UMA HEURÍSTICA LAGRANGEANA PARA O PROBLEMA DA ÁRVORE GERADORA CAPACITADA DE CUSTO MÍNIMO

Jorge Bergson Carvalho da Silva

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

Aprovada por:

Prof. Nelson Maculan Filho, D.Sc

Prof. Abílio Pereira de Lucena Filho, Ph.D.

Prof. Márcia Helena Costa Fampa, D.Sc.

Prof. Carlos Alberto de Jesus Martinhon, D.Sc.

Prof. Luis Alfredo Vidal de Carvalho, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
JULHO DE 2002

SILVA, JORGE BERGSON CARVALHO DA

Uma heurística lagrangeana para o problema da árvore geradora capacitada de custo mínimo [Rio de Janeiro] 2002

XIV, 98 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2002)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 - Árvores Geradoras Capacitadas de Custo Mínimo

2 - Relaxação Lagrangeana

3 - Busca Local

4 - Heurística Lagrangena

I. COPPE/UFRJ II. Título (série)

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UMA HEURÍSTICA LAGRANGEANA PARA O PROBLEMA DA ÁRVORE GERADORA CAPACITADA DE CUSTO MÍNIMO

Jorge Bergson Carvalho da Silva

Julho/2002

Orientadores: Nelson Maculan Filho
Abílio Lucena Pereira Filho

Programa: Engenharia de Sistemas e Computação

Esta tese introduz uma heurística Lagrangeana para o problema da Árvore Geradora Capacitada de Custo Mínimo. A heurística é inicializada por um algoritmo guloso construtivo e é complementada por uma estratégia de busca local. A heurística opera dentro de um ambiente Lagrangeano onde limites duais (inferiores) são gerados por um algoritmo do tipo *Relax and Cut.* Nesse procedimento, limites duais são utilizados para modificar os custos de entrada da heurística gulosa. Testes para fixação de variáveis, utilizando limites primais e duais, foram desenvolvidos e se mostraram bastante efetivos quando a distância entre os limitantes não era muito grande. Através desse procedimento conseguimos provar a otimalidade de uma instância a muito em aberta. Testes computacionais foram realizados e indicaram que os nossos resultados melhoraram, sensivelmente, os limites inferiores para uma determinada classe de instâncias da literatura. Os limites superiores obtidos parecem depender fortemente da qualidade dos limites inferiores que lhes servem de entrada, e são dominados apenas por uma metaheurística recentemente proposta.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A LAGRANGEAN HEURISTIC FOR CAPACITATED MINIMUM SPANNING TREE PROBLEM

Jorge Bergson Carvalho da Silva

July/2002

Advisors: Nelson Maculan Filho
Abílio Pereira de Lucena Filho

Department: Computing and Systems Engineering

A Lagrangian based heuristic for the Capacitated Minimum Spanning Tree Problem is introduced in this thesis. The heuristic is initialized with a greedy construction phase which is complemented with local search. The proposed heuristc operates from whithin a Lagrangian framework where dual (lower) bounds are obtained through a *Relax and Cut* procedure. In this scheme, dual bounds are used to modify input data for the greedy heuristic. Tests for fixing out suboptimal variables were developed and proved effective when duality gaps are small. In this way, optimality of a long standing open instance was proved. Computational tests indicated that our lower bounds significantly improved upon the best known bounds for a class of instances from the literature. Our upper bounds appear to be strongly correlated with the lower bounds used to generate them and are only dominated by upper bounds obtained by a recently proposed metaheuristic.

"A coisa mais bela que o homem pode experimentar é o mistério. É essa emoção fundamental que está na raíz de toda ciência e toda arte."

Albert Einstein

Agradecimentos

A Deus.

Aos meus queridos pais Tarcízio e Marisete, pelo amor, incentivo e compreensão initerruptamente dispensados a mim, sobretudo, nesta época de minha ausência.

A todos os meus parentes, próximos ou não, por tudo que já fizeram por mim, e pela ajuda inestimável na minha formação. E além disso, pelo grande incentivo, que, de longe ou de perto, manifestado ou não, estiveram sempre a me oferecer quando eu decidi enfrentar mais esse desafio.

A minha eterna e querida professora, Teresinha Claudino, por ter me assistido de maneira exemplar, quando eu ainda iniciava meus estudos nas séries iniciais do primeiro grau.

Ao querido professor, Marcos Negreiros, pela apoio e inestimável carinho dispensados a mim, desde que começamos a trabalhar juntos na graduação. Ele é o maior responsável pela minha entrada no mestrado. A ele, ofereço meus sinceros agradecimentos por tudo que ele fez por mim.

Ao meu ilustríssimo orientador, Nelson Maculan, por sua inestimável generosidade, sua atenção, e, sobretudo, por me acolher como um pai. Além de, ainda, me fazer desfrutar de seus sábios conselhos, de seu profissionalismo, e, principalmente, de seu exemplo.

Ao meu co-orientador, Abílio Lucena, pela excepcional orientação, pela paciência, e, principalmente, pela imensa generosidade em passar uma parte de seus grandes conhecimentos a mim. Conhecimentos esses, não apenas técnicos, mas também de vida.

Aos professores de graduação. Em especial, aos professores Wamberto Vasconcelos (um dos responsáveis pela minha entrada no mestrado), Araújo Lima e Plácido Pinheiro que além de terem me assistido durante a graduação, foram sempre profissionais exemplares. De maneira especial também, ao professor Everardo Maia, pelo incentivo para que eu encarasse mais esse desafio.

A todos os meus colegas de graduação, em especial, aos amigos Allan, Flávio e Marcos Bendor, pelo excelente convívio e ajuda em algumas oportunidades. Este último, além de companheiro oficial dos "bonecos", tornou-se praticamente um irmão.

A todos os meus amigos que conviveram comigo durante todo esse tempo do mestrado, companheiros de repúblicas ou não. De maneira bastante especial, aos amigos, praticamente irmãos, Tibérius, Prata e Elder, pelo apoio e ajuda imprescindíveis durante minha caminhada no curso.

A todos os meus professores da Escola Normal Rural (onde cursei primeiro grau) e do Colégio Diocesano Padre Anchieta (onde cursei o segundo grau) pela excelente assistência enquanto eu estudava nesses colégios.

Aos amigos de Limoeiro do Norte, Ceará, minha terra natal.

A todos funcionários e professores do Programa. De forma especial, às funcionárias Fátima, Cláudia Prata e Gercina, de atenção e paciência invejáveis. À Lourdes, cujo cafezinho teve grande importância para a conclusão deste trabalho.

