirão incrementar ou decrementar o grau do depósito sempre partindo de uma K-árvore mínima T_K para outra K-árvore mínima T_K' fazendo apenas trocas entre as arestas do grafo.

Vejamos primeiramente a seguinte definição e lema apresentados em Glover e Klingman [84]:

Definição IV.1 - Dada uma árvore geradora T e duas arestas $e \in T$ e $e' \notin T$. O par de arestas (e, e') define uma *troca admissível* em T se e somente se T-e$\cup${e'} for uma árvore geradora. •

Lema IV.1: Dados duas árvores geradoras distintas T e T', existirá uma função biunívoca h: T-T'$\rightarrow$ T'-T gerando pares (p,q) de trocas admissíveis em T (onde p $\in$ T-T' e q $\in$ T'-T).
Prova: Glover e Klingman[84] •

A definição de trocas admissíveis para K-árvores pode ser generalizada da seguinte forma:

Definição IV.2- Considere uma K-árvore T_K, uma aresta $e \in T_K$ e $e' \notin T_K$. O par (e, e') definirá uma *troca admissível* em T_K se $T_K - e \cup \{e'\}$ for uma K-árvore. •

O lema IV.2 a seguir é uma generalização do lema IV.1 para K-árvores:

Lema IV.2: Dados duas K-árvores distintas T_K e T_K', existirá uma função biunívoca h: $T_K - T_K' \rightarrow T_K' - T_K$ gerando pares (p,q) de trocas admissíveis em T_K (onde p $\in T_K - T_K'$ e q $\in T_K' - T_K$).

Prova:

Seja $\overline{T}$ um conjunto maximal em $T_K \cap T_K'$ que não contenha ciclos. Aumentamos $\overline{T}$ quando necessário com arestas de T_K para obter uma árvore geradora T$\subseteq T_K$ e com arestas de T_K' para obter uma árvore T'$\subseteq T_K$'. Do lema IV.1, existe uma função biunívoca h: T-T'$\rightarrow$ T'-T de maneira que cada par de arestas (de T-T' e T'-T respectivamente) define uma troca admissível em T. Esta função também define um emparelhamento de arestas de $T_K - T_K'$ com arestas de $T_K' - T_K$, já que T-T'$\subseteq T_K - T_K'$ e T'-T $\subseteq T_K' - T_K$, caso contrário, teríamos um elemento de $T_K \cap T_K'$ que poderia ser adicionado a $\overline{T}$ sem a criação de um ciclo, contradizendo portanto o fato de que $\overline{T}$ é maximal em $T_K \cap T_K'$.

Seja S=T_K-T e S'=T_K'-T'. Qualquer função biunívoca de arestas de S-S' para S'-S define trocas admissíveis, completando a prova. •

O teorema seguinte (apresentado em Fisher[94.b]) garante a existência de uma K-árvore de custo mínimo:

Teorema IV.3: (K-árvore mínima inicial)

Seja T $\subseteq$ E uma árvore geradora mínima e E'=E - T o complemento das arestas de E em relação a T. Seja S $\subseteq$ E' o conjunto das K menores arestas pertencentes a E'. Então $T_K = T \cup S$ será uma K-árvore mínima de G(N_0,E).

Prova:

Suponha por contradição que exista uma K-árvore T_K' com custo c(T_K')<c(T_K). Do lema IV.2, existe uma função biunívoca h: $T_K - T_K' \rightarrow T_K' - T_K$ de maneira que cada par define trocas admissíveis em T_K. Como temos c(T_K') < c(T_K), existirá pelo menos um par (e,e') (sendo $e \in T_K - T_K'$ e $e' \in T_K' - T_K$) com c(e')<c(e). Note que não podemos ter e$\in$S, pois c(e') < c(e) contradiz a definição de S. Se $e \in$T e T-$\{e\}\cup\{e'\}$ é uma árvore, chegaríamos a uma contradição pois T é uma árvore geradora mínima.

A única possibilidade então é considerarmos $e \in$T e T- $\{e\}\cup\{e'\}$ não definindo uma árvore. Assim, como T- $\{e\}\cup\{e'\}$ é conexo, deverá existir um elemento $a \in$S tal que T- $\{e\}\cup\{a\}$ é uma árvore. Logo, teremos c(a)$\geq$c(e)>c(e'), onde a primeira desigualdade se verifica já que T é uma árvore geradora de custo mínimo. Note então que poderíamos reduzir o custo substituindo e' por a em S, contradizendo a definição de S (conjunto das K menores arestas de E'). •

O lema IV.3 e os teoremas IV.4 - IV.6 seguintes generalizam para K-árvores alguns dos resultados obtidos por Glover e Klingman[84] para árvores geradoras (0-árvores). Estes resultados servirão de base para construção dos procedimentos responsáveis pelo incremento ou decremento do grau do depósito quando necessário.

Antes de demonstrar o próximo lema apresentamos a seguinte notação:

Notação: Considere uma K-árvore T_K e uma aresta $e' \notin T_K$. Representaremos por $T_K(e')$ o sub-conjunto de arestas de T_K que definem troca admissíveis com e'. •

Lema IV.3: Consideremos e, f $\in T_K$; e', f ' $\notin T_K$ arestas adjacentes ou não ao depósito. Se pelo menos um dos pares (e,e') ou (f, f ') não dá uma troca admissível em T_K então (e,f') e (f, e') darão (cada um) uma troca admissível em T_K se e somente se $T_K' = T_K - e - f \cup \{e', f'\}$ é uma K-árvore.

Prova:

($\leftarrow$) Se T_K' é uma K-árvore, do lema IV.2 teremos uma função biunívoca relacionando as arestas e, f com e' , f' de maneira que ambos os pares, definem trocas admissíveis. Existirá somente dois emparelhamentos possíveis. Caso (e,e') e (f,f') não definam trocas admissíveis em T_K, as trocas (e,f') e (f,e') serão ambas admissíveis em T_K.

($\rightarrow$) Agora mostraremos que T_K' é uma K-árvore se (e,f') e (f,e') são ambas admissíveis em T_K. Sem perda de generalidade assumiremos que (e,e') não define uma troca admissível em T_K, assim $e \notin T_K(e')$. Seja e=(i,j). Como (f,e') define uma troca admissível em T_K, T_K'$\cup$\{e\} é conexa. Assim, para mostrar que T_K' é uma K-árvore é suficiente mostrar que T_K' contém um caminho de i para j.

Note que $e \in T_K(f')$, implica na existência de um caminho em $T_K \cup \{f'\}$ de i para j. Se este caminho não contém f, ele será também um caminho em T_K' e a prova está completa. Se este caminho contém f, considere f=(p,q) e, sem perda de generalidade, assuma que o caminho em $T_K \cup \{f'\}$ de i até j passando por (p,q) inclui um caminho P_1 de i até p e um caminho P_2 de q até j. Como $f \in T_K(e')$ e $e \notin T_K(e')$, segue que $T_K \cup \{e'\}$ contém um caminho P_3 de p até q que não contém e (se P_3 contém e, o conjunto de arestas $P_3 \cup \{f\}$-{e} define um caminho em $T_K \cup \{e'\}$ de i até j e assim teremos $e \in T_K(e')$). Portanto, P_1, P_2 e P_3 definem o caminho em T_K de i até j. •

O resultado seguinte, demonstrado em Fisher[94.a] garante a obtenção de uma K-árvore mínima com um número fixo de arestas incidentes ao depósito.

Seja P(q) o problema de encontrar uma K-árvore com grau **q** no depósito. Considere (e, f) uma troca admissível (onde $e \in T_k$ e $f \notin T_k$). Esta troca será *satisfatória* se c(e) > c(f), onde c(e) e c(f) são os custos das arestas **e** e **f** respectivamente.

Vejamos agora as condições de otimalidade apresentadas em Fisher[94.a]. Nos teoremas que se seguem o superescrito 0 indica uma aresta incidente ao nó 0.

Teorema IV.4: (Condições de Otimalidade)

Uma K-árvore T_K com grau 2K no depósito será ótima para o problema P(2K) se e somente se satisfazer as seguintes condições de otimalidade:

i) Sejam e e f duas arestas não incidentes ao depósito. Não existem trocas admissíveis satisfatórias (e, f) (onde $e \in T_K$ e $f \notin T_K$).

ii) Não haverá trocas admissíveis satisfatórias (e^0, f^0) onde $e^0 \in T_k$ e $f^0 \notin T_k$ sendo e^0 e f^0 incidentes ao depósito.

iii) Não haverá duas trocas admissíveis satisfatórias (e_1^0, f_1) e (e_2, f_2^0) com $e_1^0, e_2 \in T_k$, $f_1, f_2^0 \notin T_k$, sendo e_1^0, f_2^0 incidentes no depósito e f_1, e_2 não incidentes ao depósito.

Prova:

($\rightarrow$) Consideremos inicialmente T_K, uma K-árvore ótima para o problema $P(2K)$. Assim $c(T_K) < c(T_K')$, qualquer que seja a K-árvore T_K' com 2K arestas incidentes ao depósito. É fácil ver neste caso que, se (i), (ii) ou (iii) forem violadas, obtemos uma nova K-árvore T_K' com custo inferior a T_K (K-árvore ótima). Logo (i), (ii) e (iii) se verificam.

($\leftarrow$) Suponha agora que T_K, satisfazendo (i), (ii) e (iii) não seja ótima. Teremos neste caso, a existência de uma K-árvore $\overline{T}_K$ com custo c($\overline{T}_K$) inferior a c(T_K). Do lema IV.2, temos uma função biunívoca h: T_K-$\overline{T}_K \rightarrow \overline{T}_K$-$T_K$, de maneira que cada um dos pares obtidos gerem trocas admissíveis em T_K. Essa seqüência de trocas pode ser dividida em 2 tipos distintos: (a) trocas que não atualizam o grau do depósito ou (b), trocas que atualizam

o grau do depósito (nó 0). É fácil ver que $\overline{T}_K$ não pode ser obtida apenas através de trocas do tipo (a), pois teríamos (i) ou (ii) violadas chegando portanto a um absurdo. Logo, devemos ter também trocas do tipo (b). Considere (sem perda de generalidade) um conjunto de **p** trocas que incrementam o grau do depósito gerando uma nova K-árvore T_K' com custo $c(T_K')$ superior a $c(T_K)$. Suponha ainda que $\overline{T}_K$ seja gerada através de **p** trocas que decrementam o grau do nó 0 (a partir de T_K'), e de maneira que $c(\overline{T}_K) < c(T_K)$. Deveremos ter neste caso, pelo menos um par de trocas que incrementa e decrementa respectivamente o grau do depósito em T_K gerando uma nova K-árvore T_K'' e com custo $c(T_K'')< c(T_K)$, contrariando portanto a condição (iii). •

Note que nas 3 condições de otimalidade apresentadas acima, foram expressas todas as possíveis trocas entre as arestas de T_K sempre mantendo 2K arestas incidentes ao depósito. Em nenhuma delas poderemos ter uma troca admissível satisfatória, ou seja, não poderemos ter uma diminuição no custo da K-árvore.

Para ilustrar a condição (iii), vejamos o exemplo da figura IV.6, onde temos uma 2-árvore com 4 arestas incidentes ao depósito. As arestas f_1 e f_2^0 não pertencem a T_K.

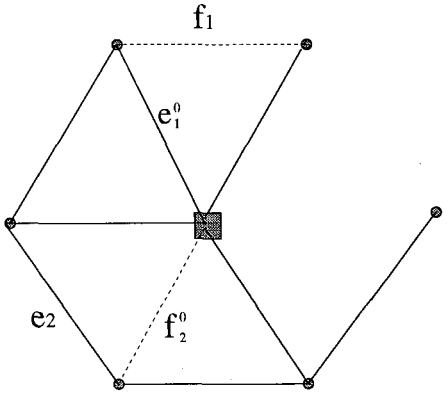


Figura IV.6: Condição (iii)

Para que as trocas admissíveis não sejam satisfatórias devemos ter $c(e_1^0) + c(e_2) \leq c(f_1) + c(f_2^0)$. Se as trocas forem efetuadas teremos uma nova 2-árvore de grau 4 no depósito e com custo superior à anterior.

Considere uma K-árvore mínima T_K onde **q** (grau do depósito) é menor que 2K. O teorema seguinte (demonstrado em Fisher[94.a]) estabelece como obter uma troca admissível de maneira que a nova K-árvore T_K' obtida seja a menor K-árvore com q+1 arestas incidentes ao depósito.

Teorema IV.5: (Incremento do número de arestas incidentes ao depósito)

Suponha que T_K seja ótima para o problema P(q) e $T_K' = T_K - e \cup \{ f^0 \}$ onde (e, f^0) é uma troca admissível com $e \in T_k$ não incidente ao depósito, $f^0 \notin T_k$ incidente ao depósito e $c(f^0)$ - c(e) seja mínimo entre todas as trocas admissíveis que aumentam o grau do depósito. Então T_k' será ótima para P(q+1).

Prova:

A prova consiste basicamente em mostrar que T_K' satisfaz as condições de otimalidade no teorema IV.4, sabendo-se de antemão, que T_K satisfaz as condições de otimalidade e $c(f^0)$ - c(e) é mínimo entre todas as possíveis trocas admissíveis que incrementam o grau do depósito.

Suponha inicialmente que a condição (i) seja violada por uma troca admissível satisfatória (e_1, f_1) em T_K' onde e_1 e f_1 são não incidentes ao depósito (sendo $e_1 \in T_K'$ e $f_1 \notin T_K'$). Seja $T_K'' = T_K' - e_1 \cup \{ f_1 \} = T_K - e_1 - e \cup \{ f^0, f_1 \}$.

Note que, se $f_1 = e$, teremos $T_K'' = T_K - e_1 \cup \{ f^0 \}$ e $c(T_K'') \geq c(T_K')$ (já que $c(f^0) - c(e_1) \geq c(f^0) - c(e)$ em razão da troca (f^0, e)). Portanto, (e_1, f_1) não seria uma troca admissível satisfatória em T_K'.

Considere agora que $e_1 \neq f_1$. Do lema IV.2, existirá um emparelhamento de e_1, e com f_1, f^0 gerando trocas admissíveis em T_K. A troca admissível satisfatória (e_1, f_1) não poderá ser admissível em T_K já que T_K satisfaz as condições de otimalidade (i). Assim, (e_1, f^0) (e, f_1) são ambas admissíveis em T_K. Deveremos ter portanto, $c(f_1) \geq c(e)$ pois T_K satisfaz as condições de otimalidade expressas em (i). Note que, $c(f^0) - c(e_1) \geq c(f^0) - c(e)$

já que (e, f^0) define uma troca de custo mínimo. Assim $c(T_K'') \geq c(T_K')$, o que é absurdo pois (e_1, f_1) é uma troca admissível satisfatória em T_K'.

Considere agora que a condição (ii) seja violada por uma troca admissível satisfatória (e_1^0, f_1^0), sendo e_1^0 e f_1^0 incidentes ao depósito. A prova de (ii) é análoga ao caso anterior. Seja $T_K'' = T_K' - e_1^0 \cup \{f_1^0\} = T_K - e - e_1^0 \cup \{f^0, f_1^0\}$. Se temos $e_1^0 = f^0$, basta observar que $c(T_K'') \geq c(T_K')$ pois (e, f^0) é uma troca admissível mínima. Se $e_1^0 \neq f^0$, do lema IV.2, existirá um emparelhamento de (e, e_1^0) com (f^0, f_1^0) definindo trocas admissíveis. A troca admissível satisfatória (e_1^0, f_1^0) não pode ser admissível em T_K já que T_K satisfaz a segunda condição de otimalidade. Portanto, (e_1^0, f^0) e (e, f_1^0) são ambas admissíveis em T_K. A primeira não pode ser admissível satisfatória em T_K (pois T_K satisfaz a condição (ii)), e a segunda não pode ser melhor que (e, f^0). Portanto, $c(T_K'') \geq c(T_K')$, o que é absurdo pois (e_1^0, f_1^0) é admissível satisfatória em T_K'.

Finalmente, se a terceira condição de otimalidade é violada, teremos um par de trocas (e_1, f_1^0) e (e_2^0, f_2) que juntas serão satisfatórias. Como as condições de otimalidade (i) e (ii) para T_K' já foram comprovadas, teremos (do lema IV.3) que estas duas trocas poderão ser feitas seqüencialmente para gerar uma K-árvore $T_K'' = T_K' - e_1 - e_2^0 \cup \{f_1^0, f_2\} = T_K - e - e_1 - e_2^0 \cup \{f^0, f_1^0, f_2\}$. Os casos onde $e = f_2$ ou $e_2^0 = f^0$ podem ser desconsiderados com base nos mesmos argumentos utilizados para as condições de otimalidade (i) e (ii). Assim, do lema IV.2, existirá um emparelhamento de e, e_1 e e_2^0 com f^0, f_1^0 e f_2, definindo trocas admissíveis em T_K. As seis possibilidades de emparelhamentos são:

$$(e_1, f_1^0); (e_2^0, f_2); (e, f^0) \qquad (1)$$

$$(e_2^0, f_1^0); (e_1, f_2); (e, f^0) \qquad (2)$$

$$(e, f_1^0); (e_2^0, f_2); (e_1, f^0) \qquad (3)$$

$$(e_2^0, f_1^0); (e, f_2); (e_1, f^0) \qquad (4)$$

$$(e_1, f_2); (e_2^0, f_2^0); (e, f_1^0) \qquad (5)$$

$$(e_1, f_2); (e_2^0, f^0); (e_1, f_1^0) \qquad (6)$$

Note, em cada um dos seis casos apresentados, que as duas primeiras trocas não podem ser admissíveis satisfatórias, já que T_K satisfaz as condições de otimalidade (i), (ii) e (iii) respectivamente. A terceira troca por sua vez, será igual a (e, f^0), ou a uma troca que não pode ser melhor que (e, f^0). Assim $c(T_K'') \geq c(T_K')$. Segue que a condição (iii) acima não poderá ser violada. •

Suponha agora que o grau do depósito seja igual a q > 2K na K-árvore T_K. O teorema seguinte estabelece como obter uma troca admissível de maneira que a nova K-árvore T_K' obtida seja a menor K-árvore de grau q-1.

Teorema IV.6: (Decremento do número de arestas incidentes ao depósito)

Suponha que T_K seja ótima para o problema P(q) e $T_K' = T_K - e^0 \cup \{f\}$ onde (e^0, f) é uma troca admissível com $e^0 \in T_K$ incidente ao depósito, $f \notin T_K$ não incidente ao depósito e $c(f) - c(e^0)$ seja mínimo entre todas as trocas admissíveis que reduzem o grau do depósito. Então T_K' será ótima para P(q-1).

Prova:

Demonstração análoga ao teorema IV.5. •

Vejamos agora, baseado nos resultados teóricos apresentados acima como incrementar ou decrementar o grau do depósito a partir de uma K-árvore mínima de grau **q** no depósito onde q <2K ou q >2K respectivamente. Caso a K-árvore mínima inicial tenha grau q=2K no depósito o problema já estará resolvido.

Vejamos primeiramente como decrementar o grau do depósito:

IV.3.1 - Decremento do Grau do Depósito:

Seja T_K uma K-árvore mínima com q > 2K arestas incidentes ao depósito. Eliminando todas as arestas incidentes ao depósito teremos **m** componentes conexas. Para cada nó **i** pertencente a uma componente atribuímos um mesmo rótulo *l(i)*. Caso uma componente seja conectada ao depósito por duas ou mais arestas (adjacentes ao depósito), qualquer uma destas arestas poderá ser eliminada sem desconectar a componente do depósito. Seja C o maior custo entre todas estas arestas.

Considere também C_i o maior custo entre todas as arestas de T_K que unem a componente **i** ao depósito.

Vejamos o seguinte exemplo (figura IV.7), onde ilustramos a obtenção de uma 2-árvore com grau 4 no depósito a partir de uma 2-árvore de grau 5 no depósito. Os respectivos valores de C e C_i para cada componente e os rótulos *l*(i) estão também representados:

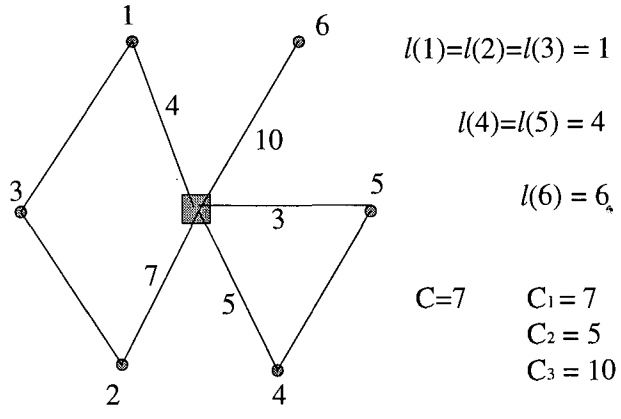


Figura IV.7: Determinção de C e Ci.

Note que C = 7 é o valor da maior aresta que une uma componente qualquer conectada ao depósito por 2 ou mais arestas (neste caso as componentes 1 e 2).

Desejamos eliminar de T_K uma aresta incidente ao depósito e substituí-la por uma aresta não pertencente a T_K e não incidente ao depósito (teorema IV.6).

Para determinar a melhor troca definimos:

$$w_{ij} = \begin{cases} c_{ij} - C & l(i) = l(j) \\ c_{ij} - max\{C, C_{l(i)}, C_{l(j)}\} & l(i) \neq l(j) \end{cases}$$

onde $i \neq 0; j \neq 0 \ \ e \ \ (i,j) \notin T_K$.

Para obtermos a melhor substituição bastará encontrar a aresta (i,j) (não pertencente a T_K) tal que w_{ij} seja mínimo entre todas as possíveis escolhas de (i,j). Logo, do teorema IV.6, adicionamos (i, j) e retiramos a aresta associada a C (caso l(i)=l(j)) ou uma das arestas associadas a $C, C_{l(i)}$ *ou* $C_{l(j)}$ respectivamente (se l(i)≠l(j)). Note que as arestas a serem retiradas são adjacentes ao depósito.

No exemplo representado na figura IV.7, teremos $w_{56} = c_{56} - max\{7,5,10\}$. Assim, se w_{56} for o mínimo entre todas as possíveis escolhas de w_{ij}, adicionamos a aresta (5,6) e retiramos a aresta (0, 6) (associada a $C_{l(6)} = 10$) obtendo uma nova 2-árvore mínima com 4 arestas incidentes ao depósito.

Caso a substituição tenha sido feita entre vértices com rótulos distintos (como no exemplo acima) deveremos atualizar todos os rótulos na componente **j** por ***l*(i).**

Resumindo, teremos o seguinte procedimento (onde q (grau do depósito) é maior que 2K).

PROCEDIMENTO: Decrementa-grau;

Início

- Determinar m componentes conexas l(i) (onde i=1,..,m);

Repita

- Calcula $C \ \ e \ \ C_i \quad p/ \quad i = 1,..,m$
- Calcula $w_{ij}, \ \forall \ (i,j) \notin T_K \ \ e \ \ i \neq 0, j \neq 0$;
- Seja (p, q) tal que w_{pq} é o mínimo de w_{ij} entre todas as possíveis escolhas de (i, j);
- Obter aresta (0, s) associada a C (se l(p) = l(q)) ou $max\left(C, C_{l(p)}, C_{l(q)}\right)$ se l(p) ≠ l(q);
- Adicionar (p, q) e retirar (0, s) de T_K;

- Atualizar rótulos se $l(p) \neq l(q)$;

- q := q-1;

Até que q = 2K;

fim. {procedimento}

Figura IV.8: Procedimento decrementa grau

Na seção IV.3.3, discutiremos algumas questões relativas à implementação e estruturas de dados utilizada. Como veremos, este procedimento poderá ser executado em no máximo $O(n^3)$ passos.

IV.3.2 - Incremento do Grau do Depósito:

Tratemos agora da situação onde temos o número de arestas incidentes ao depósito menor que 2K na K-árvore mínima inicial.

Esta situação é um pouco mais delicada que a anterior. Note que devemos ter um cuidado maior na retirada de alguma das arestas não incidentes ao depósito. A eliminação de algumas dessas ligações pode tornar o grafo desconexo.

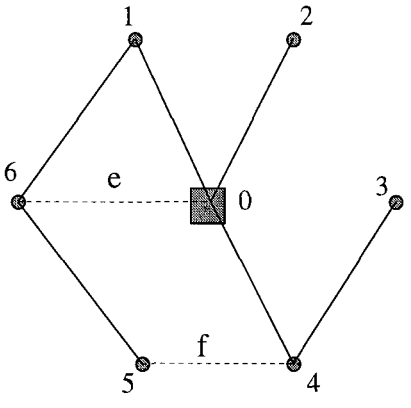
Seja $T_K = T \cup S_0 \cup S_1$, onde T é uma árvore geradora mínima contida em T_K e $S_0 \cup S_1 = T_K - T$. Todas as arestas de S_0 são adjacentes ao depósito e todas as arestas de S_1 são não-adjacentes ao depósito.

Considere T uma árvore com raiz no vértice 0 (depósito) e seja j(i) o predecessor do nó **i** nesta árvore. Utilizando as idéias apresentadas em Glover & Klingman [74], Fisher[94.a] considera:

$$w_i = \begin{cases} -\infty & i = 0 \quad ou \quad j(i) = 0 \\ max\{w_{j(i)}, c_{ij(i)}\} & caso \ contrario \end{cases}$$

Note que para $i \neq 0$ e $j(i) \neq 0$, w_i representa o custo da maior aresta não incidente ao depósito no único caminho em T de **i** até **0**.

Para ilustrar a notação acima vejamos o seguinte exemplo onde temos uma $T_K = T \cup S_0 \cup S_1$ (2-árvore). A árvore geradora T está representada em linhas contínuas. As arestas **e** e **f** pertencem a S_0 *e* S_1 respectivamente:



w(0) = - inf. i=0

w(1) = - inf. j(1) =0

w(2) = -inf. j(2) =0

w(3) = max{w4, c34} = c34 j(3) =4

w(4) = - inf. j(4) =0

w(5) = max{w6, c56} = ? j(5) =6

w(6) = max{w1, c61} = c61 j(6) =1

Figura IV.9: Incremento do grau do depósito

O vetor **w** deverá ser obtido em 3 etapas. Primeiro inicializamos w_i com -∞ para os vértices **i** que tenham o depósito como predecessor. Em seguida calculamos w_i para todos os vértices onde $w_{j(i)} \neq -\infty$, e finalmente calculamos w_i para $w_{j(i)} \neq -\infty$. Procedendo dessa maneira descobrimos o valor de w(5) na figura IV.9 acima. Como discutido adiante (na próxima seção), utilizamos uma lista de arestas na armazenagem dos dados. Neste caso, o vetor **w** poderá ser construído a medida que percorremos as arestas de T utilizando uma estratégia do tipo *backtracking* (Horowitz e Sahni[78]).

Considere agora a(i) uma aresta de T no caminho de **i** até **0** onde $c(a(i)) = w_i$, ou seja, a(i) é a aresta de maior custo neste caminho. Considere também:

$$C_1 = max\{c(f) / f \in S_1\}$$

$$C_2 = max\{w_i / (0,i) \in S_0\}$$

Seja $a_1 \in S_1$ onde $c(a_1) = C_1$, ou seja, a_1 é a maior aresta de S_1 não incidente ao depósito. Considere também (0, i*)$\in S_0$ onde $C_2 = w_{i*}$ e $a_2 = a(i*)$, ou seja, a_2 é a maior aresta de T não-incidente ao depósito no caminho de i* até 0. Finalmente, para toda aresta $(0, i) \notin T_K$, seja $\bar{c}_i = c_{0i} - max\{C_1, C_2, w_i\}$.

Vejamos agora, como identificar a troca admissível de custo mínimo incrementando paralelamente o grau do depósito (teorema IV.5).

Como observado anteriormente, dado uma K-árvore T_K e uma aresta $e' \notin T_K$, representamos por T_K(e'), o conjunto de arestas de T_K que definem uma troca admissível com e'. Note que T_K(e'), contém precisamente aquelas arestas de T_K que podem ser eliminadas de $T_K \cup$\{e'\} sem destruir sua conectividade. Analogamente, para uma árvore T e $e' \notin$T, seja T(e')$\subseteq$T, o conjunto de arestas que podem ser eliminadas de T$\cup$\{e'\} sem destruir sua conectividade.

Dado um conjunto S satisfazendo a S$\cap$T=$\varnothing$ e S$\cap T_K$=$\varnothing$, podemos estender as definições acima e representar por T(S) e T_K(S), o conjunto de arestas em T ou T_K que podem ser eliminadas de T$\cup$S ou $T_K \cup$S sem perda da conectividade.

Temos então o seguinte lema:

Lema IV.4: $T(S) = \bigcup_{e' \in S} T(e')$.

Prova:

Claramente temos $\bigcup_{e' \in S} T(e') \subseteq T(S)$. Assim, é suficiente mostrar que qualquer $e \in T(S)$ pertence também a $T(e')$ para algum $e' \in S$. Note que T $\cup$ S - e é conexa, e portanto deve conter uma árvore T'. Do lema IV.1, as arestas de T'-T podem ser emparelhadas com as arestas de T-T' de maneira que cada par defina uma troca admissível em T. Então $e \in T(e')$, onde e' é a aresta associada a e. •

O teorema seguinte (demonstrado em Fisher [94.a]), nos dá uma troca admissível para T_K incrementando o grau do depósito de maneira que a próxima K-árvore obtida seja mínima.

Teorema IV.7: (Melhor troca admissível)

A troca de menor custo exigida no teorema IV.5 corresponde a adicionarmos a aresta (0, i) que minimiza $\bar{c}_i$, e eliminarmos a_1, a_2 *ou* $a(i)$ conforme C_1, C_2 *ou* w_i seja o máximo respectivamente na expressão definindo $\bar{c}_i$.

Prova:

Devemos mostrar que $max\{C_1, C_2, w_i\}$ é o custo da maior aresta em $T_k^1(\{0,i\})$ (arestas de $T_K(\{0,i\})$ não incidentes ao nó 0). Note que as arestas de $T_K^0(\{0,i\})$ contém S_1 e as arestas de T não incidentes ao nó 0, que podem ser eliminadas sem a perda da conectividade. Seja $T_1(e)$, as arestas de T não incidentes ao nó 0 que podem definir uma troca admissível com *e*. Do lema IV.4 , temos:

$$T_K^1(\{o,i\}) = S_1 \cup \left(\bigcup_{e \in S_1} T_1(e) \right) \cup \left(\bigcup_{e \in S_0} T_1(e) \right) \cup T_1(\{0,i\})$$

Aqui, C_1 é o custo máximo associado a uma aresta de S_1. Nenhuma aresta de $\bigcup_{e \in S_1} T_1(e)$ poderá ter um custo superior a C_1 ou chegaríamos a uma contradição já que T é árvore geradora mínima (contida em T_K). Finalmente, C_2 *e* w_i são os custos máximos, associados a alguma aresta de $\bigcup_{e \in S_0} T_1(e)$ e $T_1(\{0,i\})$ respectivamente. •

Para ilustrar esse resultado vejamos o seguinte exemplo onde temos uma 4-árvore com grau 6 no depósito:

Supondo $c_{05} = 9$ *e* $c_{03} = 11$ teremos:

$$\bar{c}_3 = c_{03} - max(5,8,3) = 11 - 8 = 3$$

$$\bar{c}_5 = c_{05} - max(5,8,7) = 9 - 8 = 1$$

Segue, do teorema IV.3.5 que a melhor substituição será colocarmos (0,5) e retirarmos $a_2 = (4,5)$ associado a $C_2 = 8$.

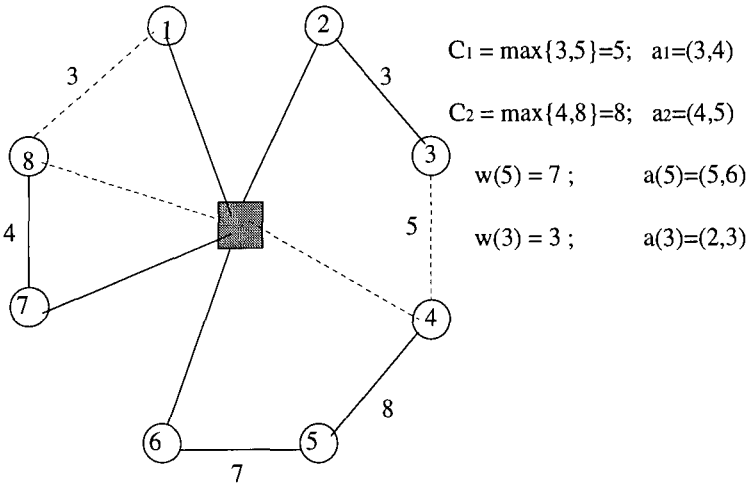


Figura IV.10: Incremento do Grau do Depósito

Note que se adicionamos $(0,i) \notin T_k$ e retiramos a_1 por exemplo, não tornaremos o grafo desconexo já que $a_1 \notin T$. Da mesma forma, se a_2 for retirado não teremos um grafo desconexo pois $a_2 = a(i^*)$ pertence ao caminho de i* até 0 em T (note que que i* já está conectado ao depósito por (0,i*)$\in S_0$). Assim, com a presença das arestas (0,i*) e a_2 teremos a formação de um ciclo e a_2 poderá ser eliminada sem destruir a conexidade de T_K. Finalmente, se eliminamos a(i)$\in$ T, ao acrescentarmos (0,i) continuamos com um grafo conexo com mais uma aresta incidente ao depósito. A inserção de (0, i) forma um ciclo e a(i) poderá ser retirada sem nenhum problema.