Aos professores Luis Gouveia (Universidade de Lisboa, Portugal), Pedro Martins (Instituto Superior de Contabilidade e Administração de Coimbra, Portugal) e Antonio Frangioni (Universidade de Pisa, Itália) por terem me fornecido informações valiosas para a concepção deste trabalho.

À Universidade Estadual do Ceará.

À Universidade Federal do Rio de Janeiro.

À CAPES pelas bolsas de estudos concedidas, tanto na época de graduação quanto durante o período de mestrado.

Aos meus queridos pais,

Tarcízio e Marisete.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Seja $G = (V, E)$ um grafo não-direcionado, onde $V = \{v_i : i \in I\}$ é um conjunto de nós indexados por $I = \{1, 2, 3, ..., n\}$ e $E = \{e = [v_i, v_j],\ i, j \in I, i < j\}$ é um conjunto de arestas. O nó v_1 é denominado *nó central* ou *raiz* e $\overline{V} = V \setminus \{v_1\}$ é um conjunto de nós *terminais* indexados por $\overline{I} = I \setminus \{1\}$. Seja c_e o *custo* associado a cada aresta $e \in E$ e $q_i > 0$, $i \in \overline{I}$, o *peso* ou *demanda* associado a cada terminal. Dada uma árvore geradora $T = (V, E')$, $E' \subseteq E$, de G, considere as componentes distintas obtidas ao se remover de T o nó central v_1. Cada uma dessas componentes define, necessariamente, uma árvore ou um nó isolado. Quando a soma dos pesos dos nós de cada componente não exceder Q, a árvore geradora T define uma solução viável para o Problema da Árvore Geradora Capacidada de Custo Mínimo (PAGCM). O custo dessa solução será igual à soma dos custos de suas arestas, isto é, $\sum_{e \in E'} c_e$. Quando nenhuma outra solução viável para o PAGCM tiver custo inferior, T irá definir, obviamente, uma solução ótima para o problema.

Na literatura, é feita uma distinção entre o caso em que todas as demandas são idênticas (caso homogêneo), e aquele onde as demandas podem diferir (caso heterogêneo). Geralmente, o problema é chamado de PAGCM apenas no primeiro caso. Note que o PAGCM com demandas idênticas pode sempre ser reduzido, sem perda de generalidade, a outro em que as demandas são unitárias. Dessa forma, assumiremos que em um PAGCM com deman-

das idênticas, tais demandas possuem valor 1. Nesse caso, podemos melhor descrevê-lo como sendo o problema de encontrar uma Árvore Geradora de Custo Mínimo com raiz em v_1 onde cada subárvore incidente à raiz contenha no máximo Q nós. Nesse trabalho, denominamos Problema da Árvore Geradora Capacitada de Custo Mínimo tanto o caso envolvendo demandas idênticas quanto aquele em que as demandas são heterogêneas.

O PAGCM aparece em muitas aplicações relacionadas ao projeto de redes elétricas e de telecomunicações [29]. Em particular, um problema fundamental no projeto de topologias de redes de telecomunicações que podemos modelar como um PAGCM é o de projetar uma rede de computadores centralizada. Sejam dados um computador (processador) central, um conjunto de terminais remotos com demandas específicas (que representam o tráfego que deve fluir entre o computador central e cada terminal), e todas as ligações possíveis entre pares de terminais e entre os terminais e o computador central. Nessa aplicação, os custos associados às ligações possíveis são todos não-negativos e o objetivo é encontrar uma topologia de custo mínimo para fazer as conexões necessárias entre os terminais e o computador central. Em tal topologia, o tráfego máximo em cada ligação não deve exceder a capacidade de cada porta do computador central (isto é, uma ligação direta do computador a um terminal) e o custo total (somatório dos custos de todas as ligações utilizadas) deve ser minimizado. Gavish [31] descreve, detalhadamente, todo o contexto em que o PAGCM aparece no projeto de topologias de rede de telecomunicações.

É importante ressaltar também que limites inferiores para o PAGCM levam a limites inferiores para o Problema de Roteamento de Veículos Capacitado (PRVC) [69, 71], um dos problemas mais estudados em Otimização Combinatória. Em particular, um algoritmo Lagrangeano para o PRV foi proposto por Martinhon *et al.* [57], no mesmo estilo daquele aqui utilizado.

Papadimitriou [59] demonstrou que o PAGCM é NP-Árduo quando $q_i = 1$, $\forall i \in \overline{I}$, e $2 < Q < n/2$. Um grande número de heurísticas foi

proposto para o problema, sendo Esau-Williams [23] a mais conhecida delas (veja Amberg *et al.* [9] e Gavish [31] para uma revisão das heurísticas propostas para o problema até o momento). Recentemente, foram também propostas algumas abordagens utilizando metaheurísticas para obter limites superiores para o PAGCM (veja Amberg *et al.* [9], Sharaiha *et al.* [64], Patterson *et al.* [60], Ahuja *et al.* [8, 5], Souza *et al.* [65]). Essas abordagens conseguiram melhorar sensivelmente os limites superiores conhecidos para quase todas as instâncias da literatura [14]. Dentre esses trabalhos, vale destacar especialmente, aqueles de Ahuja *et al.* [8, 5], que igualaram ou melhoraram os limites superiores para todas as instâncias testadas. Esse fato é ainda mais evidente para instâncias com demandas heterogêneas (melhoria em quase todas as instâncias consideradas).

Alguns métodos exatos e/ou esquemas para obtenção de limites inferiores para o PAGCM foram propostos na literatura. Em geral, os algoritmos exatos conseguem provar a otimalidade de instâncias com até 50 nós. Gouveia e Martins [39] apresentam um apanhado geral sobre a maioria desses métodos, incluindo os trabalhos de Gavish [28, 29, 30], os algoritmos exatos propostos por Chandy e Lo [17], Malik e Yu [55] e Toth e Vigo [69] (neste, o número de subárvores é fixado através de uma restrição adicional e somente instâncias com demandas heterogêneas são tratadas), o esquema Lagrangeano utilizado em Gouveia [35] e o método de planos-de-corte proposto por Hall [42]. Além desses, podemos citar ainda os métodos baseados em planos-de-corte de Gouveia e Martins [39, 40]. É importante ressaltar que com o resultado dos trabalhos de Hall [42] e Gouveia e Martins [39, 40], muitos dos limites superiores em Amberg *et al.* [9] e Ahuja *et al.* [8] tiveram sua otimalidade provada (isso vale apenas para instâncias com demandas idênticas), como será melhor detalhado posteriormente.