Resumindo todos os passos descritos acima teremos o seguinte algoritmo:

PROCEDIMENTO: Incrementa-Grau;

Início

Repita

- Utilizando T calcular j(i) p/ i=1,..,n;

- Utilizando **j** e T calcular w_i p/ i=1,..,n;

- Calcular : C_1 *e* C_2;

- Calcular: $\bar{c}_i = c_{0i} - max(C_1, C_2, w_i)$ p/ (0,i) $\notin T_K$;

- Adiciona em T_K a aresta (0,i) associada ao mínimo de $\bar{c}_i$ p/ (0,i) $\notin T_K$;

- Retira a_1, a_2 *ou* $a(i)$ associado ao máximo de C_1, C_2 e w(i) respectivamente.

- Recalcular as arestas pertencentes a T, S_0 *e* S_1;

- q:=q+1;

Até que q=2K;

fim. {procedimento}

Figura IV.11: Procedimento increnta grau

Observe que após a troca das arestas, necessitamos reorganizar nossa K-árvore determinando quais arestas pertencem a T, S_0 *e* S_1 respectivamente.

Como veremos na seção seguinte, o procedimento incrementa-grau poderá ser implementado em $O(n^3)$ passos.

IV.3.3 - Complexidade e Estruturas de Dados (K-árvore):

Todas as arestas do grafo são armazenadas em uma lista L de **m** arestas formada por 3 campos (dois campos ENDV1 e ENDV2 representando os extremos de uma aresta e outro campo WEIGHT representando o custo desta aresta). Note que, no máximo teremos m=n(n+1)/2 arestas nesta lista (grafo completo).

Como discutido anteriormente, devemos calcular uma K-árvore mínima inicial onde $T_K = T \cup S_0 \cup S_1$. O procedimento utilizado é uma variação do algoritmo de Kruskal[56] onde se faz uma ordenação parcial das arestas do grafo. Através de trocas sucessivas, todas as arestas de L serão armazenadas satisfazendo a estrutura de *heap*, de maneira que a raiz contenha sempre a aresta de menor custo do grafo.

Definimos uma nova lista TK com n+K componentes que irá armazenar nossa K-árvore mínima inicial. Teremos então **n** arestas em T (já que nosso grafo é composto por n+1 vértices) e K arestas pertencentes a S_0 *e* S_1 respectivamente. Apresentamos na figura IV.12, a estrutura utilizada.

Cada aresta retirada desse *heap* (lista L) deverá ser transportada (se possível) para TK até que n+K arestas tenham sido inseridas. A idéia é armazenar nossa árvore T nas n posições iniciais de TK e as arestas pertencentes a S_0 *e* S_1 nas posições que variam entre n+1 e n+K. Este procedimento será importante principalmente no incremento do grau do depósito onde o vetor W (discutido na seção anterior), e o vetor de predecessores J é calculado apenas utilizando-se as arestas de T, situado nas n primeiras posições de TK.

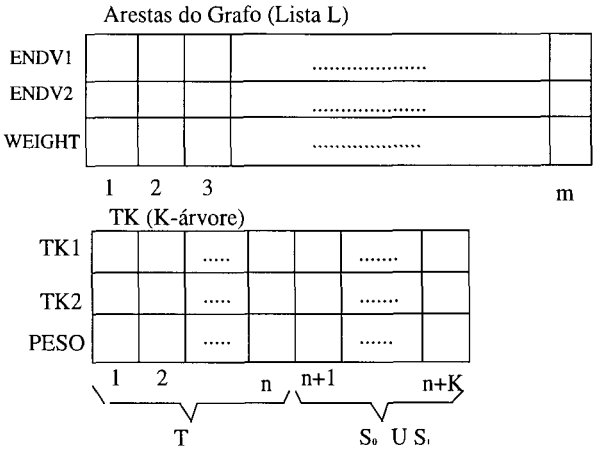


Figura IV.12 : Lista de Arestas

Toda aresta retirada do *heap*, será adicionada à árvore geradora mínima caso sua inclusão não forme um ciclo com as arestas de T já selecionadas. Caso contrário, esta aresta deve ser inserida nas posições correspondentes a S_0 *e* S_1. Este procedimento é repetido até que a árvore T esteja completa.

Necessitamos O(m) passos para construção do *heap* inicial e O(log m) passos para sua reconstrução após a retirada de cada aresta presente na raiz (Syslo et al.[83]). Se p arestas forem pesquisadas até a obtenção de TK, teremos uma complexidade total de O(m+plog m) passos para a determinação da K-árvore mínima inicial. Podemos expressar

essa complexidade em função do número de vértices fazendo p=O(m)= $O(n^2)$ (pior caso). Neste caso teremos $O(n^2 + 2n^2 \log n)$, ou ainda $O(n^2 \log n)$ passos para construção da K-árvore mínima inicial.

Note que, após a construção de TK, temos n_a = m-(n+K) = $O(n^2)$ arestas na lista L resultante (já que m=$O(n^2)$). Nos procedimentos decrementa-grau e incrementa-grau, fazemos trocas entre as arestas de L (arestas não pertencentes a TK) e TK, atualizando o grau do depósito. Serão necessários portanto, $O(n^2)$ passos (no pior caso) para executarmos cada uma das trocas que atualizam o grau do depósito. Assim, nos procedimentos decrementa-grau e incrementa-grau, teremos O(n) trocas (no pior caso) até que se tenha grau 2K no depósito. Segue então que a complexidade total será de $O(n^3)$ iterações em ambos os casos.

Analisando-se os procedimentos calcula K-árvore mínima inicial, decrementa-grau e incrementa-grau separadamente, obtemos uma complexidade total de $O(n^3)$ passos na obtenção da K-árvore mínima com 2K arestas incidentes ao depósito. Como discutido anteriormente, o custo associado à K-árvore mínima irá contribuir na determinação de um limite inferior para o valor da solução ótima do problema original (roteamento).

Feitas as devidas considerações para a resolução da relaxação lagrangeana (obtenção da K-árvore mínima), vejamos como identificar algumas das restrições violadas por esta solução:

IV.4 - Geração de Restrições:

Trataremos agora da abordagem utilizada por Fisher[94.b] na solução do problema lagrangeano. Seja $X = \left\{ x \,/\, x \in \{0,1\}^{|E|} \text{ define uma K - arvore e } \sum_{i=1}^{n} x_{0i} = 2K \right\}$. Como discutido anteriormente (seção IV.2), um limite inferior para z* (solução ótima do problema original) é obtido resolvendo-se o seguinte problema lagrangeano:

$$z_D(u,v) = min \sum_{ij \in E(N_0)} \bar{c}_{ij} x_{ij} + 2\sum_{i=1}^{n} u_i + 2\sum_{S \subseteq N} v_S r(S) \qquad (9)$$

$$s.a. \quad x \in X$$

onde:

$$u_i \in \Re, \quad v_S \geq 0, \quad u_0 = 0 \quad e \quad \bar{c}_{ij} = c_{ij} - u_i - u_j - \sum_{\substack{i \in S,\ j \in \bar{S} \\ ou \\ i \in \bar{S},\ j \in S}} v_S$$

Os valores $u_i \in \Re$ (para i=1,..,n) e $v_S \geq 0$ podem ser arbitrados (usualmente inicializados em zero) e ajustados com a utilização do método subgradiente. Como temos um número muito grande de restrições de capacidade, devemos nos restringir apenas a um subconjunto destas. Assim, restrições de capacidade serão dualizadas apenas quando violadas pela solução do problema lagrangeano corrente. Da mesma forma, restrições de capacidade podem ser descartadas quando seus multiplicadores associados se tornarem iguais a *zero*. Fisher[94.b] entretanto, mantém explicitamente dualizadas, restrições com multiplicadores nulos durante um número pré-determinado de iterações do subgradiente.

Antes de tratarmos diretamente da geração do conjunto de restrições inicial, vejamos um exemplo particular onde são atribuídos valores aos multiplicadores $u_i \in \Re$ *e* $v_S \geq 0$, obtendo $z_D(u^*, v^*) = z^*$ na resolução do problema lagrangeano.

O seguinte exemplo apresentado por Fisher[94.b], serve como motivação para sua estratégia na determinação de um número inicial de restrições de eliminação de sub-rotas candidatas à dualização.

Considere um problema de roteamento onde cada cliente **i** está a uma distância d_i do depósito, $c_{ij} = |d_i - d_j|$, $\forall\ i, j \in N$; $K = \lceil n/b \rceil = n/b$ e finalmente, $\bar{d}_j = 1$ para todo cliente **j** em N (excepcionalmente nesta seção representaremos por $\bar{d}_j$ a demanda do cliente j). Para simplificar a notação, assumimos n/b inteiro e $d_1 \leq d_2 \leq \ldots \leq d_n$. É fácil ver que uma solução viável para este problema pode ser obtida atribuindo o veículo 1 aos clientes 1,2,..,b; o veículo 2 aos clientes b+1,b+2,...,2b e assim sucessivamente até que o veículo K seja atribuído aos clientes n-b+1,...,n. O valor da função objetivo associado a esta solução será $Z = 2(d_b + d_{2b} + \ldots + d_{n-b} + d_n)$. Para ilustrar essa situação vejamos o seguinte exemplo onde temos n=12, b=4 e consequentemente K=3 veículos:

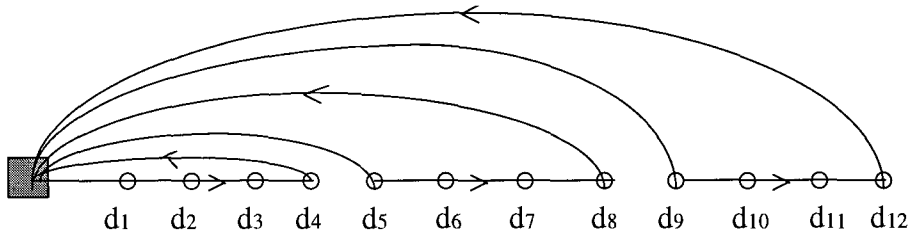


Figura IV.13: Determinação do multiplicadores de Lagrange (exemplo particular)

Note que, se temos apenas distâncias euclideanas entre os clientes, todos esses clientes deverão estar em uma mesma reta, já que $c_{ij} = |d_i - d_j| \ \forall \ i,j \in N$!

Considere agora a seguinte atribuição dada aos multiplicadores de Lagrange $u_i \in \Re \ e \ v_S \geq 0$:

$$u_i^* = 0, \quad para \ i = 1,2,\ldots,n-1$$
$$u_n^* = d_n - d_{n-1};$$
$$v_{S_k}^* = d_k - d_{k-1}; \quad para \ k = 1,2,..,n-1 \quad onde \ S_k = \{k,..,n\}$$
$$v_S^* = 0, \quad \forall \ S \subseteq N \ e \ S \neq S_k \quad (\text{sen}do \ |S| \geq 2)$$

Mostraremos que, com a atribuição dada aos multiplicadores acima obtemos $z_D(u^*, v^*) = z^*$. De fato, fazendo a substituição dos valores u* e v* na função lagrangeana obtemos:

$$z_D(u^*, v^*) = min \sum_{(i,j)\in E(N_0)} \bar{c}_{ij} x_{ij} + 2(d_n - d_{n-1}) + 2\sum_{k=1}^{n-1} v_{S_k} \left\lceil \bar{d}(S_k)/b \right\rceil \qquad (10)$$
$$s.a. \quad x \in X$$

Note entretanto que:

$$\bar{c}_{ij} = c_{ij} - u_i - u_j - \sum_{\substack{i\in S_k, j\in \bar{S}_k \\ ou \\ i\in \bar{S}_k, j\in S_k}} v_{S_k} = 0, \quad \forall (i,j) \in N. \qquad (11)$$

Para ilustrar esse fato, vejamos o seguinte exemplo onde temos i=3, j=6 n=6:

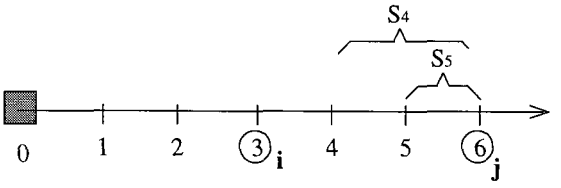


Figura IV.14: Cálculo de $\bar{c}_{ij}$

Como definido acima, S_k é um conjunto onde $i \in S_k$ e $j \in \bar{S}_k$. Assim, teremos S_k={k,..,6} para k=4,5. Calculando $\bar{c}_{63}$ e desenvolvendo (11) para o exemplo da figura IV.13 temos:

$$\bar{c}_{63} = d_6 - d_3 - u_3 - u_6 - (d_4 - d_3 + d_5 - d_4)$$

Fazendo $u_3 = 0$ e $u_6 = d_6 - d_5$ obtemos $\bar{c}_{63} = 0$ após as devidas simplificações. Este caso é facilmente extensível ao caso geral onde temos **i** e **j** pertencentes a N.

Generalizando o processo temos $\bar{c}_{ij} = 0$, $\forall\, i, j \in N$, segue que o custo associado à K-árvore mínima também será nulo. Voltando à expressão (10) obtemos:

$$z_D(u^*, v^*) = 2(d_n - d_{n-1}) + 2\sum_{k=1}^{n-1}(d_k - d_{k-1})\left\lceil \frac{n+k-1}{b} \right\rceil$$

$$= 2(d_n - d_{n-1}) + 2\left((d_1 - d_0)\left\lceil \frac{n}{b} \right\rceil + (d_2 - d_1)\left\lceil \frac{n-1}{b} \right\rceil + \ldots + (d_{n-1} - d_{n-2})\left\lceil \frac{n+1-(n-1)}{b} \right\rceil\right)$$

Efetuando o produto e reagrupando as parcelas obtemos:

$$z_D(u^*, v^*) = 2\sum_{k=1}^{n}\left(\left\lceil \frac{n+1-k}{b} \right\rceil - \left\lceil \frac{n-k}{b} \right\rceil\right)d_k = 2\left(d_b + d_{2b} + \ldots + d_{n-b} + d_n\right) = z^*$$

O exemplo da figura IV.13, irá sugerir a construção de uma heurística particular na determinação de um conjunto de restrições inicial. Observe no exemplo apresentado, que os subconjuntos S_k eram "aninhados" em torno do cliente mais afastado do depósito. Para

reproduzir essa situação, Fisher[94.b] define um conjunto de **m** clientes semente $s_1, s_2, \ldots, s_m$ e constrói um "aninhamento" de clientes em torno de cada uma destas sementes.

Em seu trabalho, Fisher[94.b] utiliza m=K+3, de maneira que os clientes semente $s_1, s_2, \ldots, s_K$ representem os clientes mais distantes do depósito em cada uma das K rotas geradas por uma solução heurística inicial. Os 3 clientes restantes são escolhidos um a um de tal forma que, a distância entre os outros clientes e o depósito seja a maior possível. Assim, se S_D representa o conjunto de clientes semente escolhidos até o momento, o próximo cliente s_i é selecionado da seguinte forma:

$$\max_{s_i \notin S_D} \left\{ \min \left\{ c_{os_i}, \min_{j \in S_D} c_{js_i} \right\} \right\}$$

Sejam $i_1, i_2, \ldots, i_{n-1}$ os índices associados aos clientes (em ordem crescente de proximidade) da semente s_i. Assim, o conjunto de restrições inicial associado a s_i corresponde aos conjuntos:

$$\{s_i, i_1\}$$
$$\{s_i, i_2\}$$
$$\{s_i, i_1, i_2\}$$
$$\{s_i, i_1, i_2, i_3\}$$
$$\{s_i, i_1, i_2, i_4\}$$
$$\{s_i, i_1, i_2, i_3, i_4\}$$
$$\vdots \quad \vdots$$

No trabalho desenvolvido por Fisher[94.b] o número máximo de conjuntos utilizados para cada semente s_i é igual a 60. Além disso, as restrições associadas às partições acima, são adicionadas à função lagrangeana a medida que forem sendo violadas pelo solução do problema relaxado (K-árvore mínima c/ 2K arestas incidentes ao depósito). Isto ocorre após as primeiras 50 iterações do método subgradiente. Como discutido anteriormente, as restrições de eliminação de sub-rotas são eliminadas do conjunto ativo caso se tenha 3 iterações consecutivas do subgradiente com multiplicadores nulos.

Note que Fisher utiliza um procedimento heurístico na identificação de restrições violadas! No capítulo VI, resolvemos de maneira exata o problema da identificação de sub-rotas violadas. Como veremos, assim como em Fisher[94.b], trabalhamos com um "pequeno" número de restrições dualizadas a cada iteração do subgradiente. Em nosso caso entretanto, permitimos que um número bem maior de restrições candidatas à dualização seja utilizado (2^n restrições). Isto irá permitir a geração de limites inferiores melhores que os obtidos por Fisher[94.b] para o problema de roteamento. Resultados computacionais comparando essas duas abordagens serão apresentados no capítulo IX.

IV.5 - Heurísticas Lagrangeanas:

Façamos agora uma breve discussão acerca de algumas das heurísticas lagrangeanas utilizadas por Fisher[94.b].

Nas heurísticas lagrangeanas, soluções viáveis são obtidas a cada iteração do método subgradiente e podem ser implementadas utilizando-se, por exemplo, a solução do problema lagrangeano $\bar{x}_{ij}$ ou os custos reduzidos $\bar{c}_{ij}$. Fisher[94.b] propõe a utilização de 3 heurísticas lagrangeanas distintas. Testes computacionais entretanto, indicam que não há predominância de nenhuma destas heurísticas em relação as outras.

Ao final de cada heurística faz-se uma busca local na intenção de se obter uma melhor solução viável (menor custo). A busca local pode ser realizada em cada uma das K rotas geradas através de uma das heurísticas lagrangeanas. Ao final da heurística lagrangeana, Fisher utiliza o algoritmo 3-optimal para busca local (vide Lin & Kernigham[73]).

Apresentamos a seguir, uma das heurísticas apresentada em Fisher[94.b]. O procedimento é iniciado selecionando-se o cliente mais próximo ao depósito. As distâncias neste caso são obtidas computando-se os custos lagrangeanos $\bar{c}_{ij}$ a cada iteração do subgradiente. Em seguida, selecionamos um cliente que esteja mais próximo do último cliente já selecionado. O processo é repetido até que a inclusão de um novo cliente supere a capacidade do veículo ou se o depósito for selecionado como próximo vértice.

Note que para evitar a presença de soluções inviáveis, não permitimos a ligação do último cliente com o depósito se a demanda atendida na rota (sendo construída) for inferior a $\sum_{i \in N} d_i - (K-1)b$ (onde d_i é a demanda do cliente **i** e **b** a capacidade de cada veículo). Caso isto ocorra, os K-1 veículos restantes não serão suficientes para atender os clientes ainda não percorridos. Repetimos este procedimento gerando outras rotas até que todos os clientes tenham sido pesquisados.

Chamaremos de S, o conjunto de nós abertos ou ainda não percorridos. Um nó será fechado quando for atendido por algum veículo. O conjunto de nós fechados será representado por $\overline{S}$.

Rescrevendo os passos descritos acima temos o seguinte algoritmo:

PROCEDIMENTO: Heurística Lagrangeana;

Início

- custototal = 0;
- demrota := 0; {demanda parcial da rota pesquisada}
- Inicialmente temos i$\in$S , $\forall i \in N_0$; {S é o conjunto de nós abertos}
- base := 0;
- **Repita**
  - Seleciona nó **i** de S mais próximo da base s/ exceder a capac. **b** e s/ permitir soluções inviáveis;
  - **Se** i $\neq$ 0 **então**
    - demrota := demrota + dem[i];
    - Colocamos **i** em $\overline{S}$ {vértices já percorridos}
    - S := S - {i}; {decrementamos abertos}

  senão
    - demrota := 0;
  - **fim se-senão**;
  - base := i;

- custototal := custototal + distância[base, i];

- **Até que** S = ∅;

fim.

Figura IV.15: Heurística Lagrangeana

Note no algoritmo que definimos 2 conjuntos de nós abertos e fechados (S e $\overline{S}$ respectivamente). Inicialmente todos os nós (clientes) pertencem a S. Sempre que um novo cliente for percorrido, este nó será fechado. Nunca colocamos o depósito em $\overline{S}$ já que devemos percorrê-lo sempre que uma nova rota for fechada.

Note que nem sempre é possível a determinação exata das K rotas (como desejado). Podemos eventualmente obter soluções viáveis com um número maior de veículos. A determinação de uma solução viável para o problema de roteamento de veículos com exatamente K rotas, definidas a priori, é um problema NP-completo. Trata-se na verdade, de uma variação do problema de *bin packing* (Garey & Johnson [79]). Desta forma, a solução produzida pela heurística, em uma dada iteração do subgradiente, deverá ser descartada caso não se tenha exatamente as K rotas desejadas.

Na outra heurística apresentada por Fisher[94.b], ele considera apenas aquelas iterações cuja relaxação lagrangeana correspondente contenha exatamente K componentes conexas após a eliminação das arestas incidentes ao depósito. Se $C_1, C_2, .., C_K$ é o conjunto de clientes em cada uma destas componentes, a obtenção de K rotas viáveis parciais, é realizada tomando-se cada k com d(C_k)>b. Elimina-se em seguida, os clientes por ordem de proximidade do depósito até que d(C_k)≤b.

Em uma terceira heurística, apresentada por Fisher, tenta-se utilizar o maior número possível de arestas da solução lagrangeana. Seja $\overline{x}_{ij}$, as arestas presentes na solução lagrangeana em uma dada iteração. Escolhe-se o cliente i_1 mais distante do depósito e seleciona-se a menor aresta (i_1, i_2) tal que $\overline{x}_{i_1 i_2} = 1$ e $d_{i_1} + d_{i_2} \leq b$. Note que em um dado momento do algoritmo teremos uma rota parcial $i_1, i_2, \ldots, i_k$ para um veículo e de maneira que $\overline{x}_{i_1 i_2} = \overline{x}_{i_2 i_3} = \ldots = \overline{x}_{i_{k-1} i_k} = 1$. Neste momento, escolhe-se a menor aresta ligando algum cliente **j** ao cliente i_1 *ou* i_k e tal que $\overline{x}_{i_1 j} = 1$ ou $\overline{x}_{i_k j} = 1$ e $d(\{i_1, i_2, \ldots, i_k\}) \leq b$. Quando

nenhuma aresta puder ser encontrada e que satisfaça as condições exigidas, selecionamos um cliente ainda não pesquisado e que esteja mais distante do depósito. Repetimos o processo gerando rotas parciais até que K rotas tenham sido obtidas.

Capítulo V

Roteamento de Veículos e Combinatória Poliédrica

V.1 - Introdução:

A utilização de combinatória poliédrica tem se tornado cada vez mais freqüente na resolução de problemas de otimização combinatória. Alguns dos resultados mais expressivos desta utilização são, sem dúvida, aqueles obtidos para o problema do caixeiro viajante (equivalente ao problema de roteamento de veículos com um único veículo de capacidade infinita). Um maior conhecimento da estrutura poliédrica deste problema tem possibilitado a resolução exata de instâncias com milhares de clientes (vide Padberg e Rinaldi[87], [89], [91]; Jünger et al. [92]).

Atualmente, os códigos exatos mais sofisticados para a resolução do problema do caixeiro viajante incorporam um número grande de famílias de desigualdades válidas fortes para o problema (desigualdades de eliminação de sub-rotas, *2-matchings*, *combs*, *path*, *clique-trees*, entre outras). Isto é feito partindo-se de uma relaxação linear e utilizando-se algoritmos de separação exatos ou aproximados para a obtenção de desigualdades válidas

violadas pela relaxação corrente. Os cortes são inseridos, e o processo é repetido até que não se "consiga" determinar novas restrições violadas.

Apesar dos resultados expressivos obtidos para o problema do caixeiro viajante, ainda pouco se conhece sobre a estrutura poliédrica do problema clássico de roteamento de veículos (PRVRC). Isto se aplica mesmo para o caso particular envolvendo demandas idênticas. Trata-se na verdade, de um problema onde a determinação de uma solução exata parece ser bastante difícil. Pode-se constatar na literatura existente, soluções exatas para problemas contendo cerca de 50 a 60 clientes e algumas instâncias contendo cerca de 100 a 150 clientes (p. ex. Araque[90], Fisher[94.b]). Isto pode parecer pouco quando comparado ao problema do caixeiro viajante! Na verdade, mesmo a determinação da dimensão do poliedro associado ao problema de roteamento é um problema NP-completo (Campos et al.[91], Cornuejols e Harche[93]), fato que não ocorre com o problema do caixeiro viajante.

Faremos primeiramente algumas considerações iniciais sobre a complexidade do problema e a notação utilizada. Em seguida, apresentamos alguns resultados existentes na literatura sobre a estrutura poliédrica do problema de roteamento com demandas distintas e apresentamos algumas desigualdades conhecidas para o problema (restrições de eliminação de sub-rotas, *combs* e *multistars*) .

A identificação de restrições violadas a partir da solução do problema dual lagrangeano (K-árvore mínima) será discutida com mais detalhes no capítulo seguinte.

V.2 - Notação e Considerações Iniciais:

Considere que nosso problema de roteamento com K veículos esteja definido em um grafo G=(N_0 , E) onde $N_0 = N \cup \{0\}$ é o conjunto de vértices (como descrito no capítulo IV), $|N_0| = n+1$ e E é o conjunto de todas as arestas de G. Note que, se G=(N_0 , E) é um grafo qualquer, a determinação de uma solução viável para o problema do K-caixeiro viajante (isto é, um problema de roteamento de veículos sem as restrições de capacidade) já é NP-completo! Na verdade, isto ocorre mesmo para K=1, quando temos o problema do ciclo hamiltoniano (vide Garey e Johnson[79]). Assume-se geralmente na literatura que

G=(N_0 ,E) é um grafo completo (já que custos bastante altos podem ser associados às arestas inexistentes).

No problema de roteamento, temos que o somatório da demanda dos clientes em cada uma das K rotas não deve ultrapassar a capacidade de cada veículo. Note que, responder se existe ou não uma partição dos n clientes tal que a demanda em cada rota seja atendida é o mesmo que resolver o problema *bin-packing* (decisão), reconhecidamente NP-completo (Garey e Johnson [79]). Assim, a determinação de K rotas viáveis para o problema de roteamento define um problema NP-completo mesmo se G=(N_0 , E) é um grafo completo!

Dado um grafo não direcionado $G = (N_0, E)$ e $W \subseteq N_0$, denotamos E(W) o conjunto de todas as arestas com ambos os extremos em W, e por δ(W), o conjunto de todas as arestas com apenas uma extremidade em W. Dizemos que G(W)=(W, E(W)) é o grafo induzido por W (conjunto de vértices). De maneira análoga, se F é um conjunto de arestas de E, G(F)=(V(F), F) é um subgrafo induzido por F (conjunto de arestas) onde onde V(F) é um conjunto de vértices incidentes em pelo menos uma das arestas de F.

Dizemos que um subconjunto F de arestas em nosso problema de roteamento define uma *rota*, se o subgrafo induzido G(F) é um ciclo elementar contendo o depósito 0 (cada vértice de G(F) tem grau 2). Uma *k-rota,* é definida como um subconjunto R de arestas (rotas) de E de forma que $R_1,\ldots,R_k$ determine uma partição de R e cada um dos clientes de N apareça em apenas um dos subconjuntos $V(R_i)$, para i=1,...,k. Note, em nosso caso, que estamos interessados na determinação de uma k-rota de comprimento mínimo.

Para simplificar a notação (quando necessário), faremos a associação de uma variável x_e a cada uma das arestas de E. Estas variáveis representam o número de vezes que utilizamos uma aresta em uma k-rota (solução viável). Note que se permitimos a existência de rotas simples (com apenas um cliente), uma mesma aresta será utilizada 2 vezes. De maneira análoga, c_e representa o custo associado a cada uma das arestas **e** de E.

Para cada subconjunto U de arestas, definimos x(U) como a soma de todas as variáveis x_e associadas às arestas pertencentes a U.

Feita as devidas considerações, e assumindo a existência de rotas simples, podemos rescrever a seguinte formulação para o problema de roteamento de veículos (PRVRC). Todos os K veículos tem capacidade **b** e d(S) representa a demanda total de todos os clientes na partição S (como definido no capítulo anterior).

$$min \sum_{e \in E} c_e x_e \qquad (1)$$

$$s.a. \begin{cases} x(\delta(\{0\})) = 2K & (2) \\ x(\delta(\{i\})) = 2; \quad \forall\, i \in N & (3) \\ x(\delta(\{S\})) \geq 2\left\lceil \frac{d(S)}{b} \right\rceil; \quad \forall\, S \subseteq N \;\; e \;\; S \neq \varnothing & (4) \\ 0 \leq x_e \leq 1; \quad \forall\; e \in E(N) & (5) \\ 0 \leq x_e \leq 2; \quad \forall\; e \in \delta(\{0\}) & (6) \\ x_e \in \mathrm{Z}; & (7) \end{cases}$$

Seja P_{PRV} o conjunto viável definido pelas restrições (2)-(7) acima. Representamos por conv(P_{PRV}), a envoltória convexa dos pontos de P_{PRV}. Assim,

$$\mathrm{conv}(P_{PRV}) = \mathrm{conv}\left\{x \in \Re^m \;/\; x \; satisfaz \;\; (2)-(7)\right\}$$

onde m = n(n+1)/2 representa o número de variáveis presentes na formulação.

Esta formulação foi utilizada por Laporte et al.[85],[87]; Cornuejols e Harche[93]; Harche e Rinaldi[91]; Augerat [95].

Como discutido no capítulo anterior, as restrições de capacidade (4), também chamadas de restrições de sub-rotas generalizadas não definem facetas em geral. Campos et al. [91], fazem uma análise da estrutura poliédrica do problema de roteamento supondo que todos os clientes tenham mesma demanda. Neste trabalho, é demonstrado que as restrições de eliminação de sub-rotas definem facetas para conv(P_{PRV}) desde que algumas condições sobre as partições S e a capacidade de cada veículo sejam atendidas.

Veremos na seção seguinte duas versões mais fortes das restrições de eliminação de sub-rotas apresentadas em Cornuejos e Harche[93]:

V.3 - Restrições de Eliminação de Sub-rotas:

Seja S⊆N um subconjunto de clientes. Como discutido no capítulo anterior, definimos r(S) = $\lceil d(S)/b \rceil$, como sendo um limite inferior para o número mínimo de veículos que atendem os clientes de S. Considere agora, r'(S), o menor número de veículos necessários para atender a demanda de todos esses clientes. Em outras palavras, r'(S) é a solução de um problema *bin packing* com itens de capacidade **b**.

Note que as restrições de capacidade (4), podem ser substituídas pelas restrições:

$$x(\delta(S)) \geq 2r'(S); \qquad \forall\ S \subseteq N\ e\ S \neq \varnothing \qquad (8)$$

Apesar de mais forte (já que r'(S) ≥$\lceil d(S)/b \rceil$), a utilização de r'(S) não é interessante do ponto de vista prático. Como discutido em Augerat[95], quando o número de clientes de S é pequeno, r'(S) pode ser calculado de maneira exata (Martello e Toth[90a]). Para conjuntos maiores um limite inferior não trivial (substituindo r'(S)), pode ser calculado em tempo razoável (Martello e Toth[90b]).

Se o cálculo de r'(S) ou de outro limite inferior exigir um alto custo será mais conveniente a utilização de r(S) =$\lceil d(S)/b \rceil$.

Note que o conjunto de restrições (4) e (8) definem, de certo modo, "restrições locais" para o PRV. Elas não levam em consideração a demanda dos clientes que não estão na partição S. Assim, se nos preocurpamos apenas em atender os clientes de S com um número mínimo de veículos, poderemos gerar soluções inviáveis para o problema de roteamento de veículos.

Chamaremos $\pi = (P_1,...,P_K)$ uma *partição* dos vértices de N (conjunto de clientes) se cada um dos subconjuntos satisfizerem $d(P_j) \leq b;\ para\ 1 \leq j \leq K.$

Seja Π o conjunto de todas as K partições (como definidas acima). Para cada partição $\pi \in \Pi$ e S⊆N, seja $\beta_\pi(S)$ o número de subconjuntos desta partição contendo pelo menos um vértice de S. Ou seja:

$$\beta_\pi(S) = \left| j \in \{1,..,K\} / P_j \cap S \neq \varnothing \right|; \quad \forall\, S \subseteq N \; e \; \pi = (P_1,\ldots,P_k) \in \Pi .$$

Fazendo $R(S) = \min_{\pi \in \Pi} \left(\beta_\pi(S)\right)$, obtemos o seguinte conjunto de restrições de capacidade:

$$x(\delta(S)) \geq 2R(S), \qquad \forall\, S \subseteq N \; e \; S \neq \varnothing \qquad (9)$$

É fácil ver que: $\lceil d(S)/b \rceil \leq r'(S) \leq R(S), \quad \forall\, S \subseteq N.$ (10)

Uma partição $\pi' \in \Pi$, será *ideal,* relativa a um conjunto S⊂N se $R(S) = \beta_{\pi'}(S)$.

O exemplo da figura V.1 (apresentado em Cornuejols e Harche[93]), mostra uma situação onde temos 4 veículos de capacidade b=7. As demandas de cada cliente estão representadas ao lado de cada nó.

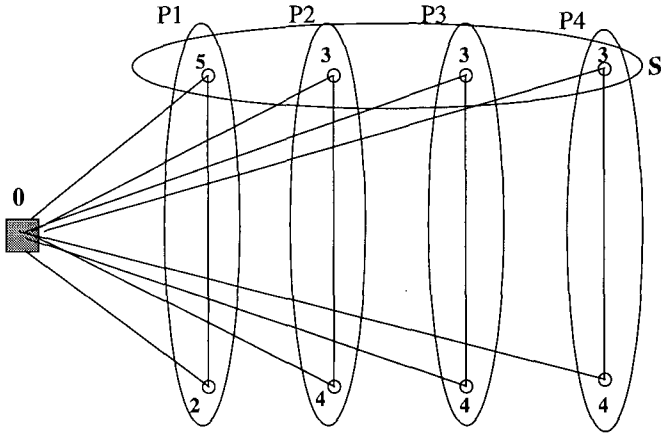


Figura V.1 : $\lceil d(S)/b \rceil < r'(S) < R(S), \quad \forall\, S \subseteq N.$

Observe no exemplo que $\lceil d(S)/b \rceil < r'(S) < R(S)$, para a partição S considerada (temos respectivamente 2 <3 < 4).

As restrições (9), podem ser consideradas "globais", já que levam em consideração todas as partições viáveis do conjunto de clientes. Entretanto, como ocorre com r'(S), a determinação de R(S) é bastante difícil (problema NP-completo).

O seguinte teorema (demonstrado em Cornuejols e Harche[93]), apresenta algumas condições necessárias para que o conjunto de restrições (9) definam facetas para o problema de roteamento de veículos (PRVRC).

Teorema V.1: Considere S⊆N, onde $2 \leq |S| \leq |N| - 1$ tal que, para cada tripla de clientes *i,j* e $l \in$N, exista uma K-partição ideal $\pi = (P_1,...,P_K)$ relativa a S (sendo *i,j* e $l \in P_1$). Então $x(\delta(S)) \geq 2R(S)$ define uma faceta para o problema de roteamento. •

Como discutido em Cornuejols e Harche[93], condições necessárias e suficientes para que as restrições (9) definam facetas, ainda não são conhecidas.

Dizer se nosso problema de roteamento admite ou não uma solução viável é um problema NP-completo, mesmo se trabalhamos em um grafo completo. Segue-se que a determinação da dimensão de nosso poliedro conv(P_{PRV}) também é NP-completo! Outra dificuldade surge em decorrência das igualdades (2) e (3) introduzidas na formulação do problema. A determinação de facetas do politopo associado se torna mais difícil já que não temos uma envoltória convexa conv(P_{PRV}) de dimensão plena.

Cornuejols e Harche[93], tratam de situações como esta fazendo uma inclusão de nosso poliedro em outro poliedro de dimensão plena que o contenha como uma face. Este outro poliedro é associado ao problema gráfico de roteamento de veículos (para maiores detalhes vide Cornuejols e Harche[93]). Esta abordagem foi utilizada com sucesso para o problema do caixeiro viajante. Cornuejols, Fonlupt e Naddef[85], definiram inicialmente a relaxação gráfica do problema do caixeiro viajante (*graphical tsp*), e descobriram novas famílias de desigualdades válidas para o problema do caixeiro viajante (como por exemplo as desigualdades *path, path-tree*).

V.5 - Desigualdades *Comb* (Pente):

As desigualdades *comb* foram introduzidas por Chvátal[73] para o problema do caixeiro viajante, sendo generalizadas posteriormente por Grötschel e Padberg[79].

Laporte e Nobert[84.a], foram os primeiros a adotar as desigualdades *comb* para

uma formulação do problema de roteamento de veículos. Estas desigualdades, consideram a demanda associada a cada um dos clientes, embora excluam o vértice 0 (depósito) da expressão que as define. Ao contrário, Cornuejols e Harche[93], apesar de não trabalharem com as demandas associadas aos clientes, utilizam variantes daquelas propostas por Laporte e Nobert[84.a] permitindo a inclusão do depósito.

Vejamos primeiramente a formulação apresentada por Cornuejols e Harche[93]:

Considere K≥2 o número de veículos utilizados. Seja $G=(N_0,E)$ um grafo completo e $H,T_1,T_2,..,T_s$, subconjuntos de N_0 satisfazendo as seguintes propriedades:

(i) $|T_i \setminus H| \geq 1$, *para* $i=1,...,s$;
(ii) $|T_i \cap H| \geq 1$, *para* $i=1,...,s$;
(iii) $|T_i \cap T_j| = 0$, *para* $1 \leq i \leq j \leq s$;
(iv) s é ímpar e s ≥ 3.

Segue então que:

$$x(E(H)) + \sum_{i=1}^{s} x(E(T_i)) \leq |H| + \sum_{i=1}^{s} |T_i| - \frac{3s+1}{2} + \alpha(K-1) \qquad (11)$$

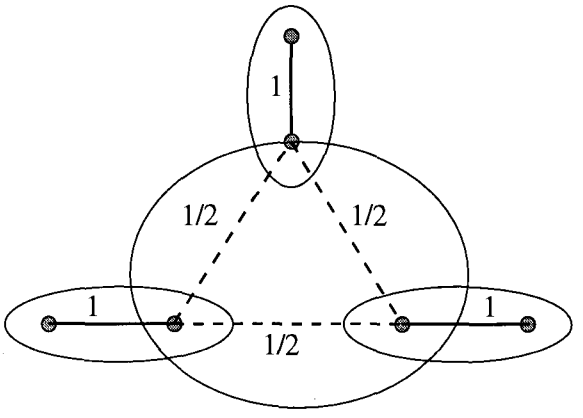
onde:

$$\alpha = \begin{cases} 0; & se \quad 0 \notin H \cup \left(\bigcup_{i=1}^{s} T_i \right) \\ 1; & se \quad 0 \in H \setminus \left(\bigcup_{i=1}^{s} T_i \right) \\ 2; & se \quad 0 \in H \cap T_j; \quad p/\ algum \quad j=1,..,s \end{cases}$$

define uma desigualdade válida para o problema de roteamento de veículos (PRVRC). O conjunto H será chamado *handle* e $T_1,...,T_s$ os dentes de H. Cornuejols e Harche[93], definem ainda outras expressões para as desigualdades *comb,* envolvendo, por exemplo, a interseção de vários dentes.

Na figura V.2, exibimos um suporte (conjunto $H \cup T_1 \cup ... \cup T_s$) a partir de uma relaxação linear para o problema de roteamento. Note neste caso, que podemos ter obviamente a presença de variáveis fracionárias:

Observe na expressão (11) que, se o depósito não pertence ao suporte $H \cup T_1 \cup ... \cup T_s$ teremos $\alpha = 0$, assim $3/2 + 3 > 3 + 6 - 5 \Rightarrow 9/2 > 4$ (restrição violada).



FiguraV.2: Suporte clássico de uma *comb* c/ 3 dentes (relax. linear)

Laporte e Nobert[84.a] e Araque[90] apresentaram formulações para as desigualdades *comb* envolvendo a demanda de cada cliente. Na formulação apresentada por Laporte e Nobert[84.a], o depósito não pertence ao suporte da *comb*. Como discutido em Augerat[95], a descrição proposta por Laporte e Nobert[84.a] pode ser escrita equivalentemente da seguinte forma:

$$x(\delta(H)) \geq (s+1) - \sum_{j=1}^{s} [x(\delta(T_j)) - 2r(T_j)] \qquad (12)$$

onde todos os dentes T_j (para j=1,..,s) devem satisfazer a $r(T_j / H) + r(T_j \cap H) > r(T_j)$.

Estudamos a identificação de *combs* violadas com mais detalhes no capítulo seguinte. Como veremos posteriormente, trabalharemos com a formulação utilizada por Cornuejols e Harche[93]. A idéia é utilizar uma expressão que permita uma "fácil" identificação das desigualdades *comb* a partir da solução do problema lagrangeano. Obviamente, a expressão (12) pode também ser utilizada.

Em nosso caso, ao contrário da relaxação linear, a determinação de *combs* violadas será mais simples já que trabalhamos apenas com soluções de coordenadas inteiras (K-árvores mínimas).

V.6 - Desigualdades *Multistars* (Estrelas):

Em Araque et al.[90], são apresentadas diversas restrições válidas para o problema de roteamento (PRVRC) quando todos os clientes possuem demanda unitária. Estas restrições entretanto, continuam válidas na maioria dos casos, mesmo quando os clientes possuem demandas distintas (Augerat[95]). O mesmo ocorre para as restrições *multistars,* apresentadas a seguir:

Seja **v** um vértice de N, e $\Omega = \{S_i / i = 1,..,s\}$ um conjunto de sub-conjuntos de N satisfazendo a seguinte condição:

$$\begin{aligned} &\text{i)} \quad S_i \cap S_j = \{v\}, && \forall\, i,j \in \{1,..,s\} \;\; e \;\; i \neq j \\ &\text{ii)} \quad d(S_i) \leq b; && \forall\, i \in \{1,..,s\} \\ &\text{iii)} \quad d(S_i \cup S_j) > b; && \forall\, i,j \in \{1,..,s\} \;\; e \;\; i \neq j \end{aligned}$$

Se as condições acima forem satisfeitas então:

$$\sum_{i=1}^{s} x(\delta(S_i)) \geq 4s - 2 \qquad (13$$

será válida para o problema de roteamento (PRVRC).

Para illustrar essa classe de restrições vejamos o exemplo da figura V.3, onde temos uma solução viável para o problema de roteamento com 2 veículos de capacidade 11. Os valores associados a cada nó representam as demandas de cada cliente.

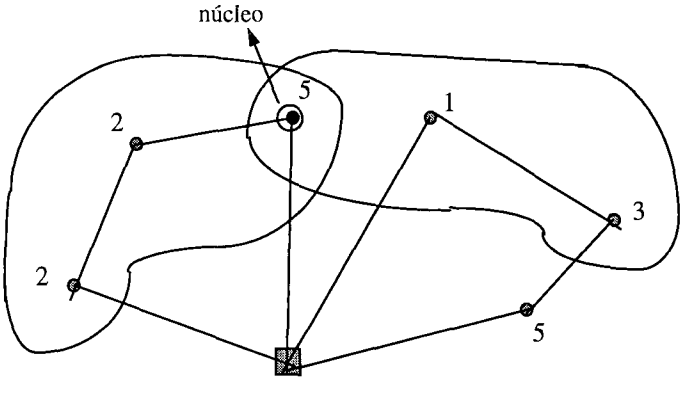


Figura V.3: Desigualdade válida para o PRVRC

É fácil ver que as condições (i), (ii), (iii) e a restrição (12) se verificam, ou seja: $6 \geq 8-2$.

Como veremos adiante (capítulo VI), estaremos interessados em situações onde *multistars* violadas sejam obtidas através de uma relaxação lagrangeana. Em nosso caso, K-árvore mínima com 2K arestas incidentes ao depósito (solução inteira). Araque et al.[94] e Augerat[95] trabalham na identificação de *multistars* violadas a partir de uma relaxação linear para o problema de roteamento de veículos.

Outros tipos de *multistars* são apresentadas em Araque et al.[90] e [94], Gouveia[95] e Letchtford e Eglese[96].

Capítulo VI

Identificação de Restrições Violadas

VI.1 - Introdução:

Neste capítulo, será discutida nossa abordagem que trata da identificação de restrições violadas a partir da solução de um problema lagrangeano, que consiste, basicamente, na determinação de uma K-árvore mínima c/ 2K arestas incidentes ao depósito. A identificação de restrições violadas neste caso, é significativamente mais simples se comparada à relaxação linear (onde temos soluções podendo assumir valores fracionários).

Não é conhecido até o momento, nenhum algoritmo exato polinomial para a separação de sub-rotas violadas por uma relaxação linear (associada a uma formulação do problema de roteamento). Como Harche e Rinaldi[91] mostraram que este problema de separação é NP-completo, sua resolução em tempo polinomial será possível se e somente se P=NP.

Trabalhamos na identificação de sub-rotas, *combs* e *multistars* violadas partindo sempre de uma K-árvore mínima c/ 2K arestas incidentes ao depósito (solução inteira). Como veremos a seguir, a cada iteração do método subgradiente, podemos resolver de maneira exata e em tempo polinomial o problema de identificação de sub-rotas violadas.

Questões relativas à implementação e estruturas de dados utilizadas por 3 classes de restrições serão também discutidas neste capítulo.

VI.2 - Geração de Restrições de Eliminação de Sub-rotas (outra abordagem):

Na abordagem alternativa apresentada aqui, não nos restringimos à geração de sub-rotas a partir de sementes (como proposto por Fisher[94.b]). Em nosso caso, a determinação de restrições violadas é realizada buscando sempre informações presentes na solução do problema lagrangeano corrente (K-árvore). Isto difere da proposta apresentada inicialmente por Fisher[94.b], que apresentava um conjunto de partições independentes sem se ater à evolução de cada problema lagrangeano obtido.

Uma alternativa interessante na geração de sub-rotas violadas é particionarmos nosso conjunto de clientes em **m** componentes conexas (onde **m** é um valor entre K e 2K). Para isso, bastará eliminarmos todas as arestas $\bar{x}_{0i}$ da K-árvore mínima (solução do problema lagrangeano) que ligam os clientes ao depósito.

Para cada uma das componentes conexas S_k (onde k = 1,...,m), construímos a restrição:

$$\sigma_k = 2r(S_k) - x(\delta(S_k)) \leq 0 \qquad (1)$$

onde $r(S_k) = \lceil d(S_k) / b \rceil$ e $x(\delta(S_k)) = \sum_{i \in S_k} \sum_{j \in \bar{S}_k} \bar{x}_{ij}$ (sendo $\bar{x}_{ij} = 1$ *para* $i \in S_k$ *e* $j \in \bar{S}_k$).

As variáveis binárias $\bar{x}_{ij}$ representam as arestas presentes na solução do problema lagrangeano.

Como estamos interessados apenas nas restrições que violem nossas restrições de capacidade devemos nos preocupar com partições onde temos $\sigma_k > 0$ na solução do problema lagrangeano (K-árvore). Desta forma, dualizamos apenas um subconjunto das **m** partições obtidas nesta iteração do método subgradiente! Como conseqüência do teorema

III.2 (capítulo III), apenas as restrições violadas "poderão" proporcionar um incremento estrito do limite inferior.

Para ilustrar a geração de restrições aqui proposta, vejamos o seguinte exemplo onde temos uma 3-árvore mínima com um vértice fixo (depósito) de grau 6 e 3 veículos de capacidade b=20. As demandas de cada cliente estão representadas ao lado de cada nó entre parênteses. A demanda da partição S_i (p/ i=1,2,3) é representada por d(S_i):

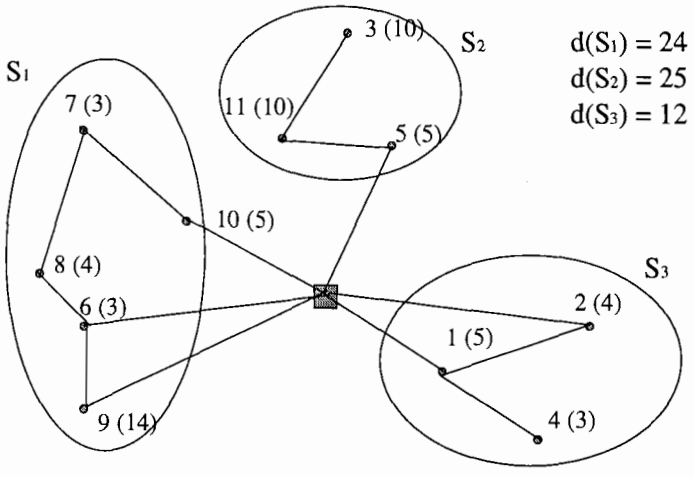


Figura VI.1: Determinação das partições Si

Note que $r(S_1)=2$, $r(S_2)=2$ *e* $r(S_3)=1$, são limites inferiores para o número mínimo de veículos que atende a cada subconjunto. Verificando as restrições violadas pela solução do problema lagrangeano para cada uma das partições S_k obtemos:

$$\sigma_1 = 2r(S_1) - \sum_{i\in S_1}\sum_{j\in\bar{S}_1}\bar{x}_{ij} = 4-3=1>0 \qquad (\textit{rest. violada})$$

$$\sigma_2 = 2r(S_2) - \sum_{i\in S_2}\sum_{j\in\bar{S}_2}\bar{x}_{ij} = 4-1=3>0 \qquad (\textit{rest. violada})$$

$$\sigma_3 = 2r(S_3) - \sum_{i\in S_3}\sum_{j\in\bar{S}_3}\bar{x}_{ij} = 2-2\le 0 \qquad \textit{OK}$$

Segue-se que as restrições: $\sum_{i \in S_1} \sum_{j \in \bar{S}_1} x_{ij} \geq 4$ *e* $\sum_{i \in S_2} \sum_{j \in \bar{S}_2} x_{ij} \geq 4$ devem ser dualizadas e adicionadas ao nosso conjunto de restrições ativas.

Temos então a seguinte heurística para determinação de sub-rotas violadas:

PROCEDIMENTO VI.1: {Heurística p/ determinação de sub-rotas violadas}

Início

- achou_rv = FALSO; {indica se achamos ou não rest. violadas}
- desconectamos todos os clientes do depósito;
  {obtemos m componentes conexas S_i (para i=1,..,m)}
- calculamos x(δ(S_i)), para i=1,..,m;
- **para** (i=1,..,m) **faça**
  - calcula $\sigma_i = 2\left\lceil \frac{d(S_i)}{b} \right\rceil - x(\delta(S_i))$;
  - **Se** ($\sigma_i > 0$) **então**
    - adiciona partição S_i ao conjunto ativo;
    - achou_rv = VERD;
  - **-fim;**
- **fim;**
- retorna achou_rv; {retorna falso ou verdadeiro}

fim.

Figura VI.2: Heurística p/ determinação de sub-rotas violadas.

Podemos constatar facilmente que, caso nenhuma restrição violada tenha sido encontrada ao final de nossa heurística, não teremos garantida a existência ou não de sub-rotas violadas. Para ilustrar essa situação, vejamos o exemplo da figura VI.3 onde apresentamos uma 2-árvore com 4 arestas incidentes ao depósito. Ao lado de cada nó estão representadas as demandas de cada cliente:

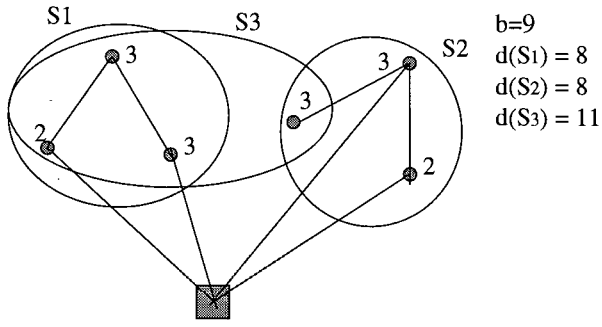


Figura VI.3: Identificação de sub-rotas violadas

Observe que, embora as partições S_1 e S_2 geradas por nossa heurística (componentes conexas) não sejam violadas, a partição S_3 é violada pois:

$$\sigma_3 = 2\left\lceil \frac{d(S_3)}{b} \right\rceil - x(\delta(S_3)) = 2.2 - 3 = 1 > 0 \;!!$$

Apresentamos a seguir um algoritmo exato polinomial na identificação de sub-rotas violadas. Este algoritmo retorna VERD ou FALSO conforme existam ou não sub-rotas violadas em nossa K-árvore.

PROCEDIMENTO VI.2: {alg. exato p/ identificação de sub-rotas violadas - versão 1}

Início

achou_rv := procedimento VI.1; {retorna VERD ou FALSO}

Se (não achou_rv) **então**

Se (existir alguma aresta c/ um extremo de grau 1) **então**

- adiciona vértices c/ grau 1 ao conjunto ativo (restrições dualizadas);
- achou_rv = VERD;

fim;

fim;

retorna achou_rv;

fim.

Figura VI.4: Determinação de sub-rotas violadas (alg. exato).

É fácil ver que cada vértice **v** com grau 1 em T_K determina uma partição $S_v = \{v\}$ violada c/ $\sigma_v = 1 > 0$ (já que $\left\lceil \frac{d(S_v)}{b} \right\rceil = 1$ *e* $x(\delta(S_v)) = 1$).

Mostraremos formalmente a seguir que o algoritmo VI.2 sempre resolve, de maneira exata, o problema da identificação de sub-rotas violadas em uma iteração qualquer do subgradiente, ou seja, se não existirem vértices com um extremo de grau 1, não existirão sub-rotas violadas. Vejamos inicialmente a seguinte notação:

Notação: Seja S_i (para i=1,..,m), cada uma das componentes conexas obtidas desconectando-se os clientes do depósito. Representaremos por $G_i(V_i, E(V_i))$ onde i=1,..,m os subgrafos induzidos por $V_i = S_i \cup \{0\}$. •

Lema VI.1: Se não existirem partições violadas ao final da heurística VI.1, teremos $d(S_i) \leq b$ para i=1,..,m.

Prova:

Suponha inicialmente que apenas uma aresta ligue uma componente conexa qualquer S_l ao depósito. Ou seja, $x(\delta(S_l)) = 1$ para algum $l \in \{1,..,m\}$. Como $\left\lceil \frac{d(S_i)}{b} \right\rceil \geq 1$, para i=1,..,m temos que $\sigma_l = 2.\left\lceil \frac{d(S_l)}{b} \right\rceil - 1 > 0$! O que é absurdo, pois não temos sub-rotas violadas ao final da heurística VI.1. Portanto:

$$x(\delta(S_i)) \geq 2 \quad p/\ i = 1,..,m \qquad (I)$$

Suponha (por absurdo) que $d(S_l) > b$ para algum $l \in \{1,..,m\}$. Teremos então $\left\lceil \frac{d(S_l)}{b} \right\rceil \geq 2$. Desta forma, o subgradiente associado a S_l será dado por $\sigma_S = 2.\left\lceil \frac{d(S_l)}{b} \right\rceil - x(\delta(S_l)) \leq 0$ (já que não temos restrições violadas ao final da heurística VI.1). Ou seja:

$$x(\delta(S_l)) \geq 4 \qquad (II)$$

Calculando o número de arestas para cada subgrafo G_i (como definido acima) e observando que cada componente conexa S_i tem pelo menos $|S_i|-1$ arestas, temos de (I) e (II) que $|E(V_i)| \geq |S_i|+1$ para i=1,..,m sendo i≠*l* e $|E(V_i)| \geq |S_i|+3$ para i=*l*. Ou seja, o número mínimo de arestas em nosso grafo G, será obtido fazendo-se:

$$\begin{aligned} x(\delta(S_i)) &= 2 \qquad p/ \quad i \neq l \\ x(\delta(S_i)) &= 4 \qquad p/ \quad i = l \end{aligned} \qquad (III)$$

É fácil ver que, como temos a componente ***l*** ligada ao depósito por 4 arestas, e as demais componentes ligadas ao depósito por 2 arestas, teremos m=K-1 componentes conexas obtidas ao final da nossa heurística.

Segue que o número mínimo N_{min} de arestas em G será dado por:

$$N_{min} = |E(V_1)|+..+|E(V_l)|+..+|E(V_m)| = |S_1|+1+..+|S_l|+3+..+|S_m|+1 \qquad (IV)$$

Como $\sum_{i=1}^{m} |S_i| = n$ e m=K-1, temos de (IV) que, N_{min} = n+m+2 = n+K+1, o que é absurdo pois nossa K árvore contém exatamente n+K arestas. Portanto, $d(S_i) \leq b$ para i=1,..,m. •

Vejamos agora o seguinte lema:

Lema VI.2: Se não existirem partições violadas após o término da heurística VI.1, teremos exatamente 2 arestas unindo cada componente conexa ao depósito. Além disso, cada uma dessas componentes define uma árvore.

Prova:

Caso não tenhamos partições violadas ao final da heurística VI.1, teremos $d(S_i) \leq b$ para i=1,..,m (lema anterior). Segue diretamente que $x(\delta(S_i)) \geq 2$, para i=1,..,m (já que $\sigma_i = 2 - x(\delta(S_i)) \leq 0$ qualquer que seja $i \in \{1,..,m\}$).

Suponha agora que para alguma componente S_l (onde $l \in \{1,..,m\}$) tenhamos $x(\delta(S_l)) \geq 3$. Teremos pelo menos $|S_i|-1$ arestas em S_i para i=1,..,m (já que cada S_i é conexa). Segue que $G_i(V_i, E(V_i))$ (subgrafo induzido por $V_i = S_i \cup \{0\}$), contém pelo menos $|S_i|+2$ arestas se i=l, e $|S_i|+1$ arestas se i≠l..

Se m componentes conexas forem geradas então:

$$\sum_{i=1}^{m} |E(V_i)| \geq |S_1| + 1 + ... + |S_i| + 2 + ... + |S_m| + 1$$

Como $\sum_{i=1}^{m} |S_i| = n$ e $\sum_{i=1}^{m} |E(V_i)| = n + k$, teremos n+K ≥ n+m+1. Ou seja m ≤ K-1.

Assim, m ≤ K-1 componentes conexas foram geradas. Note entretanto, que precisamos no mínimo, K partições (K veículos) para atender toda a demanda de maneira satisfatória. Logo, para alguma componente S_l (onde $l \in \{1,..,m\}$), teremos $d(S_l) > b$, o que é absurdo pois $d(S_i) \leq b$ para i=1,..,m.

Assim, qualquer que seja a componente conexa S_i obtida (para i=1,..,m) teremos $x(\delta(S_i)) = 2$.

De maneira análoga, mostramos que cada componente conexa S_i (p/ i=1,..,m) contém exatamente $|S_i| - 1$ arestas ao final da heurística VI.1. Suponha p/ absurdo, que alguma componente conexa S_l tenha pelo menos $|S_l|$ arestas. Note que, cada componente conexa S_i p/ i≠l, possui pelo menos $|S_i| - 1$ arestas. Como $x(\delta(S_i)) = 2$ p/ i=1,..,m e m=k, obtemos a seguinte desigualdade estrita:

$$\sum_{i=1}^{m} |E(V_i)| = \sum_{i=1}^{m} (|E(S_i)| + 2) > \sum_{i=1}^{m} (|S_i| - 1 + 2) = \sum_{i=1}^{m} |S_i| + m = n + K$$

Chegamos portanto a um absurdo já que $\sum_{i=1}^{m} |E(V_i)| = n + K$. Logo, cada componente conexa S_i define uma árvore. •

Teorema V1.1: Seja T_K o subgrafo gerado a partir da solução do problema lagrangeano. O algoritmo VI.2 sempre determina, caso exista, uma sub-rota violada.

Prova:

Suponha que nenhuma partição violada S_i seja obtida após o término da heurística VI.1. Mostraremos que, se não existir nenhuma aresta c/ uma extremidade com grau 1, então não existirão sub-rotas violadas em nossa K-árvore.

Note que, se ao término do algoritmo VI.2, sub-rotas violadas não forem encontradas, todos os vértices de G terão grau superior ou igual a 2.

Do lema anterior, temos exatamente 2 arestas incidentes ao depósito para cada subgrafo G_i obtido. Como cada componente conexa S_i define uma árvore, teremos a formação de um único ciclo C_i em G_i. Sejam $(0,v_1)$ e $(0,v_2)$ (onde $v_1 \neq v_2$), as arestas que unem S_i ao depósito. Obviamente teremos um único caminho em S_i que une v_1 a v_2.

Suponha que exista algum vértice neste caminho com grau superior ou igual a 3. É fácil ver neste caso que $\overline{V}_i = V_i \setminus C_i$ é não vazio. Como S_i é uma árvore (lema anterior) e $\overline{V}_i \subset S_i$ segue que $\overline{G}_i(\overline{V}_i, E(\overline{V}_i))$ define uma floresta onde cada árvore possui sua raiz no caminho de v_1 a v_2. Desta maneira, teremos vértices de V_i com um extremo de grau 1, o que é absurdo pois teríamos encontrado uma sub-rota violada (algoritmo VI.2).

Portanto, não teremos nenhum vértice (diferente do depósito) no ciclo C_i com grau diferente de 2. Assim, se para cada subgrafo G_i, as restrições de capacidade forem satisfeitas (lema VI.1) e todos os vértices em G_i tiverem grau 2, teremos uma solução viável para o PRV(problema de roteamento de veículos). Logo, não existirão sub-rotas violadas em nossa K-árvore (teorema IV.2.1). •

Uma nova versão mais sofisticada do algoritmo VI.2 pode ser apresentada. Neste caso, buscamos a geração de um número maior de sub-rotas violadas a cada iteração. Esta idéia busca combinar a heurística VI.1 com a heurística de Fisher (capítulo IV) na geração de sub-rotas violadas. Em sua heurística, Fisher constrói partições (utilizando a idéia de nó semente) sem se ater ao problema lagrangeano corrente. Em nosso caso, buscamos a construção de um "aninhamento" de partições, todas violadas e geradas a partir da K-árvore mínima T_K (solução do problema lagrangeano).

Consideramos inicialmente $D_K \subset T_K$, o conjunto de todas as arestas de T_K não adjacentes ao depósito. Definimos agora $\Phi = \{T \subset D_K \,/\, T \; e \; arvore \; e \; x(\delta(T)) = 1\}$. Observe que cada um dos subconjuntos T de Φ define uma sub-rota violada. A construção destas árvores é iniciada sempre a partir dos vértices de grau 1 em D_K. Na figura VI.5 apresentamos um exemplo particular da geração de sub-rotas:

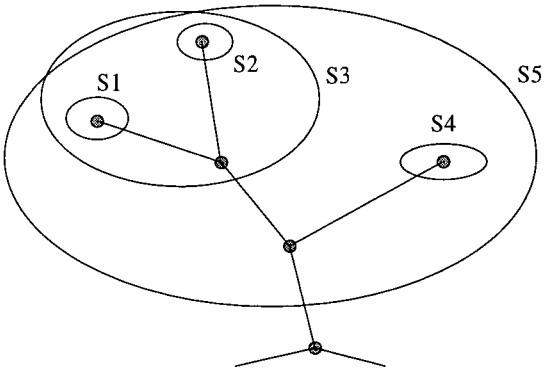


Figura VI.5: Aninhamento de sub-rotas

Note que cada partição S_i como gerada acima determina uma sub-rota violada, já que $x(\delta(S_i)) = 1$ e $\left\lceil \frac{d(S_i)}{b} \right\rceil \geq 1$. Desta forma, teremos sempre $\sigma_S = 2.\left\lceil \frac{d(S_i)}{b} \right\rceil - 1 > 0!$

Chegamos então ao seguinte algoritmo para a geração de sub-rotas "aninhadas":

PROCEDIMENTO VI.3: {Alg. exato p/ identificação de sub-rotas violadas - versão 2}

Início

- insere restrições violadas (no conjunto ativo) pela heurística VI.1;
- **Enquanto** (existir alguma aresta c/ um extremo de grau 1 em D_K) **faça**
  - D = D_K ;
  - **Enquanto** (D $\neq \varnothing$) **faça**
    - seleciona aresta (a, b) de D;
    - D = D - {(a, b)};
    - **Se** (grau(a) = 1 ou grau(b) = 1 (em D_K)) **então**
      - **Se** (grau(a) = 1) **então**
        - adiciona contração **a** ao conjunto ativo;
        - grau(b) = grau(b) - 1;

      senão
        - adiciona contração **b** ao conjunto ativo;
        - grau(a) = grau(a) - 1;

      fim se;
    - D_K = D_K - {(a, b)};

  fim se;

fim enq;

fim enq;

fim.

Figura VI.6 : Construção de sub-rotas violadas

Note que, a cada passo do "aninhamento" de sub-rotas fazemos a contração de uma aresta (a,b) em D_K. Se o grau de **a** for 1, adicionamos **a** e todos os vértices associados ao extremo **a** no conjunto ativo (conj. de restrições dualizadas). Em seguida, o extremo **a** desaparece e todos os vértices associados a **a** serão associados ao extremo **b**. Chegamos ao final do algoritmo quando não existirem mais vértices de grau 1 em D_K.

Observe que o algoritmo VI.3 também resolve, de maneira exata, o problema de identificação de sub-rotas. O "aninhamento" de sub-rotas é obtido sempre a partir de um vértice de grau 1!

Resultados computacionais satisfatórios serão apresentados no capítulo IX, onde mostramos a eficiência do algoritmo exato proposto.

VI.2.1- Implementação e estrutura de dados utilizada na geração de sub-rotas:

Como discutido no capítulo III, devemos estabelecer critérios que permitam atualizar dinamicamente o conjunto das restrições dualizadas.

No caso das restrições de eliminação de sub-rotas, se considerarmos o *lifting* introduzido por Fisher[94.b], temos que $\sigma_S = 2r(S) - \sum_{j=0}^{n} e_j \sum_{i \in S} x_{ij} \leq 0$, é a coordenada do vetor de subgradientes associada à partição S⊆N, onde $|S| \geq 2$ (capítulo IV). Assim, sempre que $\sigma_S > 0$, adicionamos a partição S ao conjunto ativo.