Neste trabalho, propomos uma Heurística Lagrangeana para o PAGCM. Na apresentação dos algoritmos, não utilizamos nenhuma estrutura de dados particular que não seja comum na literatura. Assim, evitamos comentários acerca das estruturas de dados utilizadas. Porém, vale enfatizar que o uso

de uma estrutura de dados eficiente foi crucial para o bom desempenho de nossos algoritmos.

No segundo capítulo, apresentamos uma formulação multifluxos para o PAGCM juntamente com algumas desigualdades válidas para o problema. No terceiro capítulo, é feito um histórico dos algoritmos *Relax and Cut* e são também tratadas questões relativas às suas implementações. O algoritmo *Relax and Cut* que propomos para o PAGCM é descrito nesse capítulo. Ainda no terceito capítulo, apresentamos algumas formas de reduzir o tamanho da entrada de instâncias para o PAGCM.

No quarto capítulo, apresentamos a heurística Esau-Williams [23]. No quinto, fazemos uma revisão das principais vizinhanças de busca propostas na literatura (algumas delas foram utilizadas em nosso trabalho). O sexto capítulo mostra algumas das principais metaheurísticas para o problema.

No sétimo capítulo, descrevemos algumas heurísticas Lagrangeanas que implementamos, porém, nosso estudo está focado em apenas uma delas. No oitavo capítulo, confrontamos os nossos resultados computacionais com os melhores resultados da literatura. E, por fim, no nono capítulo, fazemos um resumo de nossos resultados e apresentamos sugestões de trabalhos futuros.

Capítulo 2

Uma Formulação Multifluxos para o PAGCM e Algumas Desigualdades Válidas

Nesse capítulo, apresentaremos uma formulação multifluxos para o PAGCM e algumas desigualdades válidas para o problema.

Optamos por uma formulação multifluxos para o PAGCM porque tal formulação tende a ser mais *"forte"* que as formulações monofluxo existentes para o problema. Essa afirmação se aplica quando ambos os tipos de formulações não são reforçadas por desigualdades válidas (veja Rardin e Choe [62] e Magnanti e Wolsey [54]). Uma razão adicional é o fato de nossa proposta de trabalho ser centrada no uso de Relaxação Lagrangeana (veja Beasley [12]) o que não impõe limitações severas de memória diante do uso de formulações multifluxos (ao contrário, por exemplo, de Programação Linear).

Existem diversas formas de gerar formulações multifluxos. Aquela que utilizaremos aqui foi inspirada em uma formulação proposta por Maculan [52] para o problema da arborescência de custo mínimo. Formulações nessa mesma linha foram propostas por Beasley [13], Maculan [53] e Wong [70] para o Problema de Steiner em Grafos. Tais formulações foram adaptadas por Gouveia [37] para um Problema de Steiner diante de restrições adicionais e por Gouveia [36] para um problema da Árvore Geradora de Custo Mínimo

diante de restrições adicionais. Antes de apresentarmos a formulação aqui utilizada, devemos, no entanto, reformular o PAGCM em um grafo direcionado.

Um grafo direcionado $D = (V, A)$, onde A é o conjunto de arcos, pode ser obtido do grafo original não-direcionado $G = (V, E)$, através da substituição de cada aresta $e = [v_i, v_j] \in E$ por dois arcos (v_i, v_j) e (v_j, v_i), de mesmo custo $d_{ij} = d_{ji} = c_e$. Além disso, assumimos que em qualquer solução viável não existem arcos entrando no nó central v_1, ou seja, substituimos cada aresta $[v_1, v_i]$, $\forall i \in \overline{I}$, somente por um arco (v_1, v_i). É importante ressaltar que apesar de termos direcionado a formulação, não vamos tratar instâncias direcionadas do PAGCM neste trabalho.

2.1 Formulação

Dado o grafo direcionado $D = (V, A)$ e custos $\{d_{ij} : (v_i, v_j) \in A\}$, seja $q_i > 0$, a demanda associada ao nó v_i, $i \in \overline{I}$ e Q, a capacidade imposta a toda arborescência que emana de v_1. Associe a cada arco $(v_i, v_j) \in A$, uma variável de fluxo, z_{ij}^k, $k \in \overline{I}$, para indicar se o fluxo do tipo (endereçado ao nó) k pelo nó central v_1 irá passar pelo arco (v_i, v_j) ou não. Introduza, ainda, uma variável de decisão x_{ij} representando a presença ou não do arco (v_i, v_j) na solução do problema. Uma formulação multifluxos para o PAGCM pode então ser dada por:

$$\min \sum_{(v_i,v_j)\in A} d_{ij}x_{ij} \tag{2.1}$$

sujeito a:

$$\sum_{(v_j,v_i)\in A} x_{ji} = 1, \quad \forall i \in \overline{I} \tag{2.2}$$

$$\sum_{(v_i,v_j)\in A} z_{ij}^k - \sum_{(v_h,v_i)\in A} z_{hi}^k = \begin{cases} +1, & \text{se } i = 1 \\ -1, & \text{se } i = k \\ 0, & \text{se } i \neq 1, i \neq k, \forall i \in I, \\ & \forall k \in \overline{I} \end{cases} \tag{2.3}$$

$$z_{ij}^k \leq x_{ij}, \qquad \forall (v_i, v_j) \in A, \ \forall k \in \overline{I} \tag{2.4}$$

$$\sum_{\forall k \in \overline{I}} z_{ij}^k q_k \leq Q x_{ij}, \ \forall (v_i, v_j) \in A \tag{2.5}$$

$$z_{ij}^k \geq 0, \qquad \forall (v_i, v_j) \in A, \forall k \in \overline{I} \tag{2.6}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall (v_i, v_j) \in A \tag{2.7}$$

onde as restrições (2.2) asseguram que cada nó v_i, $i \in \overline{I}$, deverá ter exatamente um arco de entrada na solução. As restrições (2.3) garatem a conservação de fluxo para os nós de *passagem*. As desigualdades (2.4) impõem que uma unidade de fluxo do tipo k só passará pelo arco (v_i, v_j), se o arco (v_i, v_j) estiver na solução, ou seja, $z_{ij}^k = 1$ *somente se* $x_{ij} = 1$. O conjunto de restrições (2.2), (2.3), (2.4), (2.6) e (2.7) assegura que existirá um caminho do nó central v_1 até cada um dos outros nós. Diante de custos não-negativos é fácil mostrar que as variáveis x_{ij}, $\forall (v_i, v_j) \in A$, vão induzir a formação de uma arborescência geradora de V. As restrições (2.5) impõem que em cada arco, dessa arborescência, não passará uma quantidade de fluxo maior do que a capacidade previamente estabelecida, Q. E, por fim, isoladamente, (2.6) estabelece a não negatividade das variáveis de fluxo e (2.7) estabelece a integralidade das variáveis de decisão.

O número de variáveis inteiras (binárias) na formulação (2.1)-(2.7) é da ordem de $O(n^2)$, o número de variáveis contínuas é da ordem $O(n^3)$ e o número de restrições é da ordem $O(n^3)$. Isso torna a formulação, quando utilizada diretamente (isto é, sem um tratamento mais elaborado), pratica-

mente intratável para resolver instâncias de médio e grande portes. Para contornar essa restrição, utlizaremos, a exemplo de Beasley [13], Relaxação Lagrangeana. Nesse sentido, iremos dualizar, Lagrangeanamente, algumas das restrições em (2.1)-(2.7). Esse esquema será melhor detalhado no próximo capítulo. Na seção que se segue, apresentaremos algumas desigualdades válidas para o PAGCM que visam tornar (2.1)-(2.7) mais *"forte"*.

2.2 Desigualdades Válidas para o PAGCM

Apresentaremos agora, algumas desigualdades válidas, propostas na literatura, para o PAGMC. Posteriormente, quando apresentarmos os resultados computacionais obtidos, detalharemos o impacto da inclusão de tais desigualdades na formulação (2.1)-(2.7).

2.2.1 Desigualdades para Limitação de Fluxo nos Arcos (DLFA's)

As desigualdades abaixo foram propostas em Gavish [28]. Elas são uma forma de tornar as desigualdades (2.5) mais *"fortes"*. Assumindo-se que $q_1 = 0$, temos:

$$\sum_{\forall k \in \overline{I}} z_{ij}^k q_k \leq (Q - q_i) x_{ij}, \ \forall (v_i, v_j) \in A. \tag{2.8}$$

Para provarmos a validade de (2.8) considere os seguintes casos:

Caso 1: Se $(v_i = v_1)$, a quantidade máxima de fluxo que pode entrar no nó v_j é igual a Q. Isso é claramente válido, uma vez que v_1 é o nó central e possui uma demanda igual a zero. Nesse caso, (2.8) é idêntica a (2.5).

Caso 2: Se $(v_i \neq v_1)$, o fluxo no arco (v_i, v_j), $v_i, v_j \in \overline{V}$ não pode ser superior a $(Q - q_i)$, pois a demanda q_i do nó v_i já teria sido satisfeita. Note que

a existência de um fluxo superior a $(Q - q_i)$ em $(v_i, v_j) \in A$ implicaria na existência de um arco chegando a v_i com um fluxo superior a Q, acarretando, assim, inviabilidade. ∎

Da mesma forma que as desigualdades (2.8) podem ser entendidas como uma generalização de (2.5), as desigualdades a serem apresentadas a seguir podem ser entendidas como uma generalização de (2.8). Essas desigualdades foram propostas em [38] para uma variação do PAGCM que impõe além do custo d_{ij} de cada arco (v_i, v_j), um custo para cada unidade de fluxo enviada através do arco (v_i, v_j). Em [38], elas foram utilizadas tanto em uma formulação monofluxo como em uma formulação multifluxos que, aliás, é bastante similar a que aqui utilizamos. As desigualdades são descritas por:

$$\sum_{\forall k \in \overline{I}} z_{ij}^k q_k \leq (Q - q_i - q_p)x_{ij} + q_p(1 - x_{ip} - x_{pi}), \qquad (2.9)$$

$$\forall (v_p, v_i, v_j) \in \overline{V}, v_p \neq v_i, v_p \neq v_j, v_i \neq v_j.$$

O objetivo dessa família de desigualdades é limitar, ainda mais, a quantidade de fluxo capaz de passar através de um arco $(v_i, v_j) \in A,\ i, j, \in \overline{I}$. Elas se baseiam na possibilidade de um arco diferente de (v_1, v_i) incidir em v_i numa solução viável do problema ou ainda na possibilidade de um outro arco, partindo de v_i e diferente de (v_i, v_j), existir nessa solução.

Para mostrarmos a validade de (2.9), note que $x_{ip} + x_{pi} \leq 1,\ \forall\, v_i, v_p \in \overline{V}$, $i \neq p$, e considere os seguintes casos:

Caso 1: Se $(x_{ij} = 1)$ e $(x_{ip} = 1$ ou $x_{pi} = 1)$, então (2.9) implica em

$\sum_{\forall k \in \overline{I}} z_{ij}^k q_k \leq (Q - q_i - q_p)$, que é claramente válido para a topologia implícita nessa situação;

Caso 2: Se $(x_{ij} = 1)$ e $(x_{ip} = x_{pi} = 0)$, então

$\sum_{\forall k \in \overline{I}} z_{ij}^k q_k \leq (Q - q_i - q_p) + q_p$, o que nos leva a

$\sum_{\forall k \in \overline{I}} z_{ij}^k q_k \leq (Q - q_i)$, que também é válido;

Caso 3: E por fim, se $(x_{ij} = 0)$, então $z_{ij}^k = 0, \forall k \in \overline{I}$, (2.4). Isso faz com que o lado esquerdo de (2.9) fique nulo (0). Com isso, temos dois subcasos:

3.1: Se $(x_{ip} = x_{pi} = 0)$, então o lado direito da restrição fica igual a q_p que é maior do que zero e portanto válido.

3.2: Se $(x_{ip} = 1$ ou $x_{pi} = 1)$, o lado direito da restrição fica igual a 0 que também é válido. ∎

De forma a retratar, de maneira clara, as desigualdades (2.9) como uma forma mais *"forte"* (isto é, um *lifting* de (2.8)) das desigualdades (2.8), podemos reescrevê-las como:

$$\sum_{\forall k \in \overline{I}} z_{ij}^k q_k \leq (Q - q_i)x_{ij} + q_p(1 - x_{ip} - x_{pi} - x_{ij}),, \qquad (2.10)$$
$$\forall (v_p, v_i, v_j) \in \overline{V}, v_p \neq v_i, v_p \neq v_j, v_i \neq v_j.$$

2.2.2 Desigualdades Generalizadas de Eliminação de Subrotas

Considere agora, as desigualdades válidas introduzidas por Laporte e Nobert [49] para o Problema de Roteamento de Veículos [57]. Essas desigualdades são denominadas Desigualdades Generalizadas de Eliminação de Subrotas (DGES's):

$$\sum_{(v_i, v_j) \in A(S)} x_{ij} \leq \mid S \mid - \lceil d(S)/Q \rceil \qquad S \subseteq \overline{V}, \text{ e } \mid S \mid \geq 2, \qquad (2.11)$$

onde $A(S)$ define o conjunto dos arcos com ambas as extremidades definidas por nós em S e $d(S) = \sum_{v_i \in S} q_i$

Araque *et al.* [11] estabeleceram que as DGES's definem facetas do poliedro (isto é, a envoltória convexa das soluções viáveis para o problema) da Árvore Geradora Capacidada (AGC) *não* direcionado, se $\mid S \mid$ não é múltiplo de Q. Posteriormente, Zhang [72] estabeleceu o mesmo resultado para o caso direcionado. Vale frisar que esses resultados são válidos somente para o caso em que todas as demandas são idênticas.