Em nossa abordagem, caso o multiplicador v_S (associado a S) seja nulo, retiramos a partição S do conjunto ativo. Teremos neste caso $\sigma_S \leq 0$, já que $v_S = max\{0, v_S + \theta\sigma_S\}$ é a formula de atualização dos multiplicadores. Desta forma, a partição S não irá mais contribuir para o incremento estrito do limite inferior (vide capítulo III). Cabe aqui ressaltar que, apenas as restrições ora violadas e um subconjunto das partições (violadas ou não) associadas a multiplicadores de Lagrange não nulos definem nosso conjunto ativo (conjunto de restrições dualizadas). Fisher[94.b], ao contrário, mantém restrições dualizadas (com multiplicadores nulos) por mais 3 iterações do subgradiente.

Como armazenar entretanto as partições associadas às restrições violadas? Além disso, como retirarmos as restrições associadas às partições que não pertencem mais ao conjunto ativo? Observe que, como temos uma atualização dinâmica das partições do conjunto ativo, será interessante utilizarmos uma estrutura de dados dinâmica que aloque e desaloque memória para estas partições quando necessário.

Para ilustrar a estrutura de dados utilizada vejamos a 2-árvore representada na figura VI.7. Suponha que tenhamos um veículo de capacidade b=12 e que as demandas de cada cliente estejam representadas ao lado de cada nó entre parênteses:

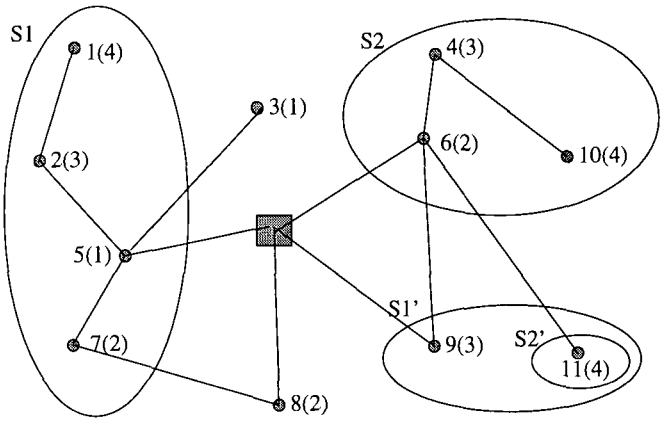


Figura VI.7: Representação das partições S1 e S2

A estrutura utilizada na representação das partições S1 e S2 é ilustrada na figura VI.8. Representamos também os conjuntos S1' e S2', necessários na implementação do *lifting* para as sub-rotas.

Note, na figura IV.8, que para cada partição S_k temos campos associados ao número de elementos $|S_k|$, demanda d(S_k), multiplicadores v_{S_k}, valor da restrição σ_k, um ponteiro para outra lista (indicando os elementos presentes nos conjuntos S_k e S_k') e finalmente, um ponteiro para a próxima partição S_{k+1}.

Como trabalhamos com o *lifting* introduzido por Fisher[94.b], temos associada a cada partição S_k uma outra partição $S_k' = \{j \in N \setminus S_k / j \geq 1 \ e \ d_j > br(S_k) - d(S_k)\}$. Assim, associamos o valor 1, a um vértice **v** se $v \in S_k$, e -1, se $v \in S_k'$. Se **v** não pertence a nenhum desses dois conjuntos associamos então o valor *zero*.

Note que $\sigma_2 = 2r(S_2) - \sum_{j=0}^{n} e_j \sum_{i \in S_2} x_{ij} = 2*1 - (x_{69} + x_{06}) = 0$, já que $e_j = 0$ para j$\in S_2'$ e $|S_2'| \leq 2$ (capítulo IV). Observe neste caso que j=11 pertence a S_2' e $|S_2'| = 1 \leq 2$, assim, $e_{11} x_{6,11} = 0$!!

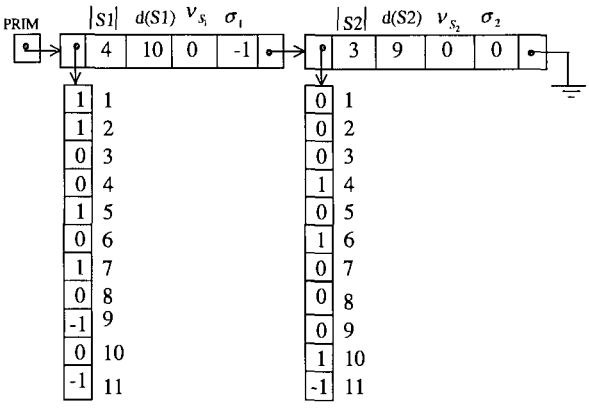


Figura VI.8: Construção das partições S_k

No cálculo do somatório $\sum_{j=0}^{n} e_j \sum_{i \in S} x_{ij}$, estaremos interessados em saber se uma determinada aresta (a,b) de T_K com um dos extremos em S pertence ao não a x(δ(S)). Desta forma, verificamos se a soma dos valores associados a **a** e **b** (no vetor de partições) é igual a 0 ou 1. Se esta soma é 2, as duas extremidades de (a,b) estão em S1 (ou S2). Se esta soma é menor ou igual a -1, nenhuma das extremidades de (a,b) pertence a S1 ou S2 respectivamente. Nestes 2 casos teremos (a,b)$\notin$ x(δ(S)). Caso esta soma seja 0, e **a** (ou **b**) pertença à partição analisada, teremos (a,b) com uma extremidade em S1 (ou S2) e outra extremidade em S1' (ou S2'). Este tipo de representação será importante na implementação do *lifting* proposto por Fisher[94.b].

Antes de adicionarmos uma restrição violada no conjunto ativo, devemos verificar se a partição correspondente já pertence ao conjunto ativo! Ao computarmos o número de elementos e a demanda total de cada partição, desenvolvemos um mecanismo que permite uma checagem rápida de partições já existentes. Note que, se uma partição S_k tem o mesmo número de elementos e a mesma demanda que uma partição S_p, teremos provavelmente duas partições idênticas. Somente neste caso, será necessário fazer uma comparação um a um entre os elementos (clientes) de S_p e S_k. Assim, dada uma partição S_p (com $\sigma_{S_p} > 0$), devemos percorrer o conjunto ativo comparando a partição S_p com todas as partições presentes no conjunto ativo (de tamanho pré-determinado M_1). Caso a partição S_p não

pertença ao conjunto ativo, adicionamos esta partição na posição apontada por PRIM (vide figura VI.8).

Ao incrementarmos o conjunto ativo, o número de contrações (partições) obtidas a cada iteração do método subgradiente é limitado por **n** (vide procedimento VI.3). Como o número de clientes de cada partição também é limitado por **n**, teremos uma complexidade total de $O(M_1 n^2)$ iterações no procedimento que adiciona sub-rotas ao conjunto ativo. Lembre-se que antes de inserir uma nova partição ao conjunto ativo devemos compará-la com as partições já existentes.

Observe que, se após a atualização dos multiplicadores (na expressão $v_{S_k} = max\{0, v_{S_k} + \theta\sigma_k\}$) tivermos ainda algum multiplicador nulo, a partição S_k associada deverá ser retirada do conjunto ativo. Para isso, percorremos nossa lista encadeada a partir da posição inicial apontada por PRIM (primeira partição do conjunto ativo).

VI.3 - Identificação de *combs* (pentes) violadas:

De maneira análoga à geração de sub-rotas generalizadas buscamos uma estratégia que identifique *combs* violadas através da solução do problema lagrangeano. Mostraremos que, para uma grande quantidade de estruturas especiais (em nossa K-árvore mínima), conseguimos identificar facilmente a *handle* e dentes *(teeth)* de uma *comb* violada.

Consideramos a situação (apresentada por Cornuejouls e Harche [93]) que, ao contrário de Laporte e Nobert[87], não trabalham com as demandas associadas a cada cliente. A utilização da formulação envolvendo demandas, apesar de interessante, desconsidera a presença do depósito. Ao escolhermos a formulação de Cornuejols e Harche [93], optamos por uma expressão que permitisse uma fácil identificação (a partir de T_K) e que redundasse obviamente em um pequeno esforço computacional. Testes computacionais entretanto, devem ser realizados comparando estas duas formulações.

Assim, devemos identificar uma *handle* H e dentes $T_1,...,T_s$ de maneira que a seguinte restrição seja violada:

$$x(E(H)) + \sum_{i=1}^{s} x(E(T_i)) \leq |H| + \sum_{i=1}^{s} |T_i| - \frac{3s+1}{2} + \alpha(K-1) \qquad (2)$$

onde: (i) $|T_i \setminus H| \geq 1$, *para* $i = 1,\ldots,s$;

(ii) $|T_i \cap H| \geq 1$, *para* $i = 1,\ldots,s$;

(iii) $|T_i \cap T_j| = 0$, *para* $1 \leq i \leq j \leq s$;

(iv) s é ímpar e s ≥ 3.

$$\alpha = \begin{cases} 0; & se \quad 0 \notin H \cup \left(\bigcup_{i=1}^{s} T_i \right) \\ 1; & se \quad 0 \in H \setminus \left(\bigcup_{i=1}^{s} T_i \right) \\ 2; & se \quad 0 \in H \cap T_j; \quad p/\ a\ \lg um \quad j = 1,..,s \end{cases}$$

Trataremos aqui, apenas das situações onde temos α = 1 e α = 0 respectivamente. Para facilitar a distinção dos 2 casos aqui discutidos, utilizaremos a seguinte definição:

Definição VI.1: Suponha que uma *comb* violada tenha sido construída a partir da solução do problema lagrangeano. Chamaremos esta *comb* de ***bigcomb***, se o depósito pertence à *handle* e não pertence a nenhum de seus dentes. Caso o depósito não pertença a *handle* (bem como a nenhum de seus dentes) esta *comb* será chamada ***smallcomb***. •

Consideramos ainda a seguinte definição:

Definição VI.2: Dada uma K-árvore T_K. Chamaremos de ***folha***, a cada uma das arestas de T_K, não adjacentes ao depósito (vértice 0) e com um extremo de grau *um*. •

Note ainda, a partir da definição VI.2 acima, que 2 folhas serão *disjuntas* se não tiverem um vértice em comum.

Como veremos a seguir, encontramos freqüentemente, estruturas especiais que permitem uma fácil identificação de *combs* violadas, a partir da solução do problema lagrangeano.

Suponha que nossa K-árvore mínima (c/ 2K arestas incidentes ao depósito) contenha pelo menos 3 folhas disjuntas entre si. A seguinte proposição garante a existência de uma *bigcomb* violada em nossa K-árvore mínima.

Teorema VI.2: (Condição suficiente p/ determinação de *combs* violadas)

Seja G(N_0, T_K) o grafo associado a uma solução do problema lagrangeano. Considere ainda Q, o conjunto de todas as folhas de T_K disjuntas entre si. Se tivermos $|Q| \geq 3$, teremos garantida a existência de uma *comb* violada em T_K.

Prova:

Seja $Q \subset T_K$, um subconjunto de arestas como definido acima. Nossa prova consiste em, utilizando o conjunto Q, construir uma *comb* a partir de T_K, e mostrar em seguida que ela é violada pela solução do problema lagrangeano.

Esta *comb* pode ser gerada tomando-se todos os vértices de G(No, T_K) com grau superior ou igual a 2, inserindo-os em seguida na *handle* H. Além disso, escolhemos 3 dentes de cardinalidade 2, cada um contendo uma aresta de Q.

Devemos mostrar agora que esta *comb* é realmente violada pela solução do problema lagrangeano. Para isto considere $\overline{Q}$, com cardinalidade n_i, o conjunto de todas as folhas de G(No, T_K) (vide figura VI.8). Note que as folhas de $\overline{Q}$ não são necessariamente disjuntas entre si. Assim, $Q \subseteq \overline{Q}$.

Tomando-se exatamente 3 arestas disjuntas de Q e lembrando que nossa K-árvore possui n+K arestas, temos que n+K-n_i, será o número de arestas com ambas as extremidades pertencentes a *handle* (as demais arestas pertencem a $\overline{Q}$). Assim, x(E(H)) = n+K-n_i e $|H| = n + 1 - n_i$.

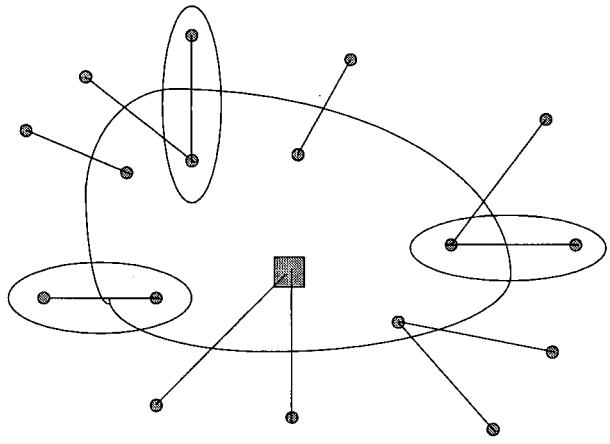


Figura VI.8: Construção de uma bigcomb

Fazendo-se s=3, e escolhendo 3 arestas de Q para construção dos dentes temos:

$$\sum_{i=1}^{3} x(E(T_i)) = 3 \qquad e \qquad \sum_{i=1}^{s} |T_i| = 6$$

Substituindo estes valores na expressão (2) e lembrando que α=1, temos:

$$\sigma_C = x(E(H)) + \sum_{i=1}^{s} x(E(T_i)) - |H| - \sum_{i=1}^{s} |T_i| + \frac{3s+1}{2} - \alpha(K-1) \qquad (3)$$

$$\sigma_C = \text{n+K-}n_i\text{+3-n-1+}n_i\text{-6 +5-K+1} = 2 > 0.$$

Como σ_C>0, temos uma *comb* violada pela solução do problema lagrangeano. •

Note que teremos garantida a existência de uma *comb* violada sempre que 3 folhas disjuntas forem encontradas em T_K. Uma alternativa interessante, será buscarmos um aumento na violação de cada coordenada σ_C do vetor subgradiente. Uma estratégia neste caso, será tentarmos a introdução de dentes grandes com o maior número possível de arestas, aumentando a violação de σ_C. É fácil observar em (3) que, se tomarmos dentes T com cardinalidade $|T|$ (maior ou igual a 3), e um número de arestas maior ou igual a $|T|$ teremos um aumento na violação de σ_C. Construímos dentes grandes facilmente

analisando-se as componentes conexas obtidas após desconectarmos o depósito dos clientes. Cada componente conexa T_i (onde i=1,..,m) terá pelo menos $|T_i|-1$ arestas. Caso tenhamos $n_a \geq |T_i|$ (onde n_a é o número de arestas na componente), toda componente conexa analisada poderá ser tomada como um dente grande de nossa *comb*. Se selecionarmos uma *comb* com apenas 3 dentes será interessante escolhermos inicialmente aquelas componentes cuja diferença $n_a - |T_i|$ seja máxima! Ou seja, escolhemos aqueles dentes que geram maior violação.

Resumindo os passos descritos acima, vejamos o seguinte algoritmo onde construímos uma *bigcomb* com 3 dentes, introduzindo, dentes grandes, sempre que possível:

PROCEDIMENTO VI.4: Heurística *Bigcomb*;

Início

- Construímos Q, conjunto de todas as folhas de T_K disjuntas entre si;
- **Se** $|Q| \geq 3$ **então**
  - Adicionamos à *handle* H, todos os vértices de grau maior ou igual a 2;
  - Construímos $P \subset Q$, subconjunto de todas as arestas de Q pertencentes a componentes conexas distintas;
  - **Se** $(|P| \leq 2)$ **então**
    -Construímos 3 dentes T_1, T_2, T_3 respectivamente, selecionando apenas 3 arestas de Q; {não determinamos dentes grandes}

  senão
    - Calculamos $n_c = n_a - |S|$ para cada componente conexa S obtida (n_a é o núm. de arestas de cada componente)
    - Ordenamos os n_c's em ordem decrescente de tamanho;
    - **Para** i = 1 **até** 3 **faça**
      - Selecionamos componentes conexas S a partir do maior n_c;
      - **Se** $n_c \geq 0$ **então**

- a comp. conexa S obtida determina um dente grande;

senão

- seleciona uma aresta de S e constrói dente pequeno;

fim para;

fim se-senão;

Se (*comb* gerada não pertence ao conj. ativo) **então**

- adiciona *comb* ao conj. ativo;

fim se;

fim.

Figura VI.9: Construção de *bigcombs*

Note no algoritmo VI.4 que teremos a introdução de dentes grandes apenas se tivermos 3 arestas de Q pertencentes a componentes conexas distintas (conjunto P)! Para entendermos melhor a utilização do conjunto P, considere por exemplo (em uma situação extrema) que todas as arestas de Q pertençam a uma mesma componente conexa S. Caso tenhamos $n_a - |S| \geq 0$, toda a componente seria selecionada e teríamos então uma *comb* com apenas um dente, contrariando a condição (iv) apresentada na definição de *comb*. Se $|P| \geq 3$, a condição (iv) nunca é violada. Assim, podemos gerar componentes conexas S (dentes grandes) sempre que $n_a - |S| \geq 0$!

Para tentarmos um incremento maior no limite inferior obtido, seria interessante buscarmos a construção de um número maior de *combs* violadas a cada iteração. Uma alternativa interessante, é gerarmos *combs* adicionais pesquisando-se apenas as componentes conexas obtidas. Seja $D = \{(0,i) \in T_K \ / \ i \in N\}$. Vejamos então o seguinte algoritmo:

PROCEDIMENTO VI.5: Heurística VCOMB;

Início

- Constrói (se possível) *bigcomb* em G(No, T_K);

- **Se** (existe *bigcomb* violada) **então**

- Constrói G'(N, T_K\D); {obtemos m componentes conexas}

-Constrói (se possível) *smallcombs* (*2-matchings*) em cada uma das m componentes;

fim se;

Fim.

Figura VI.10: Heurística VCOMB

Para ilustrar a geração de *bigcombs* com dentes grandes e *smallcombs*, vejamos o seguinte exemplo onde temos uma 2-árvore com 3 componentes conexas distintas:

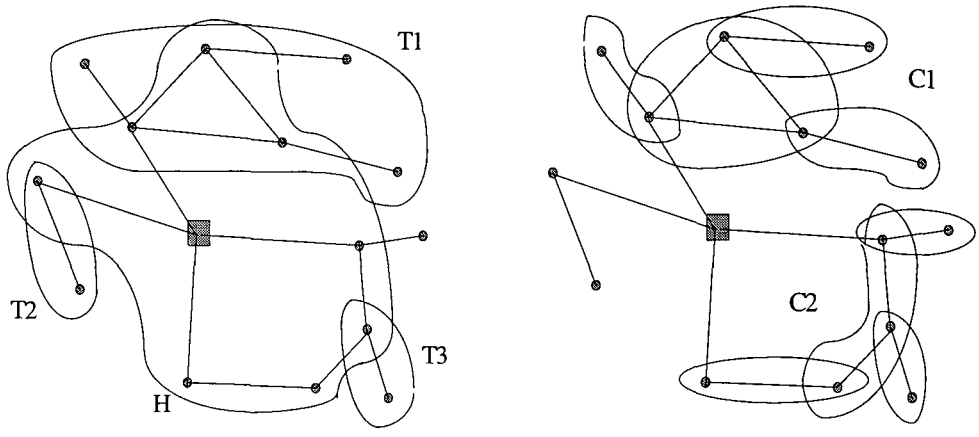


Figura VI.11: Determinação de *bigcombs* e *smallcombs*

A determinação das *smallcombs* é feita de maneira análoga à determinação das *bigcombs*. Para reduzir o esforço computacional entretanto, nos preocupamos apenas com a introdução das *2-matchings* (*combs* onde cada dente possue apenas uma aresta) em cada componente conexa gerada. Isto não descarta obviamente, a implementação de outras alternativas.

Apresentamos a seguir o procedimento utilizado na construção das *smallcombs*:

PROCEDIMENTO VI.6: Heurística *Smallcombs*;

Início

- **para** (cada uma das componentes conexas) **faça**

- construímos Q', conj. de todas as folhas (da componente) disjuntas entre si;
- **Se** $|Q'| \geq 3$ **então**

-Construímos 3 dentes T_1, T_2, T_3 respectivamente, selecionando apenas arestas de Q';

- Adicionamos a H, todos os vért. (da componente) c/ grau maior ou igual a 2;
- **Se** (*comb* violada não pertence ao conj. ativo) **então**
  - adiciona *comb* ao conjunto ativo;

fim;

fim;

fim.

Figura VI.12: Construção de *smallcombs*

Note neste caso que, como estamos trabalhando apenas com a identificação de 2-*matchings*, não será necessário a utilização do conjunto P⊆Q! Resultados computacionais são apresentados no capítulo VIII, avaliando os limites inferiores obtidos após a introdução das desigualdades *comb* (além das restrições de eliminação de sub-rotas) em nosso conjunto ativo.

VI.3.1- Implementação e estrutura de dados utilizada na geração das *combs:*

Fazemos a seguir algumas considerações sobre a estrutura de dados e a implementação realizada na determinação das *combs* violadas: Consideremos o exemplo da figura VI.13 onde temos representada uma 2-árvore com 4 arestas incidentes ao depósito:

Note que, como a componente conexa associada aos vértices $T_1 = \{1,9,4,6,8,7\}$ tem 6 arestas (ou seja, $|T_1| - |E(T_1)| \geq 0$) todos os vértices de T_1 serão utilizados na construção de um dente grande. A representação desta *comb* violada é ilustrada na figura VI.14.

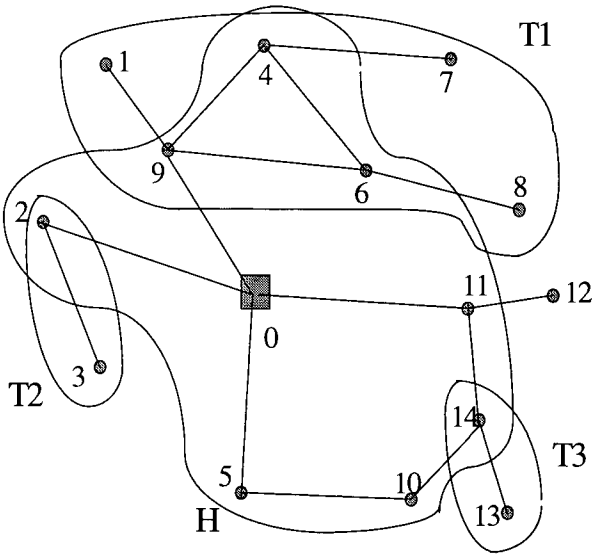


Figura VI.13 : Determinação de *bigcombs*

Temos um vetor de 0's e 1's representando os vértices pertencentes à *handle* H. Um outro vetor é utilizado para a representação dos dentes T_1, T_2 e T_3 respectivamente. O módulo dos valores negativos indicam o número de elementos presentes em cada dente.

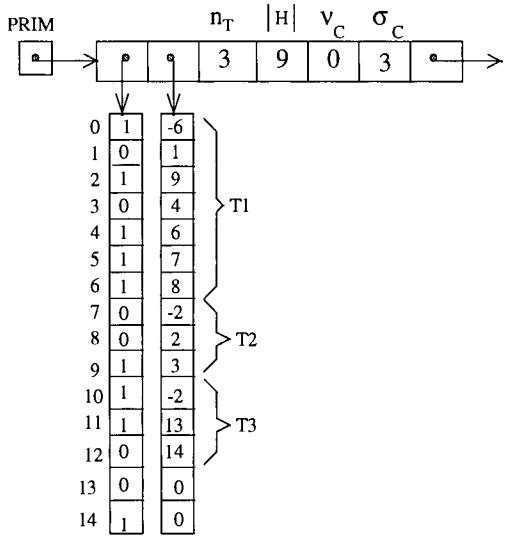


Figura VI.14: Representação do conjunto ativo (*combs*)

Antes de adicionarmos uma *comb* violada ao conjunto de restrições dualizadas (conjunto ativo A), devemos verificar se esta *comb* já pertence a A. Caso a *comb* analisada

possua o mesmo número de dentes e a mesma cardinalidade $|H|$ que alguma das *combs* presentes no conjunto ativo, fazemos a comparação um a um dos elementos presentes na *handle e dentes* dessas duas *combs*. Somente após esta checagem, decidimos se a *comb* violada será ou não adicionada ao conjunto ativo.

VI.4 - Identificação de *multistars* (estrelas) violadas:

Trabalharemos agora na identificação das *multistars* apresentadas no capítulo anterior a partir da solução do problema lagrangeano (K-árvore mínima c/ depósito de grau 2K). Assim, devemos determinar subconjuntos S_i (onde i=1,..,s) e um vértice **v** (núcleo da estrela) de forma que a seguinte restrição seja violada:

$$\sum_{i=1}^{s} x(\delta(S_i)) \geq 4s - 2 \qquad (4)$$

onde: i) $S_i \cap S_j = \{v\}$ $\quad \forall\, i, j \in \{1,..,s\}$ *e* $i \neq j$

ii) $d(S_i) \leq b;$ $\quad \forall\, i \in \{1,..,s\}$

iii) $d(S_i \cup S_j) > b;$ $\quad \forall\, i, j \in \{1,..,s\}$ *e* $i \neq j$

Adicionamos a restrição (4) no conjunto ativo caso $\sigma_S = 4s - 2 - \sum_{i=1}^{s} x(\delta(S_i)) > 0.$

Antes de descrevermos um algoritmo para as *multistars* violadas, vejamos inicialmente algumas definições:

Definição VI.2: Chamaremos de *hélice (ou ponta)* a cada um dos subconjuntos S_i de nossa "estrela". ●

Seja $G(N_0, T_K)$, o grafo associado a nossa K-árvore mínima com depósito de grau 2K (solução do problema lagrangeano). Seja ainda $D_K \subset T_K$, o conjunto de todas as arestas de T_K não adjacentes ao depósito. De maneira análoga à geração de sub-rotas violadas

construímos $\Phi = \{T \subset D_K \,/\, T \text{ e } \textit{arvore} \text{ e } x(\delta(T)) = 1\}$. Considere ainda, $\overline{\Phi} = \{T \in \Phi \,/\, d(T) \leq b\}$. Note que todos os subconjuntos de $\overline{\Phi}$ atendem a condição (ii) acima.

Definição VI.3: Diremos que T_B pertencente a $\overline{\Phi}$ é *hélice base,* se $d(T_B) \geq d(T)$, $\forall\, T \in \overline{\Phi}$. •

Definição VI.4: O conjunto $\Phi_K = \{T \in \overline{\Phi} \,/\, T \not\subset T_B\}$ será chamado *filtro* de Φ. •

Como veremos, se $|\Phi_K| \geq 2$, podemos utilizar cada um dos subconjuntos de Φ_K para a formação das hélices (ou pontas) das *multistars*. A idéia é combinar a hélice base T_B obtida, com as outras sub-árvores T de Φ_K (já filtradas). Se $d(T_B \cup T_i) > b$, (onde i ≠ B e $i \in \{1,..,|\Phi_K|\}$) a sub-árvore T_i poderá ser utilizada na formação de uma ponta de nossa estrela. Para isto, basta verificar se $d(T_i) + d(\{v\}) \leq b$, onde **v** (núcleo da estrela), pertence a T_B. Se apenas T_B e T_i forem geradas, teremos uma estrela de 2 pontas (hélices). Neste caso, $S_1 = T_B$ e $S_2 = T_i \cup \{v\}$ onde $i \in \{1,..,|\Phi_K|\}$, i ≠ B e v∈ T_B. A geração de outra ponta pode ser feita de maneira análoga retirando-se T_B e T_i de Φ_K e combinando novamente cada uma das partições T de Φ_K, com as partições T_B e T_i.

A proposição seguinte, estabelece um critério para a escolha do núcleo v∈ T_B, sempre garantindo a violação das *multistars* como definida anteriormente.

Teorema VI.3: Suponha que as pontas $S_1, S_2, .., S_s$ de uma estrela E violada, sejam geradas a partir do filtro Φ_K (onde s≥2) e satisfaçam as propriedades (i), (ii) e (iii). Se $S_1 = T_B$ e o vértice **v** (pertencente a T_B), é núcleo de E, então $grau(v) \leq 3$, $\forall\ s \geq 2$.

Prova:

Para mostrar nossa afirmação, basta notar que $x(\delta(S_1)) = 1$ (já que $S_1 = T_B$) e $x(\delta(S_i))$ = grau(v) + 1, onde v ∈ T_B e i=2,..,s. Segue então que:

$$\sum_{i=1}^{s} x(\delta(S_i)) = x(\delta(S_1)) + \sum_{i=2}^{s} x(\delta(S_i)) = 1 + (s-1)(grau(v)+1)$$

Como a restrição (4) é violada temos 1+(s-1)grau(v)+(s-1) < 4s-2. Assim, $grau(v) < \frac{3s-2}{s-1}$. Podemos mostrar facilmente por indução em **s** que: $3 < \frac{3s-2}{s-1} \leq 4$, $\forall\ s \geq 2$. Como grau(v) é inteiro e $grau(v) < \frac{3s-2}{s-1}$, temos que grau(v) ≤ 3, qualquer que seja o número de pontas de nossa estrela. •

Para ilustrar a geração de uma estrela com 3 pontas vejamos o exemplo da figura VI.16 onde temos uma 2-árvore com 2 veículos de capacidade b=32. Os valores ao lado de cada nó (entre parênteses) representam as demandas de cada cliente:

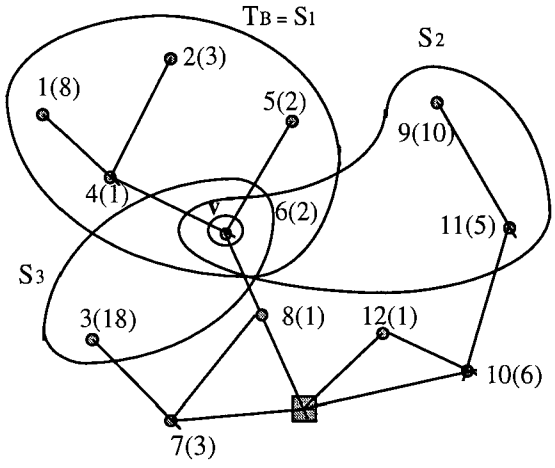


Figura VI.16: Estrela com 3 pontas

Observe que a estrela é violada já que $\sigma_S = 4.3 - 2 - 9 = 1 > 0$!

Resumindo os passos descritos anteriormente, apresentamos o seguinte algoritmo utilizado na geração de estrelas com 2 pontas:

PROCEDIMENTO VI.7: Heurística Estrela2:

Início

- Construímos: $\Phi = \{T \subset D_K \,/\, T\ e'\ arvore\ \ e\ \ x(\delta(T)) = 1\}$;

- Construímos: $\overline{\Phi} = \{T \in \Phi / d(T) \leq b\}$;

- Selecionamos $T_B \in \overline{\Phi}$ tal que demanda(T_B) $\geq$ demanda(T_i), $\forall T_i \in \overline{\Phi}$;

- Construímos: $\Phi_K = \left\{T \in \overline{\Phi} / T \not\subset T_B\right\}$;

- **Para** i=1 **até** n **faça**

- **Se** ((grau(i) $\leq$ 3) e ($i \in T_B$)) **então** {i é núcleo}

- **Para** (cada uma das sub-árvores T_j de $\Phi_K \setminus T_B$) **faça**

-**Se** $\left(d(T_B)+d(T_j) > b\right)$ *e* $\left(d(T_j)+d(\{v\}) \leq b\right)$ **então**

- determinamos estrela $E(T_B, T_j, i)$;

Se (E $\notin$ conj. ativo) **faça**

- adiciona estrela $E(T_b, T_j, i)$ ao conjunto ativo;

fim se;

fim para;

fim se;

fim para;

fim.

Figura VI.17 : Geração de *multistars*

Um grande número de estrelas pode ser gerado já que, todos os vértices **i** de T_B com grau menor ou igual a 3 poderão ser utilizados para a construção de estrelas de 2 pontas. O algoritmo da figura IV.17 pode ser utilizado ainda como base para a geração de estrelas com 3 ou mais pontas.

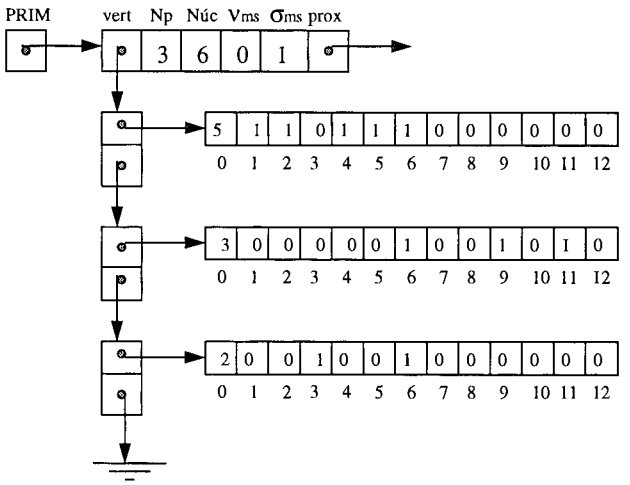
Como o número de possibilidades (e alternativas) é bastante grande na determinação das *multistars*, optamos por uma solução de baixo custo computacional e que se restringisse apenas na geração de estrelas de 2 pontas, utilizando, como núcleo, apenas vértices de grau 1. Pode-se constatar facilmente, observando a expressão (4), que núcleos de grau 1 permitem a geração de estrelas mais violadas, quando comparadas às estrelas com núcleos de grau 2 e 3 respectivamente.

A determinação de uma estratégia na geração das restrições violadas deve buscar sempre um equilíbrio entre o grau de violação das restrições (e consequentemente a qualidade dos limites inferiores obtidos) e o esforço computacional exigido para obtê-las.

VI.4.1- Implementação e estrutura de dados utilizada na geração das *mutistars:*

Apresentamos a seguir a estrutura utilizada na representação das *multistars* violadas.

Vejamos o exemplo da figura VI.16, onde temos uma estrela com 3 pontas. A estrutura de dados associada está representada na figura VI.18:



FiguraVI.18: Estrutura utilizada na representação de uma *multistar*

Os elementos pertencentes ou não a uma ponta S_k, são representados por 1's e 0's respectivamente. Todas as *multistars* violadas e adicionadas ao conjunto ativo deverão ser inseridas na posição apontada por PRIM. Os valores N_p e *Nuc* representam, respectivamente, o número de pontas e o núcleo de cada estrela. Os valores v_{ms} *e* σ_{ms} representam o multiplicador de Lagrange e a componente associada ao vetor subgradiente.

Analogamente à inserção de sub-rotas e *combs*, devemos verificar se a *multistar* violada já pertence ao conjunto ativo. Para incrementar a velocidade de busca observamos

apenas o número de pontas e o núcleo de cada estrela. Caso 2 estrelas tenham o mesmo número de pontas e o mesmo núcleo comparamos então os demais vértices presentes em cada umas de suas pontas. O número de elementos de cada ponta é representado na posição 0 de cada vetor.

Note que esta estrutura é interessante já que possibilita a geração de uma estrela com um número variável de pontas.

Capítulo VII

Fixação de Variáveis

VII.1 - Introdução:

Neste capítulo, estudaremos a utilização de critérios que permitam fixar variáveis x_{ij} em *zero* ou *um,* conforme a aresta (i,j) apareça ou não na solução ótima do problema original. Como veremos, este procedimento poderá ser realizado interiormente ao método subgradiente.

Todas as variáveis presentes na solução ótima e fixadas em *um* (em uma iteração qualquer do subgradiente), devem aparecer em cada uma das K-árvores mínimas e soluções heurísticas geradas posteriormente. Da mesma forma, as variáveis fixadas em *zero,* devem ser descartadas definitivamente e não figurar mais nas soluções parciais posteriores do subgradiente. Como veremos, a fixação de variáveis poderá permitir uma diminuição no tempo de processamento bem como uma melhoria na qualidade dos limites inferiores e superiores gerados.

Apresentamos primeiramente algumas considerações e definições iniciais e, em seguida, veremos como fixar variáveis (arestas) em *zero* e *um* respectivamente. Na última seção discutimos como "manipular" a estrutura de dados para a armazenagem das variáveis já fixadas.

VII.2 - Considerações Iniciais

Vejamos que condições devem ser observadas para que uma variável seja fixada em *zero* e *um* respectivamente. Apresentamos inicialmente a notação utilizada:

Notação (a): Analogamente ao capítulo IV (seção IV.3), representaremos por P(k), o problema de se encontrar uma K-árvore mínima com depósito de grau k. •

Notação (b): Representaremos por P(k, *f*), o problema de se encontrar uma K-árvore mínima $\overline{T}_K$ com um depósito de grau k e *f* pertencente a $\overline{T}_K$. •

Notação (c): Representaremos por Q(k,*e*), ao problema de se encontrar uma K-árvore mínima $\overline{T}_K$ com depósito de grau k e *e* não pertencente a $\overline{T}_K$. •

Considere uma K-árvore T_K solução de P(2K) e z_{UB} o valor da menor solução heurística gerada em uma iteração qualquer do subgradiente.

Uma variável (ou aresta) *f* não pertencente a T_K poderá ser fixada em *zero,* se o valor da solução associada a P(2K, *f*) for superior a z_{UB}. Note que, se uma K-árvore T_K é solução do problema P(2K) e *f* pertence a T_K, então P(2K) e P(2K, *f*) possuem a mesma solução. Estaremos interessados entretanto, nos casos onde *f* não pertence a T_K. Assim, para determinarmos uma solução de P(2K,*f*) a partir de P(2K), devemos pesquisar todas as arestas ainda não fixadas em *zero* e não pertencentes a T_K.

Analogamente, uma variável (ou aresta) *e* pertencente a T_K, poderá ser fixada em *um,* se o valor da solução associada a Q(2K,*e*) for superior a z_{UB}. Assim, se uma K-árvore T_K é solução do problema P(2K) e *e* não pertence a T_K então P(2K) e Q(2K,*e*) possuem mesma solução. Estaremos interessados entretanto, nos casos onde *e* pertence a T_K. Da mesma forma, para determinarmos uma solução de Q(2K,*e*) a partir de P(2K), devemos pesquisar todas as arestas ainda não fixadas em *um* e pertencentes a T_K.

Considere uma K-árvore T_K, uma aresta $e \in T_K$ e $e' \notin T_K$. Como já discutido na seção IV.3, o par de arestas (e,e') define uma *troca admissível* em T_K, se $T_K - e \cup \{e'\}$ for uma K-árvore. Esta troca será *admissível satisfatória* se $c(e) > c(e')$.

Rescrevemos a seguir 2 lemas de Fisher[94.b] (demonstrados no capítulo IV), que nos ajudarão como ferramentas na determinação de alguns dos critérios utilizados na fixação de variáveis.

Lema VII.1: Dados duas K-árvores distintas T_K e T_K', existirá uma função biunívoca h: $T_K - T_K' \rightarrow T_K' - T_K$ gerando pares (p,q) de trocas admissíveis em T_K (onde $p \in T_K - T_K'$ e $q \in T_K' - T_K$). •

Lema VII.2: Consideremos e, $f \in T_K$; e', f ' $\notin T_K$ arestas adjacentes ou não ao depósito. Se pelo menos um dos pares (e,e') ou (f, f') não define uma troca admissível em T_K então (e, f') e (f, e') darão (cada um) uma troca admissível em T_K se e somente se $T_K' = T_K - e - f \cup \{e', f'\}$ é uma K-árvore. •

Vejamos primeiramente como fixar em *zero* uma variável (ou aresta) não pertencente a T_K:

VII.3 - Fixando variáveis em *zero:*

Nesta seção, faremos distinção entre as arestas não pertencentes a uma K-árvore T_K (solução de P(2K)). Se estas arestas forem não-incidentes ao depósito, elas serão representadas por $\bar{f}$, caso contrário, serão representadas por $\bar{f}_0$. Assim, denotaremos por $P(k, \bar{f})$, ao problema de se encontrar uma K-árvore mínima $\overline{T}_K$ com depósito de grau k e $\bar{f}$ (não adjacente ao depósito) pertencente a $\overline{T}_K$. Da mesma forma, representaremos por $P(k, \bar{f}_0)$, ao problema de se encontrar uma K-árvore mínima $\overline{T}_K$ com depósito de grau k e $\bar{f}_0$ (incidente ao depósito) pertencente a $\overline{T}_K$.

Apresentamos a seguir as condições de otimalidade que devem ser satisfeitas por uma solução de P(k, $\bar{f}$). Sem perda de generalidade, consideramos k=2K. Como veremos, estas condições são muito semelhantes às condições de otimalidade apresentadas para P(2K) no teorema IV.4.

Teorema VII.1: (Condições de Otimalidade)

Uma K-árvore $\bar{T}_K$ com grau 2K no depósito será ótima para o problema P(2K, $\bar{f}$) se e somente se satisfizer as seguintes condições de otimalidade:

i) Sejam e_1 e f_1 duas arestas não incidentes ao depósito. Não existem trocas admissíveis satisfatórias (e_1, f_1) (onde $e_1 \in \bar{T}_K$, $e_1 \neq \bar{f}$ e $f_1 \notin \bar{T}_K$).

ii) Não haverá trocas admissíveis satisfatórias (e_1^0, f_1^0) onde $e_1^0 \in \bar{T}_k$ e $f_1^0 \notin \bar{T}_K$ sendo e_1^0 e f_1^0 incidentes ao depósito.

iii) Não haverá duas trocas admissíveis satisfatórias (e_1^0, f_1) e (e_2, f_2^0) com $e_1^0, e_2 \in \bar{T}_K$, $f_1, f_2^0 \notin \bar{T}_K$, sendo e_1^0, f_2^0 incidentes no depósito; f_1, e_2 não incidentes ao depósito e $e_2 \neq \bar{f}$.

Demonstração:

($\rightarrow$) Consideremos inicialmente $\bar{T}_K$, uma K-árvore ótima para o problema $P(2K, \bar{f})$. Assim $c(\bar{T}_K) < c(T_K')$, qualquer que seja a K-árvore T_K' com 2K arestas incidentes ao depósito e $\bar{f}$ pertencente a T_K'. É fácil ver neste caso que, se (i), (ii) ou (iii) forem violadas, obtemos uma nova K-árvore T_K' com custo inferior a $\bar{T}_K$ (K-árvore ótima). Logo (i), (ii) e (iii) se verificam.

($\leftarrow$) Suponha agora que $\bar{T}_K$, satisfazendo (i), (ii) e (iii) não seja ótima. Teremos neste caso, a existência de uma K-árvore $\tilde{T}_K$ com custo c($\tilde{T}_K$) inferior a c($\bar{T}_K$). Do lema VII.1, temos uma função biunívoca h: $\bar{T}_K - \tilde{T}_K \rightarrow \tilde{T}_K - \bar{T}_K$, de maneira que cada um dos pares obtidos gerem trocas admissíveis em $\bar{T}_K$. Essa seqüência de trocas pode ser dividida em 2 tipos distintos: (a) trocas que não atualizam o grau do depósito ou (b), trocas que atualizam o grau do depósito (nó 0). É fácil ver que $\tilde{T}_K$ não pode ser obtida apenas através de trocas

do tipo (a), pois teríamos (i) ou (ii) violadas chegando portanto a um absurdo. Logo, devemos ter também trocas do tipo (b). Considere (sem perda de generalidade) um conjunto de **p** trocas que incrementam o grau do depósito gerando uma nova K-árvore T_K' com custo $c(T_K')$ superior a $c(\overline{T}_K)$. Suponha ainda que $\tilde{T}_K$ seja gerada através de **p** trocas que decrementam o grau do nó 0 (a partir de T_K'), e de maneira que $c(\tilde{T}_K) < c(\overline{T}_K)$. Deveremos ter neste caso, pelo menos um par de trocas (entre as 2p trocas) que incrementa e decrementa respectivamente o grau do depósito em $\overline{T}_K$ gerando uma nova K-árvore T_K'' e com custo $c(T_K'') < c(\overline{T}_K)$, contrariando portanto a condição (iii). •

Como discutido anteriormente, um "novo" limite inferior é calculado após forçarmos a presença da aresta $\overline{f}$ (não pertencente a T_K), na solução do problema lagrangeano. Este limite inferior deverá ser comparado com o menor limite superior z_{UB} obtido até o momento. Caso esse limite inferior seja maior que a solução heurística corrente podemos eliminar $\overline{f}$ definitivamente de nosso grafo. Isto significa que, se uma solução contendo $\overline{f}$ está associada a um limite inferior maior que z_{UB} então $\overline{f}$ não poderá estar presente na solução ótima do problema original.

O algoritmo seguinte calcula o valor do "novo" limite inferior $c(\overline{T}_K)$ obtido após a resolução do problema P(2K, $\overline{f}$) a partir de P(2K). O valor C_{LB} abaixo, representa o termo constante presente na função lagrangeana (associado aos multiplicadores de Lagrange). Obviamente, no processo de redução de nosso grafo, este procedimento deverá ser executado para cada uma das arestas $\overline{f}$ não pertencentes a T_K (solução de P(2K)) e não adjacentes ao depósito. Adiante, estudaremos um procedimento análogo para as arestas $\bar{f}_0$ adjacentes ao depósito (e não pertencentes a T_K).

PROCEDIMENTO VII.1 : (Fixa em *zero* aresta $\overline{f} \notin T_K$ e não adjacente ao depósito)

Início

- $c(\overline{T}_K) \leftarrow +\infty$; {inicializa custo de $\overline{T}_K$}
- **Para** (cada $e \in T_K$) **faça** {pesquisa arestas não adjacentes ao depósito}

- **Se** (troca (e, $\bar{f}$) é admissível em T_K) **então**

  - $T_K' = T_K - e \cup \{\bar{f}\}$;

  - calcula menor troca admissível (e_0, f_0) em T_K' ;

  - **Se** ($c(f_0) - c(e_0) < 0$) **então**

    - $T_K'' = T_K' - e_0 \cup \{f_0\}$;

  senão

    - $T_K'' = T_K'$;

  fim;

  - **Se** $(c(T_K'') < c(\overline{T}_K))$ **então**

    - $c(\overline{T}_K) \leftarrow c(T_K'')$;

fim se;

fim para;

- **Para** (cada aresta $e_0 \in T_K$) **faça** {pesquisa arestas adjacentes ao depósito}

  - **Se** (troca $(e_0, \bar{f})$ é admissível em T_K) **então**

    - $T_K' = T_K - e_0 \cup \{\bar{f}\}$; {decrementa grau }

    - calcula menor troca admissível (e, f_0) em T_K' (sendo $e \neq \bar{f}$);

    - $T_K'' = T_K' - e \cup \{f_0\}$; {incrementa grau }

    - **Se** $(c(T_K'') < c(\overline{T}_K))$ **então**

      - $c(\overline{T}_K) \leftarrow c(T_K'')$;

  fim;

fim;

- **Se** ($(c(\overline{T}_K) + C_{LB}) > z_{UB}$) **então**

  - retiramos aresta $\bar{f}$ de $G(N_0, E)$;

fim. {procedimento}

Figura VII.1: Fixa em *zero* a variável $\bar{f}$ não adj. ao depósito

O teorema seguinte garante a otimalidade da K-árvore $\overline{T}_K$ gerada pelo procedimento VII.1:

Teorema VII.2: Seja $G(N_0,E)$ um grafo e $T_K \subset$ E uma K-árvore mínima (com grau 2K no depósito). Se $\overline{f}$ é uma aresta de E não pertencente a T_K, o algoritmo VII.1 sempre determina uma K-árvore mínima $\overline{T}_K$ com 2K arestas incidentes ao depósito e $\overline{f}$ pertencente a $\overline{T}_K$.

Demonstração:

Nossa prova consiste em mostrar que a K-árvore mínima $\overline{T}_K$ (gerada ao final do algoritmo VII.1) satisfaz as condições de otimalidade expressas no teorema VII.1.

As arestas representadas por **e** pertencem a T_K, e aquelas representadas por **f** não pertencem a T_K. Além disso, as arestas com índice 0 são adjacentes ao depósito. Ao final do procedimento VII.1, teremos apenas uma das 3 possibilidades abaixo:

i) $\overline{T}_K = T_K' = T_K - e_r \cup \{\overline{f}\}$, para algum $e_r \in T_K$ tal que c($\overline{f}$)-c(e_r) seja mínimo e $(e_r, \overline{f})$ seja troca admissível em T_K. Além disso, $c(f_0) - c(e_0) \geq 0$, $\forall\ (e_0, f_0)$ admissível em T_K'.

ii) $\overline{T}_K = T_K - e_s - e_s^0 \cup \{\overline{f}, f_s^0\}$ para alguma tripla e_s, e_s^0 e f_s^0 tais que $(e_s, \overline{f})$ e (e_s^0, f_s^0) sejam admissíveis em T_K e T_K' respectivamente. Além disso, $c(f_s^0) - c(e_s^0) < 0$ é mínimo entre todas as possíveis trocas admissíveis satisfatórias (e_0, f_0) em T_K'.

iii) $\overline{T}_K = T_K - e_t - e_t^0 \cup \{\overline{f}, f_t^0\}$ para alguma tripla de arestas e_t, e_t^0 e f_t^0 tais que $(e_t^0, \overline{f})$ e (e_t, f_t^0) são admissíveis em T_K e T_K' respectivamente (sendo $e_t \neq \overline{f}$) e $c(\overline{f}) - c(e_t^0) + c(f_t^0) - c(e_t)$ mínimo entre todos os pares de trocas que atualizam o grau do depósito.

Sem perda de generalidade, consideremos a situação onde $\overline{T}_K$ gerada pelo algoritmo é descrita conforme o item (II.) acima.

Suponha inicialmente que a 1ª condição de otimalidade seja violada (teorema VII.1). Neste caso, teremos a existência de uma troca admissível satisfatória (e_1, f_1) em $\overline{T}_K$ onde $e_1 \in \overline{T}_K$ e $f_1 \notin \overline{T}_K$. Seja $\tilde{T}_K = \overline{T}_K - e_1 \cup \{f_1\}$ a nova K-árvore obtida (note que $c(\tilde{T}_K) < c(\overline{T}_K)$). Segue então que: $\tilde{T}_K = T_K - e_s - e_s^0 - e_1 \cup \{\bar{f}, f_s^0, f_1\}$ (sendo $e_1 \neq \bar{f}$).

Do lema VII.1 temos garantida a existência de uma função biunívoca $h: \{e_s, e_s^0, e_1\} \rightarrow \{\bar{f}, f_s^0, f_1\}$ gerando 6 possíveis emparelhamentos em T_K :

1) $(e_s^0, \bar{f}); (e_s, f_s^0); \underline{(e_1, f_1)}$

2) $(e_s^0, f_s^0); (e_s, \bar{f}); \underline{(e_1, f_1)}$

3) $\underline{(e_s^0, f_1)}; (e_s, \bar{f}); \underline{(e_1, f_s^0)}$

4) $(e_s^0, \bar{f}); \underline{(e_s, f_1)}; (e_1, f_s^0)$

5) $(e_s^0, f_s^0); \underline{(e_s, f_1)}; (e_1, \bar{f})$

6) $\underline{(e_s^0, f_1)}; (e_1, \bar{f}); \underline{(e_s, f_s^0)}$

Note que pelo menos um dos itens acima determina uma seqüência de trocas admissíveis em T_K. Para ilustrar a demonstração dos 6 casos faremos apenas uma análise dos itens (1) e (3). As demais situações ocorrem de maneira análoga.

Vejamos inicialmente a situação expressa no item (1). Como T_K é solução ótima para p(2K) temos $c(f_1) - c(e_1) \geq 0$ (1ª condição de otimalidade do teorema IV.4). Note que $(e_s, \bar{f})$ e (e_s^0, f_s^0) são admissíveis em T_K e T_K' respectivamente (pois $\overline{T}_K$ é gerada como no item (ii)). Assim, desprezando a troca (e_1, f_1) temos:

$$c(\tilde{T}_K) \geq c(T_K) - c(e_s^0) - c(e_s) + c(\bar{f}) + c(f_s^0) = c(\overline{T}_K).$$

Chegamos portanto a uma situação absurda já que tínhamos $c(\tilde{T}_K) < c(\overline{T}_K)$.

No item (3) temos (e_s^0, f_1) e (e_1, f_s^0) não resultam em uma troca admissível satisfatória para T_K (3ª condição de otimalidade/ teorema IV.4). Como $\overline{T}_K$, gerada pelo

algoritmo VII.1 é expressa no item (ii), temos a existência de arestas f_s^0 *e* e_s^0 tais que $c(f_s^0) - c(e_s^0) < 0$. Segue então que:

$$c(\tilde{T}_K) \geq c(T_K) - c(e_s) + c(\bar{f}) \geq c(T_K) - c(e_s^0) - c(e_s) + c(\bar{f}) + c(f_s^0) = c(\overline{T}_K).$$

Chegamos novamente a uma situação absurda. Note que as trocas sublinhadas podem ser desprezadas por satisfazerem a 1ª ou 3ª condições de otimalidade.

Suponha agora que a 2ª condição de otimalidade (teorema VII.1) seja violada por um par de trocas admissíveis (e_1^0, f_1^0) (ambas adjacentes ao depósito). Do lema VII.1 temos garantida a existência de uma função biunívoca $h: \{e_s, e_s^0, e_1^0\} \rightarrow \{\bar{f}, f_s^0, f_1^0\}$ gerando 6 possíveis emparelhamentos em T_K:

1) $(e_s^0, \bar{f}); (e_s, f_s^0); \underline{(e_1^0, f_1^0)}$

2) $(e_s^0, f_s^0); (e_s, \bar{f}); \underline{(e_1^0, f_1^0)}$

3) $(e_s^0, f_1^0); (e_s, \bar{f}); \underline{(e_1^0, f_s^0)}$

4) $(e_s^0, \bar{f}); (e_s, f_1^0); \underline{(e_1^0, f_s^0)}$

5) $\underline{(e_s^0, f_s^0)}; (e_s, f_1^0); (e_1^0, \bar{f})$

6) $\underline{(e_s^0, f_1^0)}; (e_1^0, \bar{f}); (e_s, f_s^0)$

No item (3) temos que (e_1^0, f_s^0) não resulta em uma troca admissível satisfatória para T_K (2ª condição de otimalidade / teorema VII.1). Assim, $c(f_s^0) - c(e_1^0) \geq 0$. Segue então que:

$$c(\tilde{T}_K) \geq c(T_K) - c(e_s^0) - c(e_s) + c(\bar{f}) + c(f_1^0) \geq c(T_K) - c(e_s^0) - c(e_s) + c(\bar{f}) + c(f_s^0) = c(\overline{T}_K)$$

Logo, $c(\tilde{T}_K) \geq c(\overline{T}_K)$ o que leva a um absurdo pois tínhamos $c(\tilde{T}_K) < c(\overline{T}_K)$. Nos demais itens procedemos de maneira análoga. As trocas sublinhadas podem ser desprezadas por satisfazerem a 2ª condição de otimalidade.

Suponha agora que a 3ª condição de otimalidade seja violada por um par de trocas admissíveis $(e_1, f_1^0); (e_2^0, f_2)$ em $\overline{T}_K$ (sendo $e_1 \neq \bar{f}$). Do lema VII.2, temos a existência de uma nova K-árvore $\tilde{T}_K = \overline{T}_K - e_1 - e_2^0 \cup \{f_1^0, f_2\}$.

Segue então que $\tilde{T}_K = T_K - e_s - e_s^0 - e_1 - e_2^0 \cup \{\overline{f}, f_s^0, f_1^0, f_2\}$. Do lemaVII.1, temos a existência de uma função biunívoca $h: \{e_s, e_s^0, e_1, e_2^0\} \to \{\overline{f}, f_s^0, f_1^0, f_2\}$ de maneira que cada um dos pares obtidos gerem trocas admissíveis em T_K. As 24 possibilidades de emparelhamento são listadas abaixo. Destacamos (trocas sublinhadas), os pares que podem ser desprezados por satisfazerem a 1ª, 2ª ou 3ª condição de otimalidade respectivamente. Isto sempre pode ser feito já que T_K é K-árvore ótima (teorema IV.4).

1) $(e_s\ \bar{f}); (e_s^0, f_s^0);\ \underline{(e_1, f_1^0); (e_2^0, f_2)}$

2) $(e_s\ \bar{f}); (e_s^0, f_s^0);\ \underline{(e_1, f_2); (e_2^0, f_1^0)}$

3) $(e_s\ \bar{f}); (e_s^0, f_1^0);\ \underline{(e_1, f_s^0); (e_2^0, f_2)}$

4) $(e_s\ \bar{f}); (e_s^0, f_1^0);\ \underline{(e_1, f_2); (e_2^0, f_s^0)}$

5) $(e_s\ \bar{f});\ \underline{(e_s^0, f_2); (e_1, f_s^0);} (e_2^0, f_1^0)$

6) $(e_s\ \bar{f});\ \underline{(e_s^0, f_2); (e_1, f_1^0);} (e_2^0, f_s^0)$

7) $(e_s\ f_s^0); (e_s^0, \bar{f});\ \underline{(e_1, f_1^0); (e_2^0, f_2)}$

8) $(e_s\ f_s^0); (e_s^0, \bar{f});\ \underline{(e_1, f_2); (e_2^0, f_1^0)}$

9) $\underline{(e_s\ f_s^0);} (e_s^0, f_1^0); (e_1, \bar{f}); \underline{(e_2^0, f_2)}$

10) $(e_s\ f_s^0);\ \underline{(e_s^0, f_1^0); (e_1, f_2);} (e_2^0, \bar{f})$

11) $\underline{(e_s\ f_s^0); (e_s^0, f_2);} (e_1, \bar{f}); (e_2^0, f_1^0)$

12) $\underline{(e_s\ f_s^0); (e_s^0, f_2);} (e_2^0, \bar{f}); (e_1, f_1^0)$

12) $\underline{(e_s\ f_s^0); (e_s^0, f_2);} (e_2^0, \bar{f}); (e_1, f_1^0)$

13) $(e_s\ f_1^0); (e_s^0, \bar{f});\ \underline{(e_1, f_s^0); (e_2^0, f_2)}$

14) $(e_s\ f_1^0); (e_s^0, \bar{f});\ \underline{(e_1, f_2); (e_2^0, f_s^0)}$

15) $\underline{(e_s\ f_1^0);}\ \underline{(e_s^0, f_s^0);} (e_1, \bar{f}); \underline{(e_2^0, f_2)}$

16) $(e_s\ f_1^0);\ \underline{(e_s^0, f_s^0); (e_1, f_2);} (e_2^0, \bar{f})$

17) $\underline{(e_s\ f_1^0); (e_s^0, f_2);} (e_1, \bar{f}); (e_2^0, f_s^0)$

18) $\underline{(e_s\ f_1^0); (e_s^0, f_2);} (e_1, f_s^0); (e_2^0, \bar{f})$

19) $\underline{(e_s, f_2);} \underline{(e_s^0, \bar{f});} (e_1, f_s^0); \underline{(e_2^0, f_1^0)}$

20) $\underline{(e_s, f_2);} (e_s^0, \bar{f}); (e_1, f_1^0); \underline{(e_2^0, f_s^0)}$

22) $\underline{(e_s, f_2);\ (e_s^0, f_s^0);} (e_1, f_1^0); (e_2^0, \bar{f})$

23) $\underline{(e_s, f_2);\ (e_s^0, f_1^0);} (e_1, \bar{f}); (e_2^0, f_s^0)$

24) $\underline{(e_s, f_2);\ (e_s^0, f_1^0);} (e_2^0, \bar{f}); (e_1, f_s^0)$

Para exemplificar as operações realizadas consideremos (por exemplo) a situação expressa na linha (16). As outras situações são análogas. Assim:

$$(e_s, f_1^0);(e_2^0, \overline{f});(e_s^0, f_s^0);(e_1, f_2) \qquad (16)$$

Note que:

$$c(\tilde{T}_K) = c(T_K) - c(e_s) - c(e_s^0) - c(e_1) - c(e_2^0) + c(\overline{f}) + c(f_s^0) + c(f_1^0) + c(f_2).$$

Temos que (e_s^0, f_s^0) e (e_1, f_2) são trocas admissíveis não satisfatórias em T_K (pois T_K é K-árvore ótima). Assim, $c(f_s^0) - c(e_s^0) \geq 0$ e $c(f_2) - c(e_1) \geq 0$ respectivamente. Segue então que:

$$c(\tilde{T}_K) \geq c(T_K) - c(e_s) - c(e_2^0) + c(\overline{f}) + c(f_1^0) \geq c(T_K) + c(\overline{f}) - c(e_s) + c(f_s^0) - c(e_s^0) = c(\overline{T}_K)$$

Chegamos portanto a um absurdo já que, ao violarmos a 3ª condição de otimalidade, obtemos uma nova K-árvore $\tilde{T}_K$ com custo inferior a $\overline{T}_K$. Esta contradição ocorre de maneira análoga nas demais possibilidades expressas acima. •

No procedimento VII.1, determinamos um critério para fixar em *zero* as arestas (variáveis) não pertencentes a T_K e não adjacentes ao depósito. Vejamos agora como fixar em *zero* as variáveis não pertencentes a T_K e adjacentes ao depósito. Analogamente ao caso anterior, denotaremos por P'(2K, $\overline{f}_0$), ao problema de se encontrar uma K-árvore mínima $\overline{T}_K$ com depósito de grau 2K e tal que a aresta $\overline{f}_0$ (adjacente ao depósito) pertença a $\overline{T}_K$.

Apresentamos a seguir, as condições de otimalidade que devem ser satisfeitas por uma solução de P'(2K, $\overline{f}_0$). A demonstração é análoga ao teorema VII.1.

Teorema VII.3: (Condições de Otimalidade)

Uma K-árvore $\overline{T}_K$ com grau 2K no depósito será ótima para o problema P'(2K, $\overline{f}_0$), se e somente se satisfazer as seguintes condições de otimalidade:

i) Sejam e_1 e f_1 duas arestas não incidentes ao depósito. Não existem trocas admissíveis satisfatórias (e_1 , f_1) (onde $e_1 \in \overline{T}_K$ e $f_1 \notin \overline{T}_K$).

ii) Não haverá trocas admissíveis satisfatórias (e_1^0, f_1^0) onde $e_1^0 \in \overline{T}_k$ e $f_1^0 \notin \overline{T}_K$ sendo e_1^0 e f_1^0 incidentes ao depósito e $\bar{f}_0 \neq e_1^0$.

iii) Não haverá duas trocas admissíveis satisfatórias (e_1^0, f_1) e (e_2, f_2^0) com $e_1^0, e_2 \in \overline{T}_K$, $f_1, f_2^0 \notin \overline{T}_K$, sendo e_1^0, f_2^0 incidentes no depósito e f_1, e_2 não incidentes ao depósito e $e_1^0 \neq \bar{f}_0$.

Prova: Demonstração análoga ao teorema VII.1. ●

Apresentamos a seguir, um procedimento semelhante ao procedimento VII.1, para solução do problema P'(2K, $\overline{f}_0$) a partir da solução de P(2K). Este algoritmo é executado para cada uma das arestas $\overline{f}_0$ não pertencentes a T_K e adjacentes ao depósito. Analogamente ao caso anterior, C_{LB} representa o somatório das parcelas associadas aos multiplicadores da função lagrangeana.

PROCEDIMENTO VII.2 (Fixa em *zero* aresta $\overline{f}_0 \notin T_K$ adjacente ao depósito)

Início

- $c(\overline{T}_K) \leftarrow \infty$;
- **Para** (cada $e \in T_K$) **faça** {pesquisa arestas não adjacentes ao depósito}
  - **Se** (troca $(e, \bar{f}_0)$ é admissível em T_K) **então**
    - $T_K' = T_K - e \cup \{\bar{f}_0\}$; {incrementa grau }
    - calcula menor troca admissível (e_0, f) em T_K' (sendo $e_0 \neq \bar{f}_0$);
    - $T_K'' = T_K' - e_0 \cup \{f\}$; {decrementa grau }
    - **Se** ($c(T_K'') < c(\overline{T}_K)$ **então**
      - $c(\overline{T}_K) \leftarrow c(T_K'')$;

  fim;

fim;

- **Para** (cada $e_0 \in T_K$) **faça** {pesquisa arestas adjacentes ao depósito}
  - **Se** (troca $(e_0, \bar{f}_0)$ é admissível em T_K) **então**

- $T_K' = T_K - e_0 \cup \{\bar{f}_0\}$;

- calcula menor troca admissível (e, f) em T_K';

- **Se** ($c(f) - c(e) < 0$) **então**

- $T_K'' = T_K' - e \cup \{f\}$;

senão

- $T_K'' = T_K'$;

fim;

- **Se** ($c(T_K'') < c(\overline{T}_K)$) **então**

- $c(\overline{T}_K) \leftarrow c(T_K'')$;

fim se;

fim para;

- **Se** ($c(\overline{T}_K) + C_{LB} > z_{UB}$) **então**

- retiramos aresta $\bar{f}_0$ de $G(N_0, E)$;

fim;

fim. {procedimento}

Figura VII.2: Fixa em *zero* a variável $\bar{f}_0$ adj. ao depósito

O teorema seguinte garante a otimalidade da K-árvore $\overline{T}_K$ gerada pelo algoritmo VII.2. A demonstração é análoga ao teorema VII.2.

Teorema VII.4: Seja $G(N_0, E)$ um grafo e $T_K \subset$ E uma K-árvore mínima (com grau 2K no depósito). Se $\overline{f}_0$ é uma aresta de E adjacente ao depósito e não pertencente a T_K, o algoritmo VII.2 acima sempre determina uma K-árvore mínima $\overline{T}_K$ com 2K arestas incidentes ao depósito e $\overline{f}_0 \in \overline{T}_K$.