É importante ressaltar ainda que quando as demandas são heterogêneas, o valor de $d(S)$, da forma como foi descrito acima, representa um limite inferior para o número de subárvores viáveis necessárias para acomodar os nós (terminais) em S. Nesse caso, o melhor limite inferior possível para $d(S)$ é dado pelo valor de uma solução ótima para o Problema de *BinPacking* (PBP) [56] definido pelos nós/itens $\{v_i \in S\}$ de pesos $\{q_i : v_i \in S\}$ e por *bins* de capacidade Q. Porém, para evitarmos o esforço computacional de resolver exatamente esse problema NP-Árduo e baseando-se na experiência de Hall [42] ("quase sempre, $d(S)$ calculado de forma relaxada foi igual ao valor ótimo da resolução de um PBP para o conjunto S"), optamos por computar $d(S)$ da forma relaxada.

As DGES's têm sido bastante utilizadas na modelagem de Problemas de Projeto de Topologias de Redes [28, 30] e do Problema de Roteamento de Veículos [57]. Isso deve-se ao fato delas se mostrarem, quase sempre, capazes de reduzir consideravelmente os "gaps" de dualidade existentes (veja Hall [42]).

Capítulo 3

Um Algoritmo *Relax and Cut* para o PAGCM

Nesse capítulo, apresentaremos um histórico e os principais aspectos relacionados a implementação de Algoritmos *Relax and Cut*. Mostraremos, ainda, o esquema Lagrangeano que utilizamos, um procedimento para separação de DGES's violadas e o Algoritmo *Relax and Cut* que implementamos para o PAGCM. E, por fim, apresentaremos alguns procedimentos para reduzir o "tamanho da entrada" de instâncias do problema.

3.1 Algoritmos *Relax and Cut*

Assuma que uma formulação (0-1 binária, por simplificação) de um problema de Otimização Combinatória NP-Árduo é dada. Considere ainda, que um número exponencial de desigualdades podem ser incluídas nessa formulação. Tal formulação, pode ser descrita genericamente como

$$\min\{c^{\top}x : Ax \geq b, x \in X\}, \tag{3.1}$$

onde $x \in \mathbb{B}^m$ é um vetor de variáveis, $c^{\top} \in \mathbb{R}^m$, $b \in \mathbb{R}^p$, $A \in \mathbb{R}^{p \times m}$ e, finalmente, $X \subseteq \mathbb{B}^m$ é uma região viável associada ao problema. Assuma ainda, como é comum em Relaxação Lagrangeana, que

$$\min\{c^{\top}x : x \in X\} \tag{3.2}$$

é um problema de fácil (tempo polinomial) resolução. Por outro lado, assuma, o que não é muito comum em Relaxação Lagrangeana, que p é exponencial em m. Sem nos importarmos com esse fato, dualizemos

$$\{a_i x \geq b_i : i = 1, 2, ..., p\} \tag{3.3}$$

de forma Lagrangeana com $\lambda^{\top} \in \mathbb{R}^p_+$ sendo o vetor correspondente dos multiplicadores de Lagrange. A Otimização por Subgradientes (OS) pode ser utilizada, a princípio, para resolver

$$\max_{\lambda^{\top} \geq 0}\{\min\{(c^{\top} - \lambda^{\top}A)x + \lambda^{\top}b : x \in X\}\} \tag{3.4}$$

e obter o melhor limite dual possível neste caso. A otimização é comumente conduzida aqui de forma iterativa com os multiplicadores sendo conveninentemente atualizados de forma que se possa convergir para o valor ótimo de (3.4). Para uma melhor compreensão, faremos aqui, uma breve revisão do Método do Subgradiente (MS) [43] que utilizaremos na implementação do nosso Algoritmo *Relax and Cut* para o PAGCM. Maiores detalhes sobre o MS podem ser encontrados em Beasley [12].

Em uma certa iteração do MS, para um dado conjunto de multiplicadores de Lagrange $\lambda^{\top} \geq 0$, seja $\overline{x}$ uma solução ótima para

$$\min\{(c^{\top} - \lambda^{\top}A)x + \lambda^{\top}b : x \in X\}. \tag{3.5}$$

Denote por z_{lb} o valor correspondente dessa solução, e por z_{ub} um limite superior conhecido para (3.1). Adicionalmente, seja $g \in \mathbb{R}^p$, os subgradientes

associados as restrições relaxadas. Dada uma solução ótima $\overline{x}$ para (3.5), os subgradientes g associados à $\overline{x}$ são definidos por

$$g_i = (b_i - a_i\overline{x}),\ i = 1, 2, ..., p \tag{3.6}$$

Na literatura (veja [26]) os multiplicadores de Lagrange são costumeiramente atualizados através da determinação inicial de um "tamanho de passo" θ

$$\theta = \frac{\alpha(z_{ub} - z_{lb})}{\sum_{i=1}^{p} g_i^2} \tag{3.7}$$

onde α é um número real que assume valores no intervalo $(0, 2]$. Após a obtenção de θ, seguimos computando

$$\lambda_i^{\top} \equiv \max\{0; \lambda_i^{\top} + \theta g_i\}, i = 1, 2, ..., p, \tag{3.8}$$

e passamos para a iteração seguinte do MS.

Se levarmos em consideração as condições que foram impostas inicialmente (em particular que p é exponencial em m), a implementação das fórmulas (3.6)-(3.8) pode não ser tão simples quanto parece. A razão disso é o (potencialmente enorme) número de desigualdades que, muito provavelmente, teriam que ser dualizadas.

As desigualdades em (3.3), em qualquer iteração do MS, podem ser divididas em três grupos. O primeiro grupo é o das desigualdades que são violadas por $\overline{x}$. O segundo é o das desigualdades que possuem multiplicadores não nulos associados à elas na iteração corrente. Note que uma desigualdade pode fazer parte, simultaneamente, dos dois grupos acima mencionados. E por

último, o terceiro grupo consiste das desigualdades remanescentes (desigualdades satisfeitas com multiplicadores nulos). Vale enfatizar que o cálculo dos subgradientes das desigualdades do terceiro grupo, pode acarretar um enorme esforço computacional.