Prova:

Demonstração análoga ao teorema VII.2. •

Apesar de resolvermos P(2K, $\overline{f}$) e P'(2K, $\overline{f}_0$) para cada uma das arestas $\overline{f}$ e $\overline{f}_0$ não pertencentes a T_K (solução de P(2K)), necessitamos de um grande esforço computacional na resolução destes 2 problemas. Como temos $m = O(n^2)$ arestas em nosso grafo, podemos constatar nestes 2 procedimentos, que serão necessários pelos menos $O(n^2)$ passos, para determinação da menor troca admissível em T_K'. Note ainda que n+K arestas de T_K devem ser pesquisadas para cada aresta $\overline{f}$ (ou $\overline{f}_0$) $\notin T_K$. Necessitaremos portanto, de pelo menos $O(n^3)$ passos na solução de P(2K, $\overline{f}$) (ou P(2K, $\overline{f}_0$)) respectivamente. Assim, ao resolvermos P(2K, $\overline{f}$) (ou P(2K, $\overline{f}_0$)) para cada uma das arestas não pertencentes a T_K, teremos uma complexidade total de $O(n^5)$ iterações.

Esta complexidade é bastante elevada, já que, se utilizamos o procedimento que calcula a K-árvore mínima (de complexidade $O(n^3)$) , para cada uma das arestas $\overline{f}$ (ou $\overline{f}_0$) não pertencentes a T_K teremos também $O(n^5)$ iterações!

Uma alternativa interessante, embora não tão eficiente em termos da qualidade do resultado, é determinarmos limites inferiores para uma solução de P(2K,$\overline{f}$) ou P'(2K,$\overline{f}_0$) respectivamente. Caso esses limites sejam superiores ao custo da solução heurística, eliminamos definitivamente $\overline{f}$ ou $\overline{f}_0$ de nosso grafo.

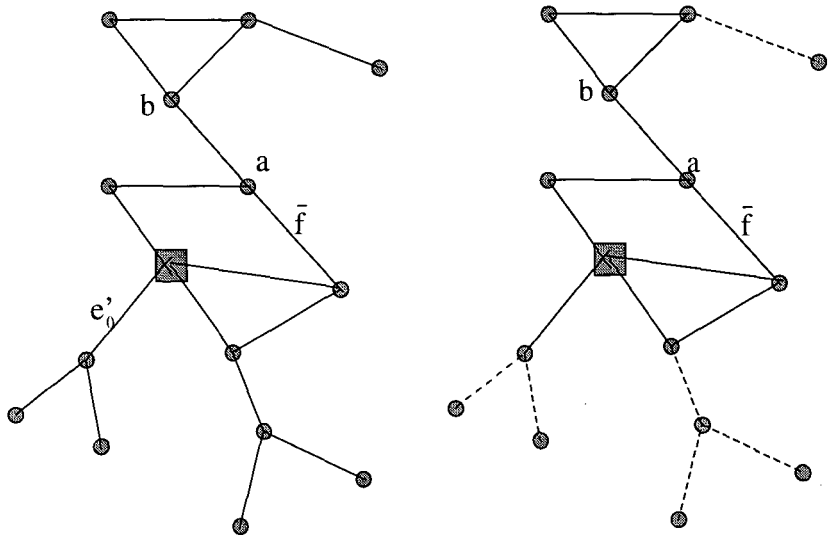
VII.3.1 - Determinação de limites inferiores para P(2K,$\overline{f}$) e P'(2K,$\overline{f}_0$):

Para pesquisarmos a determinação de limites inferiores para os problemas P(2K,$\overline{f}$) e P'(2K,$\overline{f}_0$), faremos apenas a análise de P(2K,$\overline{f}$), no outro caso procedemos de maneira análoga.

Nos preocuparemos apenas em determinar (no procedimento VII.1), se uma dada troca, é ou não admissível em T_K. No grafo T_K' resultante, calculamos a menor troca possível, sendo ela admissível ou não. Isto irá permitir uma pequena redução no esforço computacional realizado.

Assim, no procedimento VII.1, devemos determinar se $(e,\bar{f})$ e $(e_0,\bar{f})$ são ou não admissíveis em T_K.

Para sabermos se cada umas das trocas $(e,\bar{f})$ são admissíveis em T_K, fazemos uma "contração" da nova K-árvore obtida após a inserção de $\bar{f}$ (aresta pesquisada). Percorremos as arestas da nova K-árvore e "eliminamos" todas as arestas não adjacentes ao depósito com um extremo de grau *um*. O processo deverá ser repetido se existirem novas arestas com um extremo de grau *um*. É fácil ver que, todas aquelas arestas eliminadas não formam trocas admissíveis com $\bar{f}$. As arestas resultantes da "contração" determinam, em sua grande maioria, trocas admissíveis com $\bar{f}$. Para ilustrar essa situação, vejamos o seguinte exemplo:



(a) 2-árvore após inserção de f̄ **(b) 2-árvore "contraída"**

Figura VII.3: Determinação de trocas admissíveis

Cada uma das arestas representadas em linha pontilhada, não definem trocas admissíveis com $\bar{f}$. As demais arestas não adjacentes ao depósito, à exceção da aresta (a,b), definem trocas admissíveis com $\bar{f}$. Como buscamos um limite inferior p/ P(2K, $\bar{f}$) a um baixo custo computacional não nos preocupamos com a determinação daquelas arestas que desconectam o grafo resultante após sua exclusão (p.ex., a aresta (a,b)).

Para sabermos se $(e_0, \bar{f})$ define uma troca admissível em T_K, bastará fazer uma atualização do número de arestas que ligam cada uma das componentes conexas obtidas após a inserção de $\bar{f}$. Dessa maneira, uma aresta e_0 não irá determinar uma troca $(e_0, \bar{f})$ admissível se e_0 for a única aresta que une a componente conexa correspondente ao depósito. Note, em nosso exemplo que $\bar{f}$ e e_0' não definem uma troca admissível com $\bar{f}$. Ao contrário, as demais arestas adjacentes, definem, obviamente, trocas admissíveis com $\bar{f}$.

VII.4 - Fixando variáveis em *um:*

Como observado na seção VII.2, representamos por Q(k,*e*) o problema de se encontrar uma K-árvore mínima $\overline{T}_k$ com depósito de grau k de maneira que *e*, não pertença a $\overline{T}_k$. Em nosso caso, a solução de Q(2K,*e*) deverá ser obtida a partir de T_K (solução de P(2K)). Ou seja, estaremos interessados na resolução de Q(2K,*e*) tomando-se as arestas *e* pertencentes a T_K.

Nesta seção, faremos uma distinção entre as arestas de $\overline{T}_K$ adjacentes e não adjacentes ao depósito. Assim, representaremos por Q(2K, $\bar{e}$), ao problema de se encontrar uma K-árvore mínima $\overline{T}_K$ com depósito de grau 2K e $\bar{e}$ (não adjacente ao depósito) não pertencente a $\overline{T}_K$.

Apresentamos a seguir as condições de otimalidade que devem ser satisfeitas por uma solução de Q(2K, $\bar{e}$). A demonstração é análoga ao teorema VII.1:

Teorema VII.5: (Condições de Otimalidade)

Uma K-árvore $\overline{T}_K$ com grau 2K no depósito será ótima para o problema Q(2K, $\bar{e}$) se e somente se satisfizer as seguintes condições de otimalidade:

i) Sejam e_1 e f_1 duas arestas não incidentes ao depósito. Não existem trocas admissíveis satisfatórias (e_1, f_1) (onde $e_1 \in \overline{T}_K$, $f_1 \notin \overline{T}_K$ e $f_1 \neq \bar{e}$).

ii) Não haverá trocas admissíveis satisfatórias (e_1^0, f_1^0) onde $e_1^0 \in \overline{T}_K$ e $f_1^0 \notin \overline{T}_K$ sendo e_1^0 e f_1^0 incidentes ao depósito.

iii) Não haverá duas trocas admissíveis satisfatórias (e_1^0, f_1) e (e_2, f_2^0) com $e_1^0, e_2 \in \overline{T}_K$, $f_1, f_2^0 \notin \overline{T}_K$, sendo e_1^0, f_2^0 incidentes no depósito; f_1, e_2 não incidentes ao depósito e $f_1 \neq \overline{e}$.

Prova: Demonstração análoga ao teorema VII.1. •

O algoritmo seguinte calcula o valor do "novo" limite inferior c($\overline{T}_K$), obtido após a resolução do problema Q(2K, $\overline{e}$) a partir de P(2K). O valor C_{LB} abaixo, representa o termo constante presente na função lagrangeana (associado aos multiplicadores de Lagrange). Obviamente, no processo de redução de nosso grafo, este procedimento deverá ser executado para cada uma das arestas $\overline{e}$ pertencentes a T_K (solução de P(2K)) não adjacentes ao depósito. Teremos um procedimento análogo para as arestas $\overline{e}_0$ adjacentes ao depósito e pertencentes a T_K.

PROCEDIMENTO VII.3 : (Fixa em *um* aresta $\overline{e} \in T_K$ não adjacente ao depósito)

Início

- c($\overline{T}_K$) $\leftarrow$ $+\infty$; {inicializa custo de $\overline{T}_K$}
- **Para** (cada $f \notin T_K$) **faça** {pesquisa arestas não adjacentes ao depósito}
  - **Se** (troca ($\overline{e}$, f) é admissível em T_K) **então**
    - $T_K' = T_K - \overline{e} \cup \{f\}$;
    - calcula menor troca admissível (e_0, f_0) em T_K';
    - **Se** ($c(f_0) - c(e_0) < 0$) **então**
      - $T_K'' = T_K' - e_0 \cup \{f_0\}$;

    senão

      - $T_K'' = T_K'$;

    fim;

- **Se** $(c(T_K'') < c(\overline{T}_K))$ **então**

- $c(\overline{T}_K) \leftarrow c(T_K'')$;

fim se;

fim para;

- **Para** (cada aresta $f_0 \notin T_K$) **faça** {pesquisa arestas adjacentes ao depósito}

- **Se** (troca $(\overline{e}, f_0)$ é admissível em T_K) **então**

- $T_K' = T_K - \overline{e} \cup \{f_0\}$; {incrementa grau }

- calcula menor troca admissível (e_0, f) em T_K' (sendo $f \neq \overline{e}$);

- $T_K'' = T_K' - e_0 \cup \{f\}$; {decrementa grau }

- **Se** $(c(T_K'') < c(\overline{T}_K))$ **então**

- $c(\overline{T}_K) \leftarrow c(T_K'')$;

fim;

fim;

- **Se** $((c(\overline{T}_K) + C_{LB}) > z_{UB})$ **então**

- fixamos $\overline{e}$ em $G(N_0, E)$;

fim. {procedimento}

Figura VII.4: Fixa em *um* a variável $\overline{e}$ não adj. ao depósito

O teorema seguinte garante a otimalidade de $\overline{T}_K$ (solução de Q(2K, $\overline{e}$)) a partir de T_K (solução de p(2K)).

Teorema VII.6: Seja $G(N_0, E)$ um grafo e $T_K \subset$ E uma K-árvore mínima (com grau 2K no depósito). Se $\overline{e}$ é uma aresta de E não adjacente ao depósito e pertencente a T_K, o algoritmo VII.3 sempre determina uma K-árvore mínima $\overline{T}_K$ com 2K arestas incidentes ao depósito e $\overline{e}$ não pertencente a $\overline{T}_K$.

Prova:

Demonstração análoga ao teorema VII.2. •

No procedimento VII.3, determinamos um critério para fixar em *um,* as arestas (variáveis) pertencentes a T_K e não adjacentes ao depósito. Vejamos agora como fixar em *um* as variáveis pertencentes a T_K e adjacentes ao depósito. Analogamente ao caso anterior, denotaremos por Q'(2K, $\overline{e}_0$), o problema de se encontrar uma K-árvore mínima $\overline{T}_K$ com depósito de grau 2K e tal que, a aresta $\overline{e}_0$ (adjacente ao depósito) não pertença a $\overline{T}_K$.

Apresentamos a seguir, as condições de otimalidade que devem ser satisfeitas por uma solução de Q'(2K, $\overline{e}_0$). Novamente, a demonstração é análoga ao teorema VII.1.

Teorema VII.7: (Condições de Otimalidade)

Uma K-árvore $\overline{T}_K$ com grau 2K no depósito será ótima para o problema Q'(2K, $\overline{e}_0$), se e somente se satisfazer as seguintes condições de otimalidade:

i) Sejam e_1 e f_1 duas arestas não incidentes ao depósito. Não existem trocas admissíveis satisfatórias (e_1 , f_1) (onde $e_1 \in \overline{T}_K$ e $f_1 \notin \overline{T}_K$).

ii) Não haverá trocas admissíveis satisfatórias (e_1^0, f_1^0) onde $e_1^0 \in \overline{T}_K$ e $f_1^0 \notin \overline{T}_K$ sendo e_1^0 e f_1^0 incidentes ao depósito e $f_1^0 \neq \overline{e}_0$.

iii) Não haverá duas trocas admissíveis satisfatórias (e_1^0, f_1) e (e_2, f_2^0) com $e_1^0, e_2 \in \overline{T}_K$, $f_1, f_2^0 \notin \overline{T}_K$, sendo e_1^0, f_2^0 incidentes no depósito e f_1, e_2 não incidentes ao depósito e $f_2^0 \neq \overline{e}_0$.

Prova: Demonstração análoga ao teorema VII.1. •

Apresentamos a seguir um procedimento, semelhante ao procedimento VII.3, para solução do problema Q'(2K, $\overline{e}_0$), a partir da solução de P(2K). Este algoritmo é executado para cada uma das arestas $\overline{e}_0$ pertencentes a T_K e adjacentes ao depósito. Analogamente ao caso anterior, C_{LB} representa o somatório das parcelas associadas aos multiplicadores da função lagrangeana.

PROCEDIMENTO VII.4 : (Fixa em *um* a aresta $\bar{e}_0 \in T_K$ adjacente ao depósito)

Início

- $c(\bar{T}_K) \leftarrow +\infty$; {inicializa custo de $\bar{T}_K$ }
- **Para** (cada $f \notin T_K$) **faça** {pesquisa arestas não adjacentes ao depósito}
  - **Se** (troca $(\bar{e}_0, f)$ é admissível em T_K) **então**
    - $T_K' = T_K - \bar{e}_0 \cup \{f\}$; {decrementa grau do depósito}
    - calcula menor troca admissível (e, f_0) em T_K'; (p/ $f_0 \neq \bar{e}_0$)
    - $T_K'' = T_K' - e \cup \{f_0\}$; {incrementa grau do depósito}
    - **Se** $(c(T_K'') < c(\bar{T}_K))$ **então**

      $c(\bar{T}_K) \leftarrow c(T_K'')$;

  fim se;

fim para;

- **Para** (cada aresta $f_0 \notin T_K$) **faça** {pesquisa arestas adjacentes ao depósito}
  - **Se** (troca $(\bar{e}_0, f_0)$ é admissível em T_K) **então**
    - $T_K' = T_K - \bar{e}_0 \cup \{f_0\}$;
    - calcula menor troca admissível (e, f) em T_K';
    - **Se** ($c(f) - c(e) < 0$) **então**

      - $T_K'' = T_K' - e \cup \{f\}$;

      senão

      - $T_K'' = T_K'$;

    fim;

    - **Se** $(c(T_K'') < c(\bar{T}_K))$ **então**

      - $c(\bar{T}_K) \leftarrow c(T_K'')$;

  fim;

fim;

- **Se** ($(c(\bar{T}_K) + C_{LB}) > z_{UB}$) **então**
  - fixamos aresta $\bar{e}_0$ em $G(N_0, E)$;

fim. {procedimento}

Figura VII.5: Fixa em *um* a variável $\bar{e}_0$ adj. ao depósito

O teorema seguinte garante a otimalidade de $\overline{T}_K$ (solução de Q'(2K, $\bar{e}_0$)) a partir de T_K (solução de p(2K)).

Teorema VII.6: Seja $G(N_0, E)$ um grafo e $T_K \subset$ E uma K-árvore mínima (com grau 2K no depósito). Se $\bar{e}_0$ é uma aresta de E adjacente ao depósito e pertencente a T_K, o algoritmo VII.3 sempre determina uma K-árvore mínima $\overline{T}_K$ com 2K arestas incidentes ao depósito e $\bar{e}_0$ não pertencente a $\overline{T}_K$.

Prova:

Demonstração análoga ao teorema VII.2. •

Da mesma forma que nos procedimentos que fixavam variáveis em *zero,* teremos também, neste caso, um grande esforço computacional para gerarmos uma solução de Q(2K, $\bar{e}$) e Q'(2K, $\bar{e}_0$) respectivamente. Para facilitar um pouco essa tarefa, calculamos limites inferiores para para a solução destes 2 problemas.

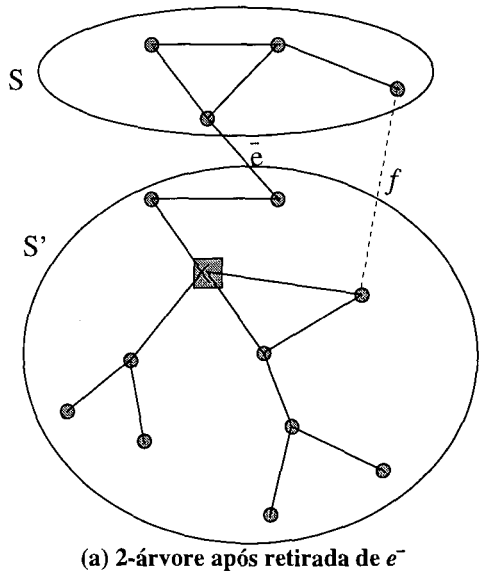
VII.4.1 - Determinação de limites inferiores para Q(2K, $\bar{e}$) e Q'(2K, $\bar{e}_0$):

Para pesquisarmos a determinação de limites inferiores para os problemas Q(2K, $\bar{e}$) e Q'(2K, $\bar{e}_0$), faremos apenas a análise de Q(2K, $\bar{e}$) , no outro caso procedemos de maneira análoga.

De maneira análoga à seção VII.3.1, nos preocupamos apenas em determinar se uma dada troca, é ou não admissível em T_K. Assim, no novo grafo T_K' obtido, calculamos

a menor troca possível, sendo ela admissível ou não. Isto irá permitir uma pequena redução no esforço computacional realizado.

Para determinarmos limites inferiores para uma solução de Q(2K, $\bar{e}$), verificamos apenas se as trocas ($\bar{e}$, f) e ($\bar{e}$, f_0) no procedimento VII.3 são ou não admissíveis em T_K (vide figura VII.6). Em seguida, analisando-se T_K', calculamos as menores trocas (e_0, f_0) e (e_0, f) (para $f \neq \bar{e}$) admissíveis ou não em T_K'.



(a) 2-árvore após retirada de e^-

Figura VII.6: Determinação de trocas admissíveis

Para sabermos se as trocas ($\bar{e}$, f) e ($\bar{e}$, f_0) são ou não admissíveis em T_K, "eliminamos" $\bar{e}$ e checamos se o grafo resultante é ou não conexo. Caso o novo grafo seja desconexo, teremos 2 componentes conexas com rótulos distintos (vide figura VII.6). Qualquer aresta $f \notin T_K$ (onde $f \neq \bar{e}$) ou $f_0 \notin T_K$ conectando estas duas componentes define uma troca admissível com $\bar{e}$.

VII.5 - Estrutura de dados utilizada na fixação de variáveis:

Para melhor usufruir das vantagens oferecidas pelos procedimentos que fixam variáveis, devemos nos preocupar com um armazenamento adequado das variáveis livres (ainda não fixadas) e as já fixadas em *um*.

Como discutido anteriormente, a idéia será mantermos as variáveis fixadas em *um* sempre presentes nas K-árvores mínimas e soluções heurísticas geradas posteriormente. Isto poderá contribuir para uma melhor qualidade na determinação dos limites inferiores e superiores respectivamente. Além disso, o tempo de processamento pode ser reduzido se desprezamos definitivamente as variáveis fixadas em *zero*.

A ilustração seguinte (figura VII.7) mostra como armazenamos as variáveis livres e as variáveis já fixadas em *zero* ou *um*. Como apresentado na capítulo IV (seção IV.3.3), uma lista L contendo 3 campos, armazena as extremidades e o custo de cada aresta. Chamaremos esta lista de CLAG, vetor de custos lagrangeanos.

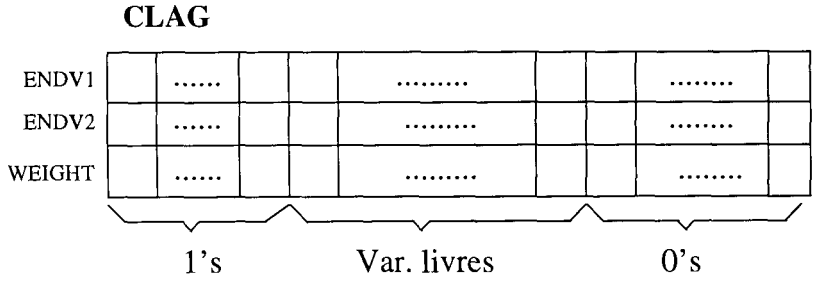


Figura VII.7: Representação das var. fixas e livres no vetor de custos lagrangeanos

Todas as variáveis fixadas em *zero* associadas às arestas não incidentes ao depósito são representadas no final do vetor de custos lagrangeanos. Para fixarmos em *zero* as arestas (variáveis) incidentes ao depósito fazemos o custo lagrangeano correspondente igual a *infinito*. Como veremos posteriormente (capítulo seguinte), isto irá facilitar a determinação das K-árvores mínimas e soluções heurísticas geradas nas demais iterações do subgradiente. Observe que, ao percorrermos CLAG, somente as posições

correspondentes às variáveis livres e iguais a *um* sejam pesquisadas. Teremos consequentemente, uma grande economia no tempo de processamento.

Esta representação permite ainda que identifiquemos facilmente as arestas presentes na solução ótima (iguais a *um*). Estas arestas deverão figurar constantemente nas soluções heurísticas e K-árvores mínimas geradas nas iterações futuras do subgradiente.

Note entretanto que, antes de iniciarmos a fixação de variáveis, utilizamos uma função POSIÇÃO que mapeia todas as arestas do grafo para uma lista de arestas. Ao fixarmos uma variável em *zero* ou *um* fazemos trocas entre as arestas de CLAG alterando a função POSIÇÃO preestabelecida. Para resolvermos este problema, criamos um vetor adicional que indica a nova posição de um par de arestas em CLAG sempre que alguma troca for efetuada.

Capítulo VIII

Resolução do problema dual lagrangeano e a busca em árvore

VIII.1 - Introdução:

Neste capítulo, descrevemos resumidamente, os principais aspectos envolvidos em nossa abordagem na solução do problema de roteamento de veículos com restrições de capacidade. Fazemos uma síntese das principais etapas presentes na solução do problema dual lagrangeano utilizando-se o método subgradiente com geração de desigualdades válidas. Destacamos as etapas de inclusão e eliminação de restrições dualizadas na função lagrangeana, bem como a etapa de fixação de variáveis. Uma nova heurística lagrangeana (seção VIII.3) é também apresentada na determinação dos limites superiores.

Em seguida (seção VIII.4), estudamos a estratégia de ramificação (*branching*) e a busca em árvore (*branch-and-bound*) utilizada em nosso trabalho. Os principais aspectos existentes em nossa implementação são também considerados.

VIII.2 - Método subgradiente com geração de restrições:

Nesta seção, sintetizamos os principais aspectos envolvidos na resolução do problema dual lagrangeano com a utilização do método subgradiente com geração de restrições. Como discutido anteriormente (capítulos VI e VII), devemos tratar da atualização das restrições dualizadas, composta pelas restrições de eliminação de sub-rotas, *combs* e *multistars,* executando-se, paralelamente, a etapa de fixação de variáveis.

Sejam μ, v_S, v_C e v_M os multiplicadores de Lagrange associados às restrições de igualdade, sub-rotas, *combs* e *multistars* respectivamente. Seja ainda X, o conjunto de todas as K-árvores mínimas com 2K arestas incidentes ao depósito. Sintetizando as diversas etapas discutidas nos capítulos IV,V e VI, formulamos o problema relaxado (PR) da seguinte forma:

$$z_D = min \sum_{ij \in E(N_0)} \bar{c}_{ij} x_{ij} + 2\sum_{i=1}^{n} \mu_i + 2\sum_{S \subseteq N} v_S r(S) + \sum_{C \subseteq N_0} v_C r(C) + \sum_{M \subseteq N} v_M r(M) \qquad \text{(PR)}$$
$$s.a. \quad x \in X$$

onde:

$$\bar{c}_{ij} = c_{ij} - \mu_i - \mu_j - \sum_{j=0}^{n} e_j \sum_{\substack{i \in S,\, j \in \bar{S} \\ ou \\ i \in \bar{S},\, j \in S}} v_S + \sum_{\substack{i \in H \\ j \in H}} v_C + \sum_{p=1}^{n_p} \sum_{\substack{i \in T_p \\ j \in T_p}} v_C - \sum_{l=1}^{n_m} \sum_{\substack{i \in S_l \\ j \notin S_l}} v_M$$

$$r(S) = \left\lceil d(S)/b \right\rceil$$

$$r(C) = \frac{3n_p + 1}{2} - |H| - \sum_{i=1}^{n_p} |T_i| - \alpha(k-1)$$

$$r(M) = 4n_m - 2$$

onde os valores e_j são como definidos no capítulo IV, n_p é o número de dentes de cada *comb* violada e, finalmente, n_m o número de pontas de cada *multistar.*

Devemos resolver então o seguinte problema dual lagrangeano:

$$max \quad z_D(\mu, v_S, v_C, v_M)$$
$$s.a. \quad \begin{cases} \mu \in \Re^n \\ v_S, v_C, v_M \geq 0 \end{cases} \qquad \text{(PD)}$$

Na resolução de (PD), utilizamos o método subgradiente (Geoffrion[74], Fisher[81]) combinado à geração de desigualdades válidas.

Seja σ_k, o valor de cada coordenada (do vetor subgradiente) associado a alguma das restrições dualizadas na função lagrangeana (sub-rotas, *combs* ou *multistars* respectivamente). Como discutido em Geoffrion[74], a atualização dos multiplicadores de Lagrange associado às desigualdades dualizadas ocorre de acordo com a seguinte expressão:

$$\lambda_k = max\{0, \lambda_k + \theta\sigma_k\}$$

onde λ_k representa o multiplicador associado à coordenada σ_k, e θ representa o tamanho do passo.

Sejam z_{LB} *e* z_{UB} os limites inferior e superior respectivamente, (obtidos em uma iteração qualquer do subgradiente). O tamanho do passo θ será dado por:

$$\theta = \pi \frac{(z_{UB} - z_{LB})}{\sum_{i=1}^{n} \mu_i^2 + \sum_{S \in A} v_S^2 + \sum_{C \in A} v_C^2 + \sum_{M \in A} v_M^2} \qquad \text{(VIII.1)}$$

onde π $(0<\pi\leq 2)$ é decrementado a uma taxa fixa sempre que tivermos um número pré-fixado de iterações do subgradiente sem um incremento estrito do limite inferior.

Note que, se os multiplicadores λ_k associados às restrições já dualizadas são inicializados com 0, eles serão incrementados no passo seguinte já que $\sigma_k > 0$ *e* $\lambda_k + \theta\sigma_k > 0$.

Para cada uma das K-árvores geradas (pertencentes a X) no método subgradiente, devemos atualizar o valor do subgradiente associado àquelas partições já dualizadas. Assim, como discutido no capítulo III, algumas destas restrições poderão ser eliminadas já que não contribuem mais para o incremento estrito do limite inferior. Além disso, a presença destas restrições reduzem desnecessariamente o tamanho do passo no método subgradiente pois, as restrições não violadas (com subgradiente $\sigma_k \leq 0$) aparecem no

denominador da expressão que define θ. Sempre que $\sigma_k \leq 0$ *e* $\lambda_k = 0$, eliminamos a restrição associada a λ_k de nosso conjunto de restrições dualizadas. Este procedimento difere daquele desenvolvido por Fisher[94.b], onde são realizadas 3 iterações no subgradiente com o multiplicador nulo antes de sua eliminação!

Como discutido no capítulo anterior, necessitamos de um esforço computacional considerável na implementação do procedimento que fixa variáveis. Além disso, é fundamental que a diferença entre o menor limite superior (menorzub) e o maior limite inferior (maiorzlb) seja suficientemente pequena para que um maior número de variáveis possam ser fixadas. Executamos então o procedimento que fixa variáveis quando o *gap* de dualidade (diferença entre o limite superior e inferior) for menor que uma porcentagem pré-determinada. Calculamos o *gap* de dualidade (gapd) de acordo com a seguinte expressão:

$$gapd = \left(\frac{menorzub - maiorzlb}{maiorzlb} \right).100 \qquad \text{(VIII.2)}$$

Em função do elevado custo computacional exigido pelo procedimento que fixa variáveis, nós o executamos apenas quando o *gap* de dualidade for menor que 7% e o limite inferior for incrementado naquela iteração do subgradiente.

Resumindo os passos descritos acima, apresentamos o procedimento da figura VIII.1 que combina o método subgradiente com a geração de restrições. A constante MAXITER representa o número máximo de iterações permitidas na execução do subgradiente (condição de parada). A constante TOTPASSSOS nos indica qual o maior número de iterações do subgradiente sem incremento estrito do limite inferior. Se após TOTPASSOS iterações, o limite inferior não tenha sido atualizado, decrementamos o tamanho de passo π a uma taxa pré-determinada TAXAPRED. A constante ε indica que o subgradiente pode ser executado até que os limites inferiores estejam a uma distancia ε (suficientemente pequena) do valor da melhor solução viável obtida até o momento (solução ε-aproximada). Este ε, pode ser modificado conforme trabalhemos ou não com distancias inteiras entre os clientes. No caso de distância inteiras, poderemos utilizar qualquer $\varepsilon \in (0,1)$ para definir o grau de

proximidade do valor inteiro z_{UB}. O mesmo não ocorre para distâncias racionais, situação onde devemos utilizar ε suficientemente "pequeno".

A heurística lagrangeana utilizada será discutida com mais detalhes na seção seguinte:

PROCEDIMENTO VIII.1: Relaxação Lagrangeana c/ Geração de Restrições;

Início

k := 0;

menorzub := +∞; {inicializa menor limite superior}

maiorzlb := -∞; {inicializa maior limite inferior}

Repita

- Calcula K-árvore mínima c/ 2K arestas incidentes no depósito;
- Calcula limite inferior z_{LB} ;
- Calcula limite superior z_{UB} associado a uma solução heurística c/ K rotas;
- **Se** $z_{UB} < menorzub$ **então** $menorzub \leftarrow z_{UB}$;
- **Se** ($maiorzlb < z_{LB}$) **então**

  $maiorzlb = z_{LB}$;

  calcula gapd; {conforme equação VIII.2}

  Se (gapd ≤ 7%) **então**

    fixa-variáveis;

  iterações := 0;

senão

  iterações := iterações+1;

  Se (iterações = TOTPASSOS) **então**

    $\pi := TAXARED * \pi$; {redução no tamanho do passo}

- **fim se**;
- **Se** (maiorzlb < (menorzub - ε)) **então**
  - Calcula componentes σ_k do vetor subgradiente associado às restrições já dualizadas;

- Dualiza restrições violadas pela solução do prob. dual lagrangeano (σ_k >0);
- **Se** (não existirem restrições violadas) **então**
  - checa condições de otimalidade;
- **Se** (solução não é ótima) **então**
  - Calcula θ; {conforme equação VIII.1}
  - Atualiza multiplicadores associados às igualdades e desigualdades dualizadas σ_k;
  - Elimina todas as partições dualizadas com multiplicadores nulos;
  - Atualiza custos lagrangeanos;

fim;

senão

- retornar ao *branch-and-bound*;

- **fim se**;

Até que (iterações > MAXITER) ou (tivermos solução ótima);

fim. {procedimento}

Figura VIII.1: Relaxação lagrangeana c/ geração de restrições

Note que, se restrições violadas não forem encontradas em uma dada iteração do subgradiente, teremos obtido uma solução viável para o problema de roteamento. Para observar esse fato, basta observar se foram encontradas as restrições de eliminação de sub-rotas. Como demonstrado no teorema VI.1, temos um algoritmo exato (algoritmo VI.1) na identificação de sub-rotas violadas. Assim, se não existirem restrições de eliminação de sub-rotas violadas teremos chegado a uma solução viável para o problema de roteamento. Isto não significa entretanto, que tenhamos chegado a uma solução ótima do problema original. Devemos checar ainda se as condições de otimalidade foram ou não satisfeitas, ou seja, devemos verificar se o produto escalar entre o vetor dos multiplicadores de Lagrange e o vetor de subgradientes é igual a *zero* (Geoffrion[74], Fisher[81]).

No capítulo seguinte, mostraremos alguns dos resultados computacionais obtidos em nossa implementação quando atribuímos diferentes valores aos parâmetros do procedimento VIII.1.

VIII.3 - Clarke&Wright lagrangeano:

Apresentamos nesta seção, uma variação do algoritmo Clarke&Wright (capítuloI/ seção I.2) na geração de limites superiores para o problema de roteamento.

A idéia será utilizarmos o algoritmo Clarke&Wright a cada iteração do subgradiente, definindo portanto, uma heurística lagrangeana na determinação de boas soluções viáveis para o problema original. Para isto, entretanto, dois aspectos principais devem ser observados . Estamos trabalhando com custos positivos e negativos (custos lagrangeanos) e estamos fixando variáveis em 0 e 1 respectivamente.

Seja $\bar{c}_{ij}$ o custo lagrangeano associado a uma aresta (i,j) qualquer pertencente a E (conjunto de todas as arestas do grafo). Seja ainda E(N)$\subseteq$E, o conjunto de todas as arestas de E não adjacentes ao depósito. Representaremos por $VL_k(N)$, o subconjunto das arestas de E(N), ainda não fixadas em 0 e 1 na k-ésima iteração do subgradiente. As economias (ou *savings)* associadas a estas arestas poderão ser definidas da seguinte maneira:

$$s_{ij} = \bar{c}_{i0} + \bar{c}_{0j} - \bar{c}_{ij} \qquad \forall (i,j) \in VL_k(N)$$

Representaremos esta lista de economias (ou *savings*) por $S_k(N)$ .Diferentemente do algoritmo Clarke&Wright[64], antes de utilizarmos o procedimento na determinação de uma solução heurística, todas as arestas já fixadas em *um* (em uma dada iteração do subgradiente), poderão ser utilizadas na construção de um conjunto de rotas iniciais para o C&W lagrangeano. Na figura VIII.2, representamos um exemplo onde temos as arestas (1,2); (2,3); (0,1) e (6,7) fixadas em *um*:

Devemos buscar agora a formação de novas rotas utilizando-se a lista de economias. Observe que, se fixamos em *um,* alguma aresta (0,i) adjacente ao depósito, poderemos novamente eliminá-la se a economia s_{ij} (para algum j$\in$N e i$\neq$j) for selecionada em nossa

lista de economias $S_k(N)$ durante a execução do C&W lagrangeano. Este fato indesejado pode ser evitado se construirmos um vetor de "graus de liberdade" **GL**, associado aos clientes do grafo. Antes de executarmos o C&W lagrangeano propriamente dito, inicializamos **GL** fazendo GL(i) = 2 para i=1,..,n. Para todas as arestas (i,j) já fixadas em 1, decrementamos o "grau de liberdade" associado a **i** e **j** respectivamente. Note por exemplo, que os vértices 5 e 6 na figura VIII.2 tem graus de liberdade iguais a 2 e 1 respectivamente enquanto que o vértice 1 tem grau de liberdade 0. Isto permitirá que a aresta (0,1) não seja mais eliminada quando estivermos construindo novas rotas através do C&W lagrangeano.

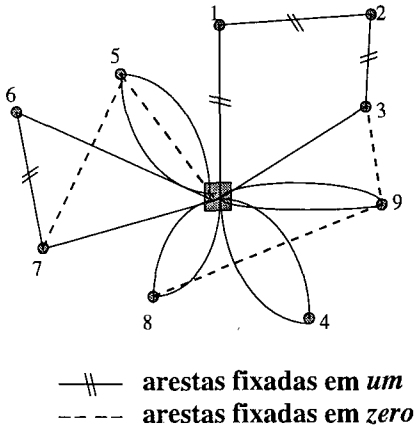


Figura VIII.2: Inicialização no C&W lagrangeano

Observe ainda no exemplo da figura VIII.2 que a aresta (0,5) (adjacente ao depósito) foi fixada em 0. Isto significa que, embora ela apareça em nossa rota inicial, ela deverá ser eliminada nas iterações seguintes do C&W lagrangeano. Isto poderá ser implementado facilmente atribuindo-se custo lagrangeano *infinito* à aresta (0,5).

Depois de construída a rota inicial, construímos novas rotas sempre selecionando as maiores economias de $S_k(N)$ em ordem decrescente, as economias associadas às arestas já eliminadas (e não adjacentes ao depósito) obviamente não serão analisadas. O processo é repetido até que K rotas tenham sido geradas (onde K é o número "mínimo" de rotas viáveis) ou a lista de *savings* esteja vazia.

Note que, ao fazermos a união de duas novas rotas, as economias negativas poderão ser selecionadas piorando o valor da nova solução heurística. Isto pode ocorrer caso uma

solução viável com K rotas ainda não tenha sido obtida. Pode-se gerar instâncias onde uma solução ótima com K+1 veículos (de mesma capacidade) tenha um valor menor que uma solução ótima com K veículos!

Antes de apresentarmos as principais etapas presentes em nosso procedimento C&W lagrangeano, vejamos inicialmente a seguinte definição:

Definição VIII.1: Chamaremos de *economias viáveis,* representada por $\overline{S}_k(N)$, a um subconjunto de economias de $S_k(N)$ tais que, p/ todas as arestas (i,j) pertences a $\overline{S}_k(N)$ temos:

(a) os vértices **i** e **j** não pertencem a mesma rota;

(b) a união das capacidades associadas às rotas que contém **i** e **j** respectivamente não excede a capacidade de cada veículo;

(c) os vértices **i** e **j** não são interiores (ou seja, **i** e **j** estão conectados ao depósito por pelo menos uma aresta). •

Temos então o seguinte procedimento, executado na k-ésima iteração do subgradiente:

PROCEDIMENTO VIII.2: Clarke&Wright lagrangeano;

Início

- Calcula GL a partir das arestas já fixadas em *um;*
- Geramos N_r rotas iniciais tomando-se as arestas fixadas em *um*;
- Calcula lista de economias $S_k(N)$;
- Ordena (ordem decrescente) a lista $S_k(N)$;

Repita

- (a,b) <-- aresta de $S_k(N)$ com maior economia;
- **Se** ($(a,b) \in \overline{S}_k(N)$) **então**

-Se (GL(a) > 0) e (GL(b) > 0) **então**

- Construímos nova rota passando por (a,b);

- GL(a) =GL (a) - 1;
- GL(b) = GL(b) - 1;
- $N_r = N_r - 1$;

- **fim;**

- fim;

reordena lista de economias $S_k(N)$;

Até que (N_r = K) ou ($S_k(N) = \varnothing$);

fim.

Figura VIII.3: Clarke&Wright lagrangeano

Observe que uma aresta (a,b) de $\overline{S}_k(N)$, poderá ser rejeitada na formação de uma nova rota se GL(a) = 0 ou GL(b) = 0. Isto significa que (0,a) ou (0,b) já foram fixadas em *um*, não podendo mais serem descartadas na formação de novas rotas.

Ao final da heurística lagrangeana poderemos ter um número de rotas maior do que K (número de veículos de nossa frota)! Caso isto ocorra, esta solução será descartada e a heurística lagrangeana novamente executada na iteração seguinte do subgradiente (vide procedimento VIII.1).

Se obtemos uma solução viável com K rotas em nosso C&W lagrangeano, fazemos uma busca local utilizando o algoritmo 3-optimal (Lin&Kernighan[73]) em cada uma das K rotas obtidas. Os resultados computacionais obtidos em nossa implementação do C&W lagrangeano serão apresentados no capítulo IX.

VIII.4 - Busca em árvore (*branch-and-bound*)

Como discutido no capítulo III (seção III.2), sempre que o limite inferior associado a um dado subproblema (ou folha) de uma árvore não corresponder ao valor de uma solução ótima daquele subproblema, particionamos seu conjunto viável associado em dois ou mais subconjuntos. Essa partição (ou *branching*) deverá ser realizada sempre levando em consideração características e informações relativas ao problema original.

Calculados os limites inferiores para cada um dos subproblemas gerados após o *branching* utilizamos uma estratégia de busca (pré-definida), escolhendo um novo subproblema a ser pesquisado. Em nosso caso, utilizamos a estratégia do melhor primeiro (ou *best-first*) que consiste basicamente em selecionarmos o subproblema de menor limite inferior entre todos os subproblemas ainda não pesquisados na árvore de busca (vide Murty[74], Horowitz e Sahni[78]). Chamaremos o subproblema selecionado na árvore de busca de **subproblema pai** e os subproblemas obtidos após o *branching* de **subproblemas filhos**.

No caso de problemas de minimização, se algum dos limites inferiores associados às folhas ainda não exploradas for superior ao valor da melhor solução viável obtida até o momento (na busca em árvore), o conjunto de soluções viáveis associado a cada uma destas folhas poderá ser descartado definitivamente. Isto permite que folhas sejam eliminadas implicitamente agilizando o processo de busca.

Na seção seguinte discutimos a estratégia de *branching* empregada em nosso trabalho e em seguida os principais aspectos relativos à implementação e estrutura de dados utilizada.

VIII.4.1 - Determinação da variável *branching*:

Em nosso trabalho, a árvore de busca é gerada sempre expandindo a folha pesquisada em duas novas folhas (ou subproblemas). Selecionamos uma aresta (variável *branching*) que será fixada em 0 e 1 respectivamente. A escolha desta aresta é realizada sempre a partir da extremidade de alguma rota parcial (ainda não finalizada) que é formada no decorrer do processo da busca em árvore. Ou seja, para determinarmos uma nova variável *branching*, verificamos a extremidade do caminho que tem início no depósito e que ainda não define uma rota completa (vide figura VIII.4). A partir desta extremidade, que chamaremos de **pta**, buscamos o próximo cliente a ser atingido:

Note que, após a utilização do procedimento que fixa variáveis, poderão existir algumas cadeias isoladas de arestas fixadas em *um*. Assim, o próximo nó (a partir de **pta**) deverá ser escolhido de maneira que o grau do próximo cliente seja menor ou igual a *um* e que a demanda associada a este novo percurso não exceda a capacidade de cada veículo. No

exemplo da figura VIII.4, se a aresta (pta,s) for escolhida como variável *branching*, o vértice **t** será a extremidade da nova rota (ainda não finalizada) que parte do depósito.

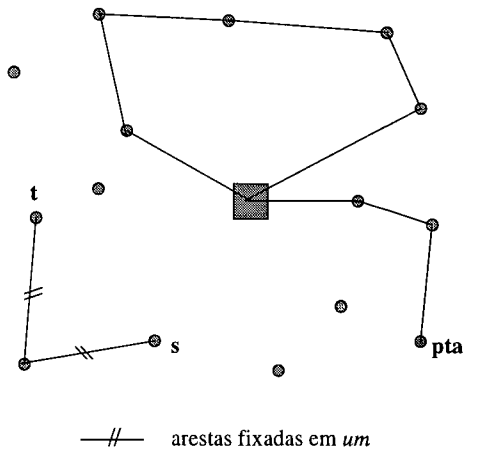


Figura VIII.4: Determinação de variáveis *branching*

Entre todas as arestas com uma das extremidades em **pta** e candidatas à variável *branching*, escolhemos aquela de menor custo lagrangeano.

Caso a ligação de **pta** com um novo cliente não seja mais possível, tomamos (pta, 0) como próxima variável *branching* (fechando uma nova rota). Se o custo associado a aresta (pta,0) for *infinito*, não será possível a determinação de uma nova variável *branching* a partir de **pta**. Teremos chegado portanto a uma solução inviável.

Outras situações de inviabilidade poderão ocorrer se, ao fecharmos uma nova rota, formarmos uma rota simples, ou ainda, se a demanda associada à rota parcial for inferior a $\sum_{i \in N} d_i - (K-1).b$, onde d_i é a demanda do cliente *i*, K o número de veículos utilizados e **b** a capacidade de cada veículo.

Se estivermos iniciando uma nova rota a partir do depósito, escolhemos o cliente **i*** de maior demanda entre todas aquelas arestas (0,i) incidentes ao depósito ainda não eliminadas.

Escolhida a próxima variável *branching*, fazemos esta variável igual a 0 e 1 respectivamente para determinação dos subproblemas filhos associados à folha pesquisada

(subproblema pai). O cálculo destes novos limites inferiores será discutido na próxima seção.

VIII.4.2 - Implementação e estrutura de dados utilizada

Nesta seção apresentamos os principais aspectos envolvidos em nossa implementação do algoritmo *branch-and-bound* e a estrutura de dados por nós utilizada.

Sempre que um limite inferior associado a um dado subproblema pai, e gerado pelo método subgradiente, não corresponder ao valor de sua solução ótima, devemos gerar dois novos subproblemas filhos (através da variável *branching*) computando em seguida os novos limites inferiores associados a estes subproblemas. Como veremos, o cálculo destes novos limites pode ser feito mais rapidamente aproveitando algumas das informações geradas durante a execução do método subgradiente.

Seja $\bar{z}_{lb}$ o valor do maior limite inferior associado ao subproblema pai obtido na k-ésima iteração do subgradiente. Se x_{ij}^k representa aquelas arestas que estão ou não na K-árvore mínima correspondente temos:

$$\bar{z}_{lb} = \sum_{ij \in E(N_0)} \bar{c}_{ij}^k x_{ij}^k + T_i^k \qquad \text{(VIII.3)}$$

onde $\bar{c}_{ij}^k$ é o vetor de custos lagrangeanos gerados na k-ésima iteração do subgradiente e T_i^k o somatório dos multiplicadores de Lagrange associados.

Os limites inferiores correspondentes aos subproblemas filhos são gerados através de uma lista de custos lagrangeanos $\bar{L}$ utilizada na obtenção de $\bar{z}_{lb}$. Fixando a variável *branching* em *zero* e *um* respectivamente, geramos duas novas K-árvores mínimas com a utilização de $\bar{L}$. O custo de cada uma destas K-árvores somado a T_i^k nos dá os novos limites inferiores associados aos subproblemas filhos.

Caso um dos novos limites inferiores obtidos tenha um valor menor que a melhor solução viável gerada durante a busca em árvore, adicionamos este limite inferior a uma lista que chamaremos de **lista de abertos** (representada por LA).

Ao escolhermos o subproblema pai na lista de abertos LA, fazemos primeiramente, um pré-processamento fixando novas arestas em *zero* antes da execução do subgradiente! Isto pode ser feito eliminando-se aquelas arestas ainda não fixadas em *zero* e *um* e que possuam uma extremidade de grau 2. Só após este pré-processamento aplicamos o método subgradiente sobre a lista resultante.

Apresentamos a seguir uma síntese das principais etapas presentes em nossa implementação do algoritmo *branch-and-bound.* A constante *menorzub* abaixo representa o valor da menor solução viável obtida durante a busca em árvore.

PROCEDIMENTO: *Branch-and-Bound*;
Início
- Calcula limite inferior (nó raiz);
- **Se** (solução não é ótima) **então**
  - faz pré-processamento tomando-se as arestas fixadas em *um*;
  - calcula var. *branching;*
  - gera 2 novas folhas e calcula limites inferiores associados;
  - **Se** (se esses limites forem menores ou iguais a *menorzub*) **então**
    - incrementa lista de abertos LA;
- **fim;**
- **Enquanto** (LA $\neq \varnothing$) **faça**
  - seleciona folha em LA (conforme estratégia *best-first*);
  - gera lista de arestas associada ao subproblema pai e faz pre-processamento;
  - calcula limite inferior (subproblema pai);

  Se (solução não é ótima) **então**
    - encontra variável *branching;*
    - gera 2 novas folhas (subpb. filhos) e calcula limites inferiores associados;
    - **Se** (se esses limites forem menores ou iguais a *menorzub*) **então**

- incrementa lista de abertos LA;

senão

atualiza *menorzub;*

- fim;

Figura VIII.5: Algoritmo *Branch&Bound*

Se após o término do subgradiente uma solução ótima não tiver sido obtida, geramos 2 novas folhas de acordo com a estratégia de *branching* utilizada e calculamos os novos limites inferiores associados salvando-se a lista de custos lagrangeanos correspondente ao maior limite inferior obtido durante o subgradiente. Como discutido anteriormente, esta lista será utilizada na determinação dos limites inferiores associados aos subproblemas filhos. Isto é feito, com a utilização da expressão VIII.3.

Vejamos agora alguns aspectos relativos à estrutura de dados utilizada. Determinado o subproblema pai a ser pesquisado, devemos recuperar primeiramente a lista de arestas associada a este subproblema. Isto pode ser feito tomando-se as variáveis *branching* no caminho da folha (associada ao subproblema pai) até a raiz da árvore de busca. Para isto, tomamos primeiramente a lista de arestas correspondente à raiz da árvore de busca. Cada vez que uma variável *branching* for percorrida trocamos a posição desta aresta com a primeira ou última aresta entre aquelas ainda não fixadas. Procedendo desta forma, geramos a lista associada a folha a ser pesquisada apenas quando o respectivo subproblema for selecionado na lista de abertos LA (vide figura VIII.6).

Os valores r_i *e* s_i, em cada folha, representam a última variável *branching* selecionada. Os valores *um* ou *zero*, indicam se a variável *branching* associada foi ou não utilizada. Finalmente, os valores ext1 e ext0 indicam a extremidade da rota parcial gerada no decorrer do processo de busca.

Como discutido anteriormente, ao utilizarmos uma estratégia de busca do tipo *best-first* selecionamos sempre uma folha (indicada por NEXT) cujo limite inferior associado seja o menor entre todos os limites inferiores na lista de abertos (apontada por FIRST).

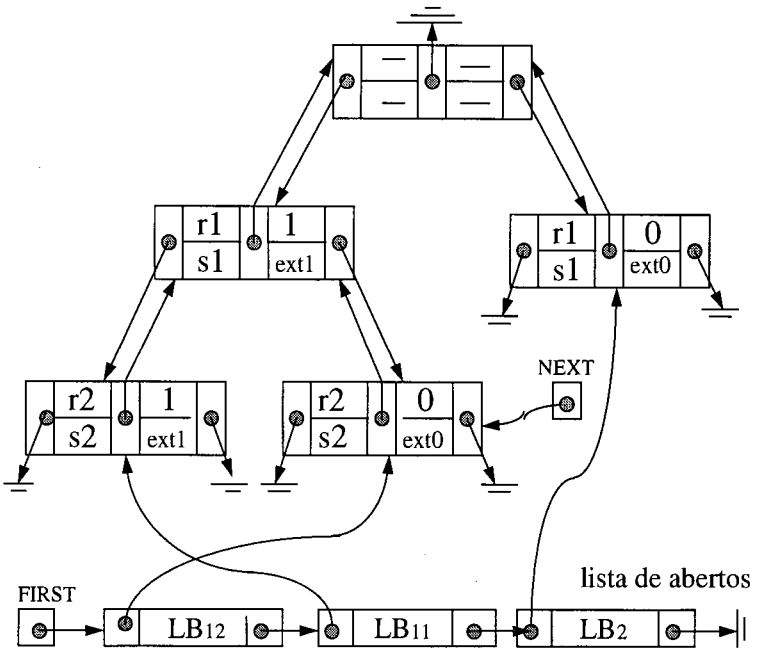


Figura VIII.6: Estruturas de dados p/ busca em árvore

Se LR é a lista de arestas associada à raiz de nossa árvore. Ao gerarmos a lista associada a folha apontada por NEXT, devemos percorrer o caminho da folha até a raiz tomando-se todas as variáveis *branching* presentes neste caminho.

Observando o exemplo da figura VIII.7, note que, ao geramos a lista associada ao limite LB_{12}, devemos trocar as arestas (r2, s2) com (p2,q2) e (r1, s1) com (p1, q1) respectivamente. Em seguida, decrementamos ext2 e incrementamos ext1:

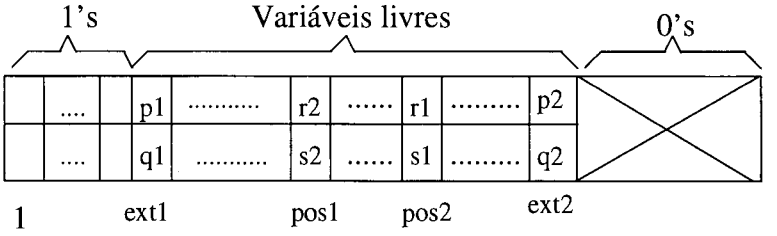


Figura VIII.7: Lista LR (associada à raiz)

Este tipo de abordagem irá permitir uma grande economia de memória já que não geramos explicitamente, a lista de arestas associadas à cada folha da árvore.

Capítulo IX

Resultados computacionais

IX.1 - Introdução:

Apresentamos neste capítulo alguns dos resultados computacionais obtidos em nossa implementação. Fazemos uma análise de algumas das instâncias disponíveis na literatura para o problema de roteamento de veículos com restrições de capacidade (vide Reinelt [91]).

Na seção IX.2, apresentamos 3 versões distintas na determinação dos limites inferiores. Na primeira versão (representada por S), utilizamos apenas as desigualdades de eliminação de sub-rotas. Na segunda versão (representada por SC), utilizamos simultaneamente as desigualdades de eliminação de sub-rotas e as desigualdades *comb*. Finalmente, na última versão (representadas por SCM), utilizamos as desigualdades de eliminação de sub-rotas, *combs* e *multistars* respectivamente. Estas 3 versões foram implementadas com a utilização de uma máquina RISK 6000 190.

Fazemos uma comparação entre alguns dos limites inferiores obtidos em nossa abordagem com as abordagens desenvolvidas por Fisher[94.b] (capítulo IV) e Miller[95] (capítulo I). Em seguida, fazemos uma avaliação de nossa heurística lagrangeana (C&W lagrangeano) e do procedimento que fixa variáveis.

Na seção IX.3, apresentamos alguns resultados computacionais obtidos por nossa implementação do algoritmo *branch-and-bound.* Para as instâncias aqui resolvidas, apresentamos o número total de nós gerados na árvore de busca bem como o tempo total de processamento utilizado.

IX.2 - Resultados computacionais (limites inferiores):

Nesta seção, são apresentamos alguns dos resultados obtidos em nossa implementação na determinação de limites inferiores para o problema de roteamento.

A tabela da figura IX.1, apresenta limites inferiores para alguns dos problemas testes da biblioteca TSPLIB (Reinelt[91]). Fazemos uma "comparação" entre os limites inferiores obtidos por Fisher[94.b] e as 3 versões distintas de nosso algoritmo, envolvendo, sub-rotas (S); sub-rotas e *combs* (SC); sub-rotas, *combs* e *multistars* (SCM) respectivamente. Os valores numéricos presentes nos nomes de cada instância (coluna (1)), indicam o número total de clientes a serem atendidos por cada frota de veículos. A quantidade de veículos em cada frota é representada na coluna (2). Os melhores limites superiores obtidos na literatura e associados a cada instância estão representados na coluna (7). Algumas soluções ótimas estão assinaladas com (*). Consideramos neste caso, assim como em Fisher[94.b], apenas distâncias euclideanas entre cada um dos clientes e o depósito.

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Prob.	**num. de veículos**	**Fisher**	**(S)**	**(SC)**	**(SCM)**	**Limite superior**
c50.dat	5	507.09	511.95	513.66	515.17	524.61 (*)
c75.dat	10	755.50	765.48	765.56	765.06	835.26
c100.dat	8	785.86	793.65	795.36	791.52	826.14
c150.dat	12	932.68	950.86	954.61	950.90	1028.42
c199.dat	16	1096.72	1149.70	1148.50	1146.63	1294.25
c100b.dat	10	817.77	817.54	817.66	817.85	819.56 (*)
f44.dat	4	720.76	721.20	722.55	721.68	723.54 (*)
f71.dat	4	237.76	238.51	238.44	238.65	241.97 (*)
f134.dat	7	1133.73	1154.22	1152.21	1150.70	1162.96

Figura IX.1: Limites inferiores (distância euclideanas)

Como discutido no capítulo VIII (seção VIII.2), devemos, no método subgradiente, atribuir diferentes valores a seus parâmetros. No método subgradiente implementado por Fisher, ele utiliza uma redução de 75% no tamanho do passo a cada 30 iterações do subgradiente sem atualização do limite inferior. Em nosso caso, consideramos uma redução de 85% no tamanho do passo a cada 50 iterações sem atualização do limite inferior. Fica difícil, neste caso, estabelecer quais os parâmetros ideais (no subgradiente) para o conjunto de problemas testes analisados. Consideramos taxas de redução (no tamanho do passo) que oscilavam entre 60% a 85%, e um número total de iterações (sem atualização dos limites inferiores) variando entre 40 e 60 unidades. Em todas as situações analisadas, obtivemos bons limites inferiores quando comparados àqueles obtidos por Fisher[94.b].

Utilizamos o mesmo número de iterações no subgradiente que nos testes realizados por Fisher[94.b]. As instâncias iniciadas pela letra ***c*** foram executadas em 3000 iterações do subgradiente, enquanto que as instâncias iniciadas por ***f*** foram utilizadas 2000 iterações.

Observando a tabela da figura IX.1, podemos constatar uma melhora dos limites inferiores, por nós obtidos, em quase todas as instâncias analisadas. Na coluna (3), temos os limites inferiores gerados por Fisher[94.b] com a utilização das restrições de eliminação de sub-rotas. Note que os resultados apresentados na coluna (4), mostram uma superioridade de nossa versão que utiliza também as desigualdades de eliminação de sub-rotas. Como discutido anteriormente, isto se deve ao fato de trabalharmos com um número maior de restrições candidatas à dualização. Na verdade, utilizamos uma formulação mais forte para o problema de roteamento que aquela apresentada por Fisher[94.b] (vide capítulo VI/ seção VI.2.1).

No gráfico da figura IX.2, buscamos fazer uma "avaliação" do desempenho das 4 versões implementadas para os problemas testes da figura IX.1. Colocamos em ordem crescente os limites inferiores obtidos e atribuímos (para cada uma das versões) pesos crescentes que variam de 1 a 4. O gráfico da figura IX.2, mostra o somatório destes pesos para cada uma das versões implementadas. Quanto maior o somatórios dos pesos, melhor a qualidade dos limites inferiores obtidos para o conjunto das instâncias avaliada.

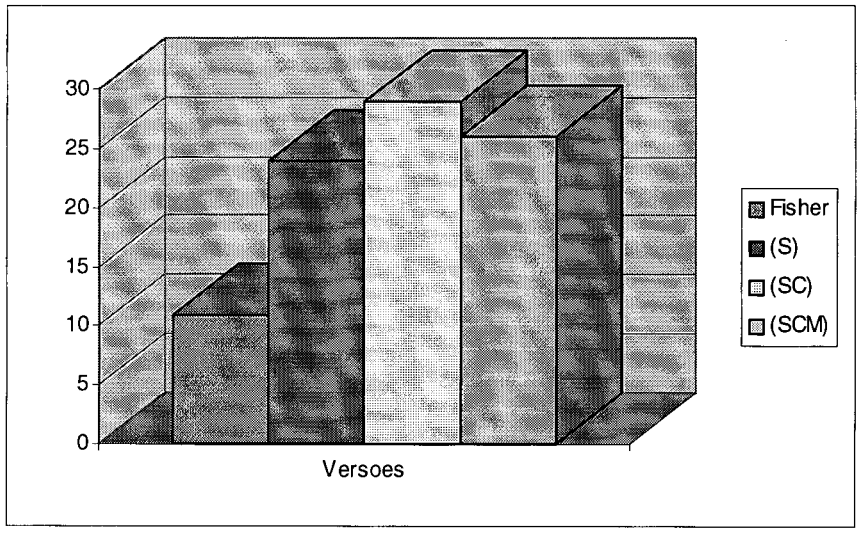


Figura IX.2: Somatório dos "pesos" de cada versão (limites inferiores)

Como discutido no capítulo III, ao menos teoricamente, a introdução de novas classes de restrições sempre proporciona novos limites inferiores maiores ou iguais aos anteriores. Em nosso caso entretanto, a introdução de novas classes de restrições (*combs* e *multistars*) não garante sempre um incremento estrito do limite inferior. Em função da lenta "convergência" do subgradiente para uma solução do problema dual lagrangeano, devemos estipular um número máximo de iterações para sua execução. Isto pode levar a situações onde a utilização de um número maior de classes de restrições não garanta uma melhora dos limites inferiores ao final do método subgradiente.

Na tabela da figura IX.3, apresentamos os tempos de processamento, calculados em minutos, para cada uma das instâncias da tabela IX.1. Vale ressaltar que as instâncias tratadas por Fisher (coluna 2) foram implementadas em uma máquina Apollo Domain 3000, o que dificulta a comparação entre os tempos de processamento.

(1)	(2)	(3)	(4)	(5)
Prob.	**Fisher**	**(S)**	**(SC)**	**(SCM)**
c50.dat	95.75	8.92	9.82	13.98
c75.dat	183.97	18.31	20.28	49.80
c100.dat	307.95	106.37	76.62	183.13
c150.dat	682.41	238.24	296.27	810.10
c199.dat	1186.00	504.24	568.97	2131.43

c100b.dat	259.64	55.13	66.55	136.00
f44.dat	49.74	20.47	13.97	51.67
f71.dat	105.33	102.05	102.77	169.47
f134.dat	253.84	635.68	711.00	1018.92

Figura IX.3: Tempo de processamento em minutos

Analogamente ao gráfico da figura IX.2, fazemos uma "avaliação" dos tempos de processamento para as 4 versões apresentadas. Colocamos os tempos de processamento em ordem crescente e atribuímos pesos que variam de 1 a 4 nesta ordem. O gráfico da figura IX.4 representa o somatório dos "pesos" para cada umas das versões analisadas. Quanto menor o somatório dos pesos, menor o tempo de processamento utilizado por cada versão.

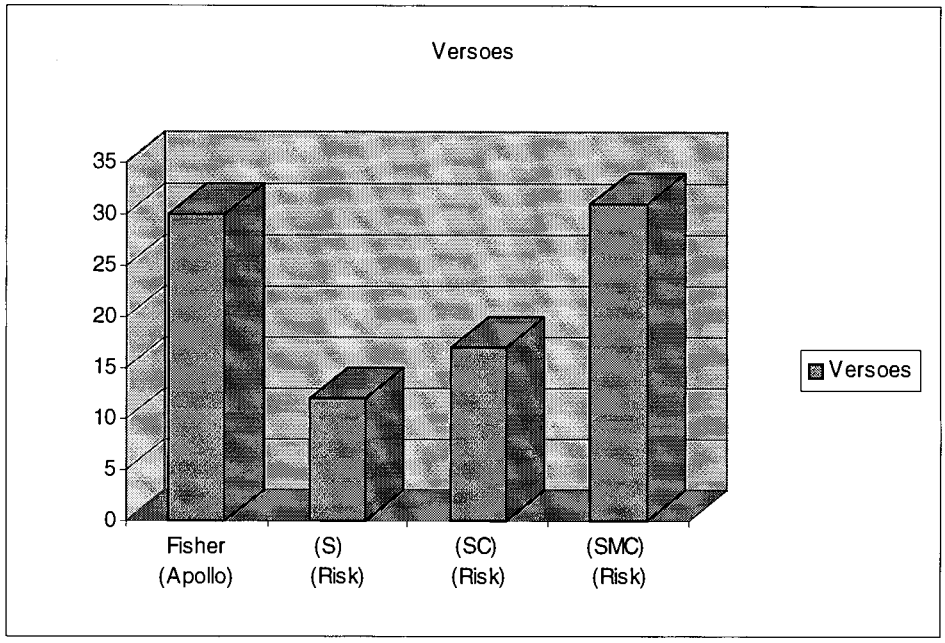


Figura IX.4: Somatório dos "pesos" de cada versão (tempo em minutos)

Note que a versão (S), levando em consideração a diferença entre as máquinas utilizadas, tem o menor tempo de processamento entre as 4 abordagens apresentadas.

Na tabela da figura IX.5, apresentamos os limites inferiores obtidos por nossa implementação, considerando agora, distâncias inteiras entre os clientes. Sempre que a

distância euclideana entre dois vértices for um valor fracionário **x**, ela será aproximada para o inteiro mais próximo (para $\lfloor x \rfloor$ *ou* $\lceil x \rceil$ respectivamente). Em todas as instâncias assinaladas com (*), foram utilizadas 3000 iterações do subgradiente. Para as demais instâncias utilizamos 2000 iterações.

Problemas	**Subt. (S)**	**S+C (SC)**	**S+C+M (SCM)**	**Limites Superiores**
c50.dat (*)	510	511	512	521
c75.dat (*)	759	761	762	830
c100.dat (*)	784	781	782	815
c150.dat (*)	938	939	943	1015
c199.dat (*)	1138	1121	1120	1280
c100b.dat (*)	819	820	818	820
f44.dat	723	723	722	724
f71.dat	233	233	233	237
f134.dat	1152	1153	1126	1162

Figura IX.5: Limites inferiores para instâncias "normalizadas" (dist. inteiras)

Todos os problemas apresentados acima, foram executados utilizando-se uma redução de 75% no tamanho do passo a cada 50 iterações do subgradiente sem atualização do limite inferior.

Na tabela da figura IX.6 a seguir, fazemos uma "comparação" entre nossa abordagem, que utiliza K-árvores mínimas, com a abordagem apresentada por Miller[95], que trabalha com o problema do *b-matching* na geração dos limites inferiores (vide capítulo I). Analogamente à tabela da figura IX.5, consideramos apenas distâncias inteiras entre os clientes e o depósito. Utilizamos novamente, uma redução de 75% no tamanho do passo, a cada 50 iterações do subgradiente sem atualização do limite inferior. Miller[95] por sua vez, utiliza um tamanho de passo que varia em função do número de atualizações dos limites inferiores.

Os valores numéricos presentes no nome das instâncias abaixo representam o número total de vértices associado ao problema (incluindo clientes e depósito).