Devemos verificar que de acordo com a classificação proposta acima, as desigualdades podem mudar de grupos de uma iteração para outra do MS. E mais, os multiplicadores que podem contribuir diretamente para os custos Lagrangeanos ($c^T - \lambda^T A$), em qualquer iteração do MS, são somente aqueles associados às desigualdades pertencentes ao primeiro e segundo grupos. Essas desigualdades serão denominadas, *desigualdades ativas.* Alternativamente, denominaremos as desigualdades do terceiro grupo como *desigualdades inativas.* É importante frisar ainda que, de acordo com (3.8), os multiplicadores associados às desigualdades pertencentes ao terceiro grupo, não mudarão seus valores (nulos) ao final da iteração corrente do MS.

É fácil verificar que as desigualdades inativas não contribuem diretamente para os custos Lagrangeanos (já que seus multiplicadores permanecem com valores nulos ao final da iteração corrente do MS). Por outro lado, elas têm um papel decisivo na determinação do valor de θ. Costumeiramente, para a aplicação descrita acima, o número de subgradientes estritamente *negativos* nas desigualdades do terceiro grupo, tende a ser enorme. Consequentemente, podemos ter como resultado, um valor para θ muito pequeno, levando à mudanças *"quase insignificantes"* no valor dos multiplicadores de iteração para iteração. Uma forma de evitar o problema é utilizar então, apenas as desigualdades nos grupos 1 e 2 para efetuar o cálculo de θ. Essa idéia foi sugerida em Lucena [50, 51] e leva a um esquema dinâmico onde o conjunto de desigualdades ativas pode mudar continuamente. Vale frisar que nesse esquema (assim como na OS convencional), uma desigualdade pode tornar-se ativa em uma certa iteração do MS, na iteração subsequente pode tonar-se inativa e numa iteração mais adiante, tornar-se ativa novamente. Porém, somente as desigualdades ativas são consideradas durante todo o processo da OS.

O esquema sugerido acima é, de certa forma, muito parecido com geração de planos-de-corte. Posteriormente, ele foi utilizado no Problema da Clique Máxima com Pesos nas Arestas em [44], no Problema de Roteamento de Veículos em [57], no Problema de Particionamento Retangular em [16] e no Problema de Ordenação Linear em [15].

3.2 Uma Relaxação Lagrangeana para o PAGCM

Em nosso esquema Lagrangeano para obtenção de limites inferiores para o PAGCM, trabalharemos com a formulação (2.1)-(2.4), (2.8), (2.6) e (2.7) e relaxaremos as restrições (2.3) e (2.8). Sejam $t_{ik} \in \mathbb{R}$, $\forall i \in I$, $\forall k \in \overline{I}$, os multiplicadores de Lagrange associados às restrições (2.3), $u_{ij} \geq 0, \forall (v_i, v_j) \in A$, os multiplicadores de Lagrange associados às restrições (2.8). Denotemos ainda por $\overline{e}_{ijk}$, os custos Lagrangenos das variáveis z_{ij}^k e por $\overline{d}_{ij}$, os custos Lagrangeanos das variáveis x_{ij}. Após dualizarmos as restrições, obtemos o seguinte problema:

$$Z_{lb}(t,u) = \min \sum_{(v_i,v_j)\in A} \overline{d}_{ij} x_{ij} + \sum_{i\in I} \sum_{j\in\overline{I},\, j\neq i} \sum_{k\in\overline{I}\ k\neq i} \overline{e}_{ijk} z_{ij}^k + \sum_{k\in\overline{I}} (t_{1k} + t_{kk}) \tag{3.9}$$

sujeito a:

$$\sum_{(v_j,v_i)\in A} x_{ji} = 1, \quad \forall i \in \overline{I} \tag{3.10}$$

$$z_{ij}^k \leq x_{ij}, \qquad \forall (v_i, v_j) \in A,\ \forall k \in \overline{I} \tag{3.11}$$

$$z_{ij}^k \geq 0, \qquad \forall (v_i, v_j) \in A, \forall k \in \overline{I} \tag{3.12}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (v_i, v_j) \in A \tag{3.13}$$

onde

$$\begin{aligned}
\overline{d}_{ij} &= \sum_{(v_i,v_j)\in A} [d_{ij} - (Q - q_i)u_{ij}] \\
\overline{e}_{ijk} &= C_{ijk} + u_{ij}q^k, \forall (v_i, v_j) \in A, k \in \overline{I}
\end{aligned}$$

$$\begin{aligned}
C_{ijk} &= -t_{ik} - t_{jk}, i \neq k, j = k \\
&= -t_{ik} + t_{jk}, i \neq k, j \neq 1, k \\
&= 0, \text{ caso contrário}
\end{aligned}$$

A restrição (3.11) impõe que uma variável z_{ij}^k, $(v_i, v_j) \in A$, $\forall k \in \overline{I}$, apenas poderá assumir um valor não-nulo quando x_{ij} estiver assumindo um valor não-nulo. Mais especificamente, dado que (3.9)-(3.13) é um problema de minimização, z_{ij}^k deverá sempre assumir o mesmo valor de x_{ij} quando $x_{ij} > 0$ e $\overline{e}_{ijk} < 0$, $\forall (v_i, v_j) \in A$, $\forall k \in \overline{I}$, $j \neq k$ (note também que $z_{ij}^k = x_{ij}$ sempre que $x_{ij} = 0$). Finalmente, podemos sempre impor $z_{ij}^j = x_{ij}$, $\forall (v_i, v_j) \in A$ em (2.4) e, independetemente do valor de $\overline{e}_{ijj}$, também em (3.11).

Podemos então reformular (3.9)-(3.13) unicamente em função das variáveis $\{x_{ij} : (v_i, v_j) \in A\}$, resolver essa reformulação e definir então, em função da solução obtida, valores ótimos para as variáveis $\{z_{ij}^k : (v_i, v_j) \in A, \forall k \in \overline{I}\}$. Para tanto, defina custos $f_{ij} = \overline{d}_{ij} + \sum_{k\in\overline{I}, k\neq j} \text{mínimo}(0, \overline{e}_{ijk}) + \overline{e}_{ijj}$, $\forall (v_i, v_j) \in A$, e considere o seguinte problema:

$$\overline{Z}_{lb}(t,u) = \min \sum_{(v_i,v_j)\in A} f_{ij}x_{ij} + \sum_{k\in\overline{I}} (t_{1k} + t_{kk}) \tag{3.14}$$

sujeito a:

$$\sum_{(v_j,v_i)\in A} x_{ji} = 1, \quad \forall i \in \overline{I} \tag{3.15}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (v_i, v_j) \in A \tag{3.16}$$

Para cada $(v_i, v_j) \in A$, os custos $\{f_{ij} : (v_i, v_j) \in A\}$ trazem embutidos, além dos custos $\{\overline{d}_{ij} : (v_i, v_j) \in A\}$ propriamente ditos, os custos das variáveis

$\{z_{ij}^k : (v_i, v_j) \in A, \forall k \in \overline{I}\}$, que assumiriam valores não nulos em (3.9)-(3.13) se $x_{ij} > 0$. Seja $\{\overline{x}_{ij} : (v_i, v_j) \in A\}$ uma solução ótima de (3.14)-(3.16) e compute valores para $\{z_{ij}^k : (v_i, v_j) \in A, \forall k \in \overline{I}\}$ da seguinte forma:

para todo $\overline{x}_{ij} > 0$, **faça**
$\overline{z}_{ij}^j = \overline{x}_{ij}$;
para todo $k \in \overline{I}$, **faça**
se $((j \neq k)$ e $(\overline{e}_{ijk} < 0))$, **então** $\overline{z}_{ij}^k = \overline{x}_{ij}$;
caso contrário, $\overline{z}_{ij}^k = 0$;

É importante ressaltar que incluir em (3.14), embutido no custo f_{ij} para $(v_i, v_j) \in A$, um custo $\overline{e}_{ijk} > 0$, onde $k \neq j$, seria claramente subótimo. Isso se aplica, pois na eventualidade de $x_{ij} = 1$ na solução ótima de (3.14)-(3.16), com $\overline{e}_{ijk} > 0$ embutido em f_{ij}, uma solução de menor custo (que não viola (3.11)) seria obtida tomando-se $z_{ij}^k = 0$.

Lema 3.1 *A solução* $\{x_{ij} = \overline{x}_{ij} : (v_i, v_j) \in A\}$, $\{z_{ij}^k = \overline{z}_{ij}^k : (v_i, v_j) \in A, \forall k \in \overline{I}\}$ *de (3.14)-(3.16) é ótima para (3.9)-(3.13).*

Prova: Substituindo-se os valores $\{x_{ij} = \overline{x}_{ij} : (v_i, v_j) \in A\}$ e $\{z_{ij}^k = \overline{z}_{ij}^k : (v_i, v_j) \in A, \forall k \in \overline{I}\}$ em (3.9)-(3.13) é fácil verificar que a solução é viável para o problema. É também fácil verificar que o valor correspondente de $Z_{lb}(u, v)$ é o mesmo de $\overline{Z}_{lb}(u, v)$. Assuma agora existir uma solução $\{x_{ij} = x_{ij}^{'} : (v_i, v_j) \in A\}$ e $\{z_{ij}^k = z_{ij}^{'k} : (v_i, v_j) \in A, \forall k \in \overline{I}\}$ de menor custo para (3.9)-(3.13). Isso, por sua vez, implicaria em dizer que $\{x_{ij} = x_{ij}^{'} : (v_i, v_j) \in A\}$, para custos $\{f_{ij} = \overline{d}_{ij} + \sum_{k \in \overline{I}, k \neq j} \text{mínimo}(0, \overline{e}_{ijk}) + \overline{e}_{ijj} : (v_i, v_j) \in A\}$, tem menor custo que $\{x_{ij} = \overline{x}_{ij} : (v_i, v_j) \in A\}$ para a reformulação (3.14)-(3.16), do problema Lagrangeano (3.9)-(3.13). Chega-se assim a uma contradição já que a segunda solução é assumida ser ótima para a reformulação. Fica então estabelecido que $\{x_{ij} = \overline{x}_{ij} : (v_i, v_j) \in A\}$, $\{z_{ij}^k = \overline{z}_{ij}^k : (v_i, v_j) \in A, \forall k \in \overline{I}\}$ é uma solução ótima para (3.9)-(3.13). ∎

Observe que qualquer solução viável para o PAGCM induz uma arborescência de $D = (V, A)$ com raiz em v_1. Dessa forma, é possível reforçar o limite inferior dado por (3.14)-(3.16) com a introdução de Desigualdades de Eliminação de Subrotas,

$$\sum_{(v_i,v_j)\in A(S)} x_{ij} \leq \mid S \mid -1 \qquad S \subseteq \overline{V},\ \text{e}\ \mid S \mid \geq 2, \tag{3.17}$$

onde $A(S)$ define o conjunto dos arcos com ambas as extremidades definidas por nós em S.

O problema definido por (3.14)-(3.17) é exatamente aquele de encontrar uma arborescência de custo mínimo no grafo $D = (V, A)$, com raiz em v_1, e custos dos arcos dados por $\{f_{ij} : (v_i, v_j) \in A\}$. Esse problema pode ser resolvido através da implementação feita por Fischetti e Toth [25] do algoritmo de solução de ordem $O(n^2)$ proposto para o problema por Edmonds [21]. Ainda em Fischetti e Toth [25] é possível obter custos reduzidos para as variáveis $\{x_{ij} : (v_i, v_j) \in A\}$ de (3.14)-(3.17) em ordem $O(n^2)$.

Dado que estamos interessados em obter o melhor limite inferior possível, queremos encontrar:

$$\max_{t\in\mathbb{R},\, u\geq 0} \{\overline{Z}_{lb}(t,u)\} = \max_{t\in\mathbb{R},\, u\geq 0} \{\min \sum_{(v_i,v_j)\in A} [\overline{d}_{ij} + \sum_{k\in\overline{I},\, k\neq j} \text{mínimo}(0,\ \overline{e}_{ijk}) + \overline{e}_{ijj}] + \sum_{k\in\overline{I}} (t_{1k} + t_{kk}) : (x,z) \in P\} \tag{3.18}$$

onde P é a região poliédrica definida pelas equações (3.10)-(3.13), (3.17).

3.3 Procedimento para Separação de DGES's Violadas

O algoritmo *Relax and Cut* proposto neste estudo é baseado no uso da formulação (2.1)-(2.4), (2.8), (2.6) e (2.7) do PAGCM, reforçada por DGES's (2.11). Em cada iteração do algoritmo um Subproblema Lagrangeano definido por (3.14)-(3.17) é resolvido e dentro do esquema proposto na seção

3.1 torna-se necessário classificar as desigualdades dualizadas em *ativas* e *inativas*. Dado que, à excessão das DGES's, o número de desigualdades dualizadas é pequeno, consideremos (2.3) e (2.8) como sendo sempre ativas.

No caso específico das DGES's deveremos resolver, à cada iteração do algoritmo, um problema auxiliar para identificar as desigualdades que violam a solução corrente do Subproblema Lagrangeano (ou concluir pela inexistência de tais desigualdades violadas). Dessa maneira, o problema a ser resolvido é semelhante ao Problema de Separação que deve ser resolvido em algoritmo *Branch and Cut* [58]. A grande diferença é que estaremos resolvendo aqui um Problema de Separação sobre uma estrutura inteira (o ponto do espaço a ser separado define um vetor de incidência de uma arborescência do grafo $D = (V, A)$). Dessa forma, em geral, o Problema de Separação em Algoritmos *Relax and Cut* tendem a ser resolvidos de maneira exata a uma complexidade mais baixa que o de seu congêneros em Algoritmos *Branch and Cut.*

Seja T_A uma arborescência do grafo $D = (V, A)$ com raiz em v_1. As desigualdades violadas por T_A são facilmente identificadas se eliminarmos de T_A todos arcos que saem de v_1 e considerarmos as componentes (subarborescência de T_A ou nós isolados) que resultem dessa operação. Assuma que o conjunto de nós em uma dessas componentes é dado por $S \subseteq \overline{V}$. É fácil verificar que sempre que a soma dos pesos dos nós em S, isto é, $d(S) = \sum_{v_i \in S} q_i$, exceder Q, isso implicará na violação em T_A da DGES definida pelo conjunto de nós S. Dado que o número de componentes considerados é linear em n, a complexidade de nosso Problema de Separação de DGES's é então, $O(n)$.

3.4 O Algoritmo *Relax and Cut* para o PAGCM

Em nossa implementação de um Algoritmo *Relax and Cut* para o PAGCM, utilizamos o esquema Lagrangeano da seção 3.2 submetendo somente as DGES's (2.11) à classificação dinâmica ((2.3) por ser uma restrição de igual-

dade será sempre considerada ativa e (2.8) por envolver um pequeno número de desigualdades será sempre classificada como ativa). Abrimos mão de utilzarmos as DLFA's (2.10) depois de verificarmos, em testes computacionais, que as DGES's dominavam as DLFA's.

O processo de incorporação das DGES's em nosso esquema Lagrangeano é realizado de forma dinâmica a medida em que elas são violadas, como descrito na seção 3.1. O procedimento de separação de DGES's violadas que utlizamos foi o da seção 3.3. Sejam $Z_{lb_{max}}$ e $Z_{ub_{min}}$, os valores correspondentes ao melhor limite inferior obtido (de maior valor) e ao melhor limite superior obtido (de menor valor), repectivamente, ao longo da aplicação do Algoritmo *Relax and Cut*. Denote ainda por L_A, a lista de DGES's ativas numa certa iteração. O algoritmo *Relax and Cut* para o PAGCM, pode então, ser assim descrito:

Passo 1. Fixar os valores iniciais dos multiplicadores de Lagrange, $Z_{lb_{max}}$, $Z_{ub_{min}}$ e L_A

$$\begin{aligned} t_{ik} &= 0, \qquad \forall i \in I,\ \forall k \in \overline{I} \\ u_{ij} &= 0, \qquad \forall (v_i, v_j) \in A, \\ Z_{lb_{max}} &= -\infty, \\ Z_{ub_{min}} &= +\infty, \\ L_A &= \emptyset \end{aligned}$$

Passo 2. Resolver o problema dual Lagrangeano com o conjunto de multiplicadores corrente. Seja Z_{lb} (limite inferior obtido na iteração), $\overline{x}_{ij}$, $\overline{z}^k_{ij}$, a solução obtida.

Passo 3. Procurar violações para DGES's em $\overline{x}_{ij}$. Para cada violação, definida por $S_l \subseteq \overline{V}$, encontrada, verificar se a desigualdade está contida em L_A. Caso não esteja, incluí-la em L_A. Associar à desigualdade definida por $S_l \subseteq \overline{V}$, um multiplicador de Lagrange $\lambda_l \geq 0$, inicialmente com valor 0.

Passo 4. Obter um novo Z_{ub} (valor correspondente a uma solução viável). Atualizar adequadamente $Z_{lb_{max}}$ e $Z_{ub_{min}}$.

Passo 5. Se $(Z_{ub_{min}} - Z_{lb_{max}} < 1)$, então Z_{ub} é uma solução ótima para o PAGCM. Senão, ir para o passo 6.

Passo 6. Calcular os Subgradientes

$$H_{1k} = 1 - \sum_{\forall(v_1,v_j)\in A} \overline{z}^k_{1j}, \quad \forall k \in \overline{I}$$

$$H_{kk} = 1 - \sum_{\forall(v_j,v_k)\in A} \overline{z}^k_{jk}, \quad \forall k \in \overline{I}$$

$$H_{ik} = \sum_{(v_h,v_i)\in A} \overline{z}^k_{hi} - \sum_{(v_i,v_j)\in A} \overline{z}^k_{ij}, \quad \forall k \in \overline{I}, \forall i \in \overline{V} - \{k\}$$

$$G_{ij} = (Q - q_i)\overline{x}_{ij} - \sum_{\forall k\in\overline{I}} \overline{z}^k_{ij} q_k, \quad \forall (v_i, v_j) \in A,$$

$$P_l = \mid S_l \mid - \lceil d(S_l)/Q \rceil - \sum_{(v_i,v_j)\in A(S_l)} \overline{x}_{ij},$$

$$S_l \subseteq \overline{V} \text{ e } \mid S_l \mid \geq 2, \quad \forall S_l \in L_A$$

onde $A(S_l)$ define o conjunto dos arcos com ambas as extremidades definidas por nós em S_l e $d(S_l) = \sum_{v_i\in S_l} q_i$.

Passo 7. Definir um tamanho de passo θ por

$$\theta = \frac{\alpha[(1+\beta)Z_{ub_{min}} - Z_{lb}]}{\sum_{i\in I}\sum_{k\in\overline{I}}(H_{ik})^2 + \sum_{(v_i,v_j)\in A}(G_{ik})^2 + \sum_{S_l\in L_A}(P_l)^2}$$

onde $0 < \alpha \leq 2$, $\beta \leq 0.03$ e atualizar os multiplicadores de Lagrange

$$t_{ik} = t_{ik} + \theta H_{ik}, \qquad \forall i \in I, \forall k \in \overline{I}$$

$$u_{ij} = \max\{0,\ u_{ij} - \theta G_{ij}\}, \qquad \forall (v_i, v_j) \