(1) Problemas	(2) Miller	(3) (S)	(4) (SC)	(5) (SCM)	(6) Sol. ótima
eil7.vrp	114	114	114	114	114
eil13.vrp	268	277	277	277	277
eil22.vrp	374	375	375	375	375
eil23.vrp	569	569	569	569	569
eil30.vrp	509	509	509	509	534
eil33.vrp	826	830	830	831	835
eil51.vrp	505	510	511	512	521

Figura IX.6: Limites Inferiores (distâncias "normalizadas")

Apesar dos resultados numéricos satisfatórios obtidos por nossa abordagem, (apresentados acima), não podemos concluir diretamente que a relaxação apresentada por Miller[95] na geração dos limites inferiores (*b-matching*) seja inferior à relaxação que trabalha com a K-árvores mínimas. Em seu trabalho, Miller[95] utiliza uma variação do algoritmo subgradiente na geração dos limites inferiores. Desta maneira, vários fatores podem influir na qualidade da relaxação utilizada.

A tabela seguinte, da figura IX.7, representa a porcentagem de variáveis fixadas em *zero* e *um* respectivamente por nosso procedimento que fixa variáveis (capítulo VII). Os limites superiores utilizados foram tomados de problemas testes da literatura (biblioteca TSPLIB). Trabalhamos neste caso, apenas com distâncias euclideanas entre cada par de vértices associado ao problema. Executamos o subgradiente empregando somente as restrições de eliminação de sub-rotas (versão (S)) e adotando uma taxa de redução de 75% no tamanho do passo a cada 50 iterações sem incremento do limite inferior.

As porcentagens de 0's e 1's foram calculadas, respectivamente, sobre o número máximo de variáveis *nulas* e iguais a *um* no grafo completo associado. Note que, se K é o total de veículos da frota, então $n_z = |E| - (|V| + K)$ é o número máximo de variáveis que podem ser fixadas em *zero* e $n_u = |V| + K$, o número máximo de variáveis que podem ser fixadas em *um*.

Problemas	Limites inferiores (S)	Limites superiores	Porcent. de 0´s	Porcent. de 1´s
eil22.dat	375.28	375.28	91.19%	100%
eil23.dat	568.56	568.56	94.34%	100%
c50.dat	513.32	524.61	70.25%	0%
c75.dat	768.81	835.26	0%	0%
c100.dat	792.43	826.14	23.75%	0%
c100b.dat	818.07	819.56	93.89%	1.11%
c120.dat	1008.01	1042.11	8.88%	0%
f44.dat	723.24	723.54	93.49%	32.50%
f71.dat	238.63	241.97	76.84%	0%
tai75c.dat	1238.48	1291.01	4.65%	0%
tai75d.dat	1346.56	1365.46	56.11%	0%
tai100a.dat	1934.45	2047.90	0%	0%
tai100b.dat	1851.35	1940.61	0%	0%
tai100c.dat	1372.27	1406.20	9.71%	0%
tai100d.dat	1476.06	1581.25	0%	0%

Figura IX.7: Porcentagem de variáveis fixadas em *zero* e *um*

Os valores numéricos presentes no nome de cada instância, à exceção dos 2 primeiros problemas, representam o número de clientes a serem atendidos. As instâncias eil22.dat e eil23.dat, possuem 21 e 22 clientes respectivamente.

Observe que ocorre uma grande irregularidade com relação a quantidade de variáveis fixadas em *zero* e *um* respectivamente. No problema c100b.dat por exemplo (com 100 clientes), fixamos 93.89% de suas variáveis em *zero,* enquanto que no problema c75.dat (com 75 clientes) não fixamos nenhuma variável!

Finalizamos esta seção apresentado a tabela da figura IX.8. Fazemos uma comparação entre nossa heurística lagrangeana (C&W lagrangeano) e uma das heurísticas lagrangeanas apresentadas por Fisher[94.b] (seção IV.5 / figura IV.5). Consideramos apenas distâncias euclideanas entre os vértices associados ao problema.

(1)	(2)	(3)	(4)	(5)	(6)
Prob.	**Heur. Fisher**	**(S)**	**(SC)**	**(SCM)**	**Limite superior**
c50.dat	533.60	553.55	548.04	554.60	524.61
c75.dat	901.09	860.87	857.02	863.57	835.26
c100.dat	856.13	854.22	851.85	855.43	826.14
c150.dat	1150.20	1103.36	1097.93	1079.10	1028.42
c199.dat	1373.20	1374.88	1382.53	1386.83	1294.25
c100b.dat	924.39	819.56*	823.86	821.11	819.56
f44.dat	727.12	723.54*	723.54*	723.54*	723.54
f71.dat	257.11	248.25	253.58	248.54	241.97
f134.dat	1201.31	1192.24	1197.30	1197.30	1162.96

Figura IX.8: Avaliação do C&W lagrangeano

Observamos, durante a execução de nosso algoritmo, que os limites superiores eram atualizados, quase sempre, nas primeiras iterações do subgradiente. A freqüência de atualizações é menor a medida que os multiplicadores de Lagrange vão se "aproximando" dos multiplicadores ótimos duais. Desta forma, a variação dos custos lagrangeanos diminui e a geração de novas soluções heurísticas (de menor custo) se torna mais difícil. Para evitar esse esforço computacional desnecessário, executamos a heurística lagrangeana até a metade das iterações do subgradiente.

IX.3 - Resultados computacionais (*Branch-and-bound*):

Apresentamos nesta seção alguns resultados computacionais obtidos por nossa implementação na determinação de soluções exatas para o problema de roteamento. Como discutido anteriormente, utilizamos um *branching* binário e fazemos uma busca em árvore selecionando sempre a folha associada ao menor limite inferior da lista de abertos (*best-first*). Depois de selecionada a folha a ser pesquisada, executamos o subgradiente calculando, paralelamente, os limites superiores (C&W lagrangeano) e o procedimento que fixa variáveis, buscando sempre uma diminuição no tempo total de processamento.

Na tabela da figura IX.9, apresentamos alguns resultados computacionais avaliando os limites inferiores obtidos (nó raiz), o número de nós e o tempo total de processamento gasto na busca em árvore. As instâncias avaliadas por Miller[95], foram implementadas em uma máquina *Sun Sparc 2*. Novamente, para os problemas apresentadas abaixo, consideramos distâncias "normalizadas" entre os clientes e o depósito.

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Prob.	**Lim. Inf. (raíz) (SCM)**	**Núm. de Nós (B&B)**	**tempo (cpu)**	**Lim. Inf. (raíz) (Miller)**	**Núm. de Nós (B&B)**	**tempo (cpu)**	**Sol. ótima**
eil7.vrp	114	1	0,05 seg	114	1	0,22 seg	114
eil13.vrp	277	1	1,02 seg	268	60	14,83 seg	277
eil22.vrp	375	1	6,85 seg	374	46	33,94 seg	375
eil23.vrp	569	1	59,54 seg	569	1	4,07 seg	569
eil33.vrp	830	74	2,03 hs	826	379	9,16 min	835
eil51.vrp	512	1865	120,49 hs	505	7988	4,15 hs	521
c100a.dat	818	333	113,12 hs	-----	-----	-----	820
f44.dat	718	21	2,33 hs	-----	-----	-----	724

Figura IX.9: Avaliação da Busca em Árvore

Analisando-se o conjunto de problemas testes apresentados acima, observamos a melhor qualidade dos limites inferiores obtidos por nossa implementação. Isto provavelmente, contribuiu para a geração de um menor número de nós na árvore de busca na determinação de uma solução ótima para o problema original. A diferença no tempo de processamento entretanto, considerando-se, obviamente, a diferença entre as máquinas utilizadas, nos leva a concluir que o esforço computacional exigido por nossa abordagem foi maior que aquele despendido por Miller[95].

Ilustramos na figura IX.10, uma solução ótima para o problema *eil51.vrp* (biblioteca TSPLIB) com custo 521.

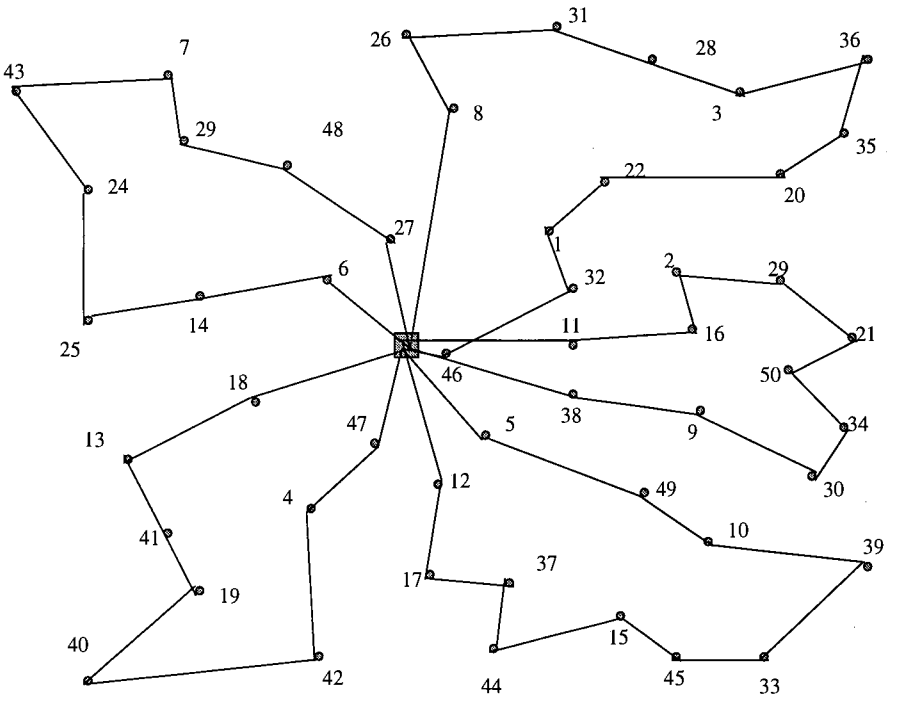


Figura IX.10: Solução exata p/ o problema *eil51.dat*

Finalmente, na tabela da figura IX.11, apresentamos resultados computacionais para algumas instâncias geradas por Christofides et al.[95]. Novamente, neste caso, fixamos um total de 2000 iterações para o método subgradiente. A heurística C&W lagrangeano é executada nas primeiras 1000 iterações do subgradiente. O total de clientes em cada instância são representados na coluna (1):

(1)	(2)	(3)	(4)	(5)	(6)
Prob.	**Num. de Veículos**	**Lim. Inf. (raíz)**	**Núm. de Nós (B&B)**	**tempo cpu**	**Sol. ótima**
c21.dat	4	374.3	1	8,30 seg	374.3
c21a.dat	6	478.6	424	84,65 min	494.7
c15a.dat	5	322.9	137	9,72 min	334.1
c15b.dat	5	267.4	134	10,93 min	277.9
c20a.dat	6	426.3	16	1,75 min	429.9
c20b.dat	4	346.4	100	14,98 min	357.6

Figura IX.11: Avaliação da busca em árvore.

Neste caso, a distância entre os vértices é calculada, multiplicando-se por 10 a distância euclideana, eliminando-se em seguida, sua parte fracionária. O número inteiro resultante é então dividido por 10.

Como ilustrado na tabela seguinte, um menor tempo de processamento pode ser conseguido no procedimento *branch-and-bound* se reduzimos o número total de iterações no subgradiente, de 2000, para aproximadamente, 900 iterações. A heurística lagrangeana (C&W lagrangeano) pode também ser executada, apenas nas primeiras iterações do subgradiente (em nosso caso 150 iterações). Obviamente outros parâmetros podem ser utilizados buscando-se ainda uma maior redução no tempo total de processamento.

(1)	(2)	(3)	(4)	(5)	(6)
Prob.	**Num. de Veículos**	**Lim. Inf. (raíz)**	**Núm. de Nós (B&B)**	**tempo cpu**	**Sol. ótima**
c21.dat	4	374.3	1	5,33 seg	374.3
c21a.dat	6	476.4	926	51,07 min	494.7
c15a.dat	5	321.0	127	2,65 min	334.1
c15b.dat	5	266.5	125	2,62 min	277.9
c20a.dat	6	424.7	27	1,38 min	429.9
c20b.dat	4	345.3	113	4,27 min	357.6

Figura IX.12: Avaliação da busca em árvore.

Observe que tivemos um ganho significativo no tempo de processamento embora o número de folhas de nossa árvore tenha aumentado em algumas das instâncias analisadas. A perda de qualidade nos limites inferiores (nó raiz) é compensada pela economia no tempo total de processamento na busca em árvore.

Conclusões

Trabalhamos na construção de um método exato, combinando, relaxação lagrangeana com a geração de desigualdades válidas na determinação de limites inferiores para o problema de roteamento. Utilizamos uma abordagem, onde a solução de cada problema relaxado se resume na obtenção de K-árvores mínimas com um número fixo de arestas incidentes ao depósito.

Entre as principais contribuições de nosso trabalho, podemos citar, a utilização de um procedimento exato, a cada iteração do subgradiente, na identificação das desigualdades de eliminação de sub-rotas. Este procedimento nos permitiu trabalhar com um número maior de restrições candidatas à dualização. Em outras palavras, obtemos uma formulação mais forte que aquela utilizada por Fisher[94.b] para o problema de roteamento de veículos. Desenvolvemos também procedimentos heurísticos para identificação de *combs* e *multistars* violadas pela solução do problema lagrangeano.

Apresentamos um procedimento para fixação de variáveis a partir de uma K-árvore mínima com 2K arestas incidentes ao depósito. Esta abordagem é interessante já que, para uma grande quantidade de problemas, pode influir positivamente na qualidade dos limites (inferiores e superiores) gerados pelo método subgradiente, bem como proporcionar uma diminuição no tempo total de processamento.

Os limites superiores foram gerados utilizando-se uma nova heurística lagrangeana para o problema de roteamento. Desenvolvemos uma variação do procedimento clássico de Clarke e Wright[64], buscando utilizar, da melhor maneira possível, as informações geradas pelo procedimento que fixa variáveis em *zero* e *um* respectivamente. Acreditamos se tratar de uma heurística lagrangeana bastante promissora, já que, ela pode ser ainda incrementada, utilizando-se outras informações presentes na solução do problema lagrangeano.

Trabalhos futuros podem ser realizados no sentido de melhorar a qualidade dos "cortes" obtidos, bem como a introdução e análise de novas classes de desigualdades válidas para o problema original. Outras estratégias de *branching* podem ser utilizadas buscando-se uma melhora na qualidade da busca em árvore realizada.

Referências Bibliográficas

- Aboudi R., Hallefjord A., Jörnsten K.[1991], " A Facet Generation and Relaxation Technique Applied to an Assignment Problem with Side Constraints"; European Journal of Operational Research 50 : 335 - 344.

- Aboudi R., Nemhauser G.L. [1991], " Some facets for an assignment problem with side constraints"; Operations Research vol. 39, n°2, 244 - 250.

- Ahuja R. K., Magnanti T. L., Orlin J. B., [1993], "Network flows: theory, algorithms and applications"; Prentice Hall, Englewood Cliffs; New Jersey 07632.

- Araque J.R., [1990] "Solutions of a 48-city vehicle routing problem by branch-and-cut", Research Memorandum 90-19, Purdue University.

- Araque J.R., Hall L., Magnanti T., [1990] "Capacitated trees, capacited routing and associated polyhedra", discussion paper 9061, CORE, Louvain la Neuve.

- Araque J.R., Kudva G., Morin T.L., Pekny J.F. [1994], " A Branch and Cut Algorithm for Vehicle Routing Problems"; Annals of Operations Research 50 : 37 - 59.

- Augerat P. [1995] "Approche polyédrale du problème de tournées de véhicules", Thèse doctorale, Institut National Polytechinique de Grenoble.

- Bachem A., Grötschel M., [1982] "New aspects of polyhedral theory", Modern Applied Mathematics - Optimization and Operations Research (North - Holland publishing company).

- Balas E., Christofides N., [1981] "A restricted lagrangean approach to the traveling salesman problem"; Mathematical Programming 21, 19 - 46.

- Balas E., Padberg M. [1979] "Set partitioning - a survey in combinatorial optimization", Wiley, London.

- Balinski M., Quandt R., [1964] "On an integer program for a delivery problem", Operations Research 12, 300-304.

- Barahona F., Anbil R. [1997] " The volume algorithm: producing primal solutions with a subgradient method", Techinical Report, IBM T. J. Watson Research Center, NY 10598.

- Bazaraa M., Jarvis J., Sherali H., [1977], "Linear Programming and Network Flows"; John Wiley & Sons.

- Beasley J. .E. [1990] " OR-Library: distributing test problems by eletronic mail"; Operational Research Society, vol 41, n°11, pp. 1069 - 1072.

- Beasley J. .E. [1993] "Modern Heuristic Techniques for Combinatorial Problems"; capítulo 6, Blackwell Scientific Publications.

- Bertsimas D., Orlin J. B., [1994]] "A technique for speeding up the solution of the lagrangean dual", Mathematical Programming (63), 23-45.

- Bodin L., Golden B.L., Assad A.A., Ball M. [1983] " Routing and Scheduling of Vehicles and Crews: The State of the Art"; Comp. Oper. Res. 10 : 62 - 212.

- Campello R., Maculan N. [1994] " Algoritmos e heurísticas: desenvolvimento e avaliação de performance"; Ed. da Univ. Federal Fluminense (EDUFF/FURNAS), Rio de Janeiro.

- Campos V., Corberan A., Mota E. [1991] " Polyhedral results for a vehicle routing problem"; European Journal of Operations Research 52 : 75 - 85.

- Christofides N., Mingozzi A., Hadjiconstantinou E., [1995] "An exact algorithm for the vehicle routing problem based on the set partitioning formulations", Department of Mathematics, University of Bologna, Italy.

- Christofides N., Mingozzi A., Toth P. [1981.a], " Exact Algorithms for the Vehicle Routing Problem, based on spanning tree and shortest path relaxation"; Mathematical Programming 20 : 255 -282.

- Christofides N., Mingozzi A., Toth P. [1981.b], "State-space relaxation procedures for the computation of bounds to routing problems", Networks 11, 145-164.

- Christofides N., [1985], "The traveling salesman problem", chapter 12 - Vehicle routing, pag. 431-448 / John Wiley & Sons Ltd.

- Christofides N., Mingozzi A., Toth P. [1979], " The vehicle routing problem "; John Wiley & Sons.

- Christofides N., Eilon S. [1969], " An algorithm for the vehicle dispaching problem "; Operations Research Quartely 20, n.3.

- Chvátal V. [1973] "Edmonds polytopes and weakly hamiltonian graphs", Mathematical Programming 5, 29-40.

- Clarke G., Wright J.W. [1964] " Scheduling of vehicles from a central depot to a number of delivery points"; Operations Research 12 : 568 - 581.

- Cornuejols G., Fonlupt J., Naddef D,. [1985] "The traveling salesman problem on a graph and some related integer polyhedra", Mathematical Programming (33) , 1-27.

- Cornuejols G., Harche F., [1993] "Polyhedral study of the capacited vehicle routing problem"; Mathematical Programming 60, 21-51.

- Dantzig [1951], "Application of the simplex method to a transportation problem", John Wiley, NewYork, pages 359-373.

- Dantzig, G; Fulkerson, F; Johnson S. [1954] "Solution of a large-scale traveling-salesman problem"; Operations Research 2, 393-410.

- Desrochers M., Desrosiers J., Solomon M. [1992] "A new optimization algorithm for the vehicle routing problem with time windows", Operations Research 40, 342-354.

- Desrosiers J., Soumis F., Desrochers M, [1984] "Routing with time windows by column generation", Networks 14, 545-565.

- Eilon S., Watson-Gandy, Christofides [1971] ; "Distribution management: mathematical modelling and practical analysis", Griffin, London.

- Escudero L.F., Guignard M., Malik K., [1994] "A lagrangian relax-and-cut approach for the sequential ordering problem with precedence relationships"; Annals of Operations Research 50, 219-237.

- Ferreira C. E., Wakabayashy Y. [1996] "Combinatória poliédrica e planos-de-cortes faciais"; Instituto de Computação da Unv. Estadual de Campinas, 10^{a} Escola de Computação.

- Fisher M. [1981] "The lagrangian relaxation method for solving integer programming problems", Managemente Science - Vol. 27 / 1-18.

- Fisher M. [1994.a] "A polynomial algorithm for the degree constrained k-tree problem"; Operations Research, vol. 42, 776 - 780.

- Fisher M. [1994.b] " Optimal solution of vehicle routing problems using minimum k-trees"; Operations Research, vol. 42, n°4, 626 - 642.

- Foster B., Ryan D. [1976] "An integer programming approach to the vehicle scheduling problem", Operational Reasearch Quartely 27, 367-384.

- Garey M. R., Johnson D. S. [1979] "Computers and intractability: a guide to the theory of NP-completeness" , Freeman, New York.

- Gendreau M., Laporte G., Potvin Y. [1994] "Metaheuristics for the vehicle routing problem", Technical Report, Centre de Recherche sur les transports, Université de Montréal.

- Geoffrion A.M. [1974], " Lagrangean relaxation for integer programming" ; Math. Progr. Study 2 : 82-114

- Glover F., Klingman D. [1974] "Finding minimum spannig trees with a fixed number of links at a node"; Colloquia Mathematica Societatis Janos Bolyai, Hungary, 425-439.

- Glover F., Klingman D. [1984] "Note on admissible exchanges in spanning trees"; Advances in Management Studies 3.2, 101-104.

- Golden B.L., Magnanti T.L. Nguyen H.Q. [1977] " Implementing vehicle routing algorithms"; Networks 7 : 113 - 349.

- Golden L. B., Stewart W. R., [1984] "A lagrangean relaxation heuristic for vehicle routing", European Journal of Operational Research (15) 84-88.

- Gouveia L. [1995] "A result on projection for the vehicle routing problem", European Journal of Operational Research 85, 610-624.

- Grötschel M., Padberg M., [1979] "On the symmetric traveling salesman problem: I-II", Mathematical Programming 16, 265-280 e 281-302.

- Grötschel M., Lovász L., Schrijver [1988] "Geometric algorithms and combinatorial optimization", Springer, Berlin.

- Grötschel M., Holland O. [1991] "Solution of large-scale symmetric traveling salesman problems", Mathematical Programming (51), 141-202.

- Grümbaum B., [1967] "Convex polytopes", Wiley, London, New York.

- Guignard M., [1992] "Efficient cuts in lagrangian relax-and-cuts schemes", Report n. 92-09-07, Decision Sciences Department, The Wharton School, University of Pennsylvania.

- Hadjiconstantinou E., Christofides N., Mingozzi A.[1995] "A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxations", Freight Transportation, Ann. Oper. Res. 61, 21-44.

- Harche F., Rinaldi G. [1991] "Vehicle routing", private communication, 1991.

- Held M., Crowder H., Wolfe P., [1974] "Validation of subgradient optimization"; Mathematical Programming 6, 62-88.

- Hill S. P. [1995] "Branch-and-Cut methods for the symmetric capacitated vehicle routing problem", Ph.D thesis, Curtin University of Technology, Australia.

- Horowitz E., Sahni S., [1978] "Fundamentals of Computer Algorithms", Computer Science Press, Inc.

- Houck, Pickard, Queyrane, Vemuganti [1980] "The traveling salesman problem as a constrained shortest path problem: theory and computational experience"; Operations Research vol.17, n°s 2 e 3.

- Jamsa, K. [1992] "Microsoft C: Dicas, segredos e truques"; Makron Books.

- Jamsa, K. [1993] "Jamsa's 1001 C/C++ tips"; Jamsa Press.

- Jünger M., Reinelt G., Rinaldi G. [1995] "The traveling salesman problem"; Handbooks in OR & MS, Vol. 7.

- Jünger M., Reinelt G., Thienel S. [1992] "Probably good solutions for the traveling salesman problem", IWR, Heidelberg - Preprint 92 - 31.

- Khachian L. G., [1979] "A polynomial time algorithm in linear programming", Doklady Akademia Nauk USSR, 244 (5) , 1093 - 1096.

- Kernighan B., Ritchie D., [1988] "C: A linguagem de programação padrão ANSI" ; Editora Campus.

- Kohl N., [1995] "Exact methods for time constrained routing and related scheduling problems", Ph.D Thesis - Technical University of Denmark.

- Kohl N., Madsen O., [1997] "An optimization algorithm for the vehicle routing problem with time windowns based on lagrangean relaxation", Operations Research, vol 45 n.3, may-june, pp 395-406.

• Kruskal J. B., [1956] "On the shortest spanning subtree of a graph and the traveling salesman problem", Proc. Amer. Math. Soc. 7, 48-50.

• Laporte G., Osman I. [1994] "Routing problems: a bibliography", Centre de Recherche sur les transports - Université de Montreal.

• Laporte G., Nobert Y. [1984.a] "Comb inequalities for the vehicle routing problem", Methods of Operations Research (51) 271 - 276.

• Laporte G., Nobert Y. [1987] "Exact algorithms for the vehicle routing problems", Surveys in Combinatorial Optimization, Annals of Discrete Mathematics, 11 North-Holland. Amsterdam, pp 147-184.

• Laporte G., Nobert Y. [1984.b] "Exact algorithms for the vehicle routing problem", Annals of Discrete Mathematics, 31: 147-184.

• Laporte G. [1992.a], " The traveling salesman problem: an overview of exact and approximate algorithms"; European Journal of Operations Research 59 : 231 - 247.

• Laporte G. [1992.b], " The vehicle routing problem: an overview of exact and approximate algorithms"; European Journal of Operations Research 59 : 345 - 358.

• Laporte G., Nobert Y., Desrochers M. [1984] " Two exact algorithms for the distance constrained vehicle routing problem"; Networks 14 : 161 - 172.

• Laporte G., Nobert Y., Desrochers M. [1985] " Optimal routing under capacity and distance restrictions"; Operations Research 33 : 1050 - 1073.

- Laporte G., Mercure H., Nobert Y. [1992] " A branch and bound algorithm for a class of asymmetrical vehicle routeing problems"; Operational Research Society , vol. 43, n° 5, 469 - 481.

- Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B. [1985], " The traveling salesman problem: a guided tour of combinatorial optimization"; Wiley, New York.

- Lenstra, Rinnooy, Kan [1981] "Complexity of vehicle routing and scheduling problems", Networks (11), 221-227.

- Letchford A. N., Eglese R. W. [1996] "New cutting planes for vehicle routing problems", working paper, The Management School, Lancaster University, England.

- Leung J. M. Y., Magnanti T. L., [1989] "Valid inequalities and facets of the capacitated plant location problem"; Mathematical Programming 44, 271 - 291.

- Lin S., Kernighan B.W. [1973] " An effective heuristic algorithm for the traveling salesman problem"; Operations Research 21 : 498 - 516.

- Lucena A.P. [1986] " Exact solution approaches for the vehicle routing problem"; Ph.D. Thesis, Depart. of Management Science, Imperial College of Science and Technology, University of London, U.K.

- Lucena A.P., Beasley J. [1995] "Branch and cut algorithms "; Advances in Linear and Integer Programming, Edited by J. Beasley , Oxfor Lecture Series in Mathematics and its Applications 4.

- Lucena A. P., [1996] "Steiner problem in graphs: lagrangean relaxation and strong valid inequalities "; Laboratório Nacional de Computação Científica (relatório técnico).

- Macambira E. M.,[1996] "Uma abordagem de programação linear inteira para o problema da clique máxima com peso nas arestas", Dissertação de Mestrado do Instituto de Computação - Universidade de Campinas.

- Maculan N., Campello R., Lopes L. [1978] "Relaxação lagrangeana e programação inteira"; Relatório Técnico - COPPE/UFRJ.

- Maculan N., Salles J.C. [1991] "A lower bound for the shortest hamiltonean path in directed graphs"; Operations Research Spectrum 13 : 99 - 102.

- Magnanti T., Wolsey L.,[1995] "Optimal trees", M. O. Ball et al., Eds Handbooks em OR & MS, vol 7, Elsevier Science B.V. pp. 503-615.

- Marstern R. [1974] "An algorithm for large scale set partitioning problem", Management Science, 20; p.779.

- Martello S., Toth P. [1990a], "Lower bounds and reduction procedures for the bin-packing problem"; Discrete Applied Mathematics (28), 59-70.

- Martello S., Toth P. [1990b] "Knapsack problems: algorithms and computer implementation", John Wiley & Sons.

- Miller D. L., Pekny J. F., [1994] "A staged primal-dual algorithm for perfect b-matching with edge capacities"; ORSA Journal on Computing, in press.

- Miller D. L., [1995] "A matching based exact algorithm for capacited vehicle routing problems"; ORSA Journal on Computing, vol. 7, N. 1.

- Minkowski H., [1911] "Gesammelte abhandlugen", edited by D. Hilbert, Teuber, Leipzig.

- Minoux M [1986] " Mathematical Programming - Theory and Algorithms " ; Wiley, New York.

- Mitchel J. [1995], "Interior point algorithm for integer programming" ; Advances in Linear and Integer Programming. Edited by J. Beasley , Oxfor Lecture Series in Mathematics and its Applications 4.

- Murty K., [1976] "Linear and combinatorial programming", John Wiley & Sons inc.

- Nelson M. D., Nygard K. E., Griffin J. H., Shreve W. E. [1985] "Implementation techniques for the vehicle routing problems", Computers & Operations Research 12/ 273-283.

- Nemhauser G.L., Wolsey L.A. [1988] " Integer and combinatorial optimization"; Wiley-Interscience, New York.

- Nobert Y., [1982] "Construction d'algorithmes optimaux pour des extensions au probleme du voyageur de commerce", Doctoral Thesis, Departement d'Informatique et Recherche Operationnalle, University of Montreal, Canadá.

- Ochi, L. S., [1989] "Contribuições à solução de problemas de percursos e sequenciamento de veículos", Tese de Doutorado - COPPE/UFRJ.

- Orloff C. [1976] "Route-constrained fleet scheduling", Transportation Science 10, 149-168.

- Padberg M., Rinaldi G., [1989] "A branch-and-cut approach to a traveling salesman problem with side constraints", Management Science, vol.35, n. 11, pp 1393 - 1412.

- Padberg M., Rinaldi G. [1990] " Facet Identification for The Symmetric Traveling Salesman Polytope"; Mathematical Programming 47 : 219 - 257.

- Padberg M., Rinaldi G. [1987] " Optimization of a 532-city Symmetric Traveling Salesman Problem by Branch and Cut"; Operations Research Letters 6 : 1 - 7.

- Padberg M., Rinaldi G. [1991] " A Branch and Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems"; SIAM Review Vol. 33, n° 1, pp. 60 - 100.

- Paessens H. [1988] "Savings algorithm for the vehicle routing problem", European Journal of Operational Research 34/ 336-344.

- Papadimitriou C.H., Steiglitz K. [1978] "Some examples of difficult traveling salesman problem", Operations Research 26, 434-443.

- Papadimitriou C.H., Steiglitz K. [1982] " Combinatorial optimization: algorithms and complexity"; Prentice-Hall, Englewood Cliffs, NJ.

- Pulleyblank W. R., [1983] "Polyhedral combinatorics", Department of Combinatorics and Optimization, University of Waterloo, Ontario N2L 3G1, Canada.

- Rao M., Zionts S., [1968] "Allocation of transportation units to alternative trips - a column generation scheme with out-of-kilter subproblems", Operations Research 16, 52-63.

- Rangel M. S. N. [1995] "Solving integer programming problems using preprocessing and cutting planes: theory and implementation of branch-and-cut"; Ph.D Thesis, Department of Mathematics and Statistics, Brunel University, West London.

- Reeves C. [1993], "Modern Heuristic Techniques for Combinatorial Problems"; Blackwell Scientific Publications.

- Reinelt G. [1991] "TSPLIB - A traveling salesman problem library", ORSA Journal on Computing 3, 376-384.

- Roy, Jonker [1986] "Traveling Salesman and assigment algorithms: design and implementation", Tese de doutorado - Lutherse Kerk Universtity (Amsterdan/Holanda).

- Ruland K. S, [1995] "Polyhedral solution to the pickup and delivery problem"; Ph. D thesis, Departament of Systems Science and Mathematics, Sever Institute of Technology, Washington University.

- Savage S., Weiner P., Bagchi A., [1976] "Neighborhood search algorithms of guaranteeing optimal traveling salesman tours must be inefficient", J. Comput. Syst. Sci., 12 / 25-35.

- Schläfli L., [1901] "Theorie der vielfachen kontinuität", Denkschrift de Schweizerischen Naturforschenden Gesellschaft 38, 1 - 237.

- Schmitz E. A., Teles A. A. S., [1988] "Pascal e técnicas de programação: incluindo turbo pascal"; Livro Técnicos e Científicos editora S.A.

- Schrijver A., [1986] "Theory of linear and integer programming"; Wileu, Chichester.

- Shor N. Z., [1977] "Cut-off method with space extention in convex programming problems", Kibernetik 13 (1) 94 - 95.

- Silveira Filho O. T., [1995] "Linguagem C" , (Apostila) - Depto. de Computação / Univ. Federal Fluminense.

- Steinitz E., Radermacher H., [1976] "Vorlesugen über die theorie de r polyeder", Springer, Berlin, Heidelberg, New York.

- Syslo M., Deo N., Kowalik J., [1983] "Discrete Optimization Algorithms with Pascal Programs", Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632.

- Szwarcfiter J. L., Markezon L., [1994] *"Estruturas de Dados e seus Algoritmos"*, LTC Editora , Rio de Janeiro /RJ.

- Volgenant T., Jonker R., [1982] "A branch-and-bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation", European Journal of Operational Research , 9/ 83-89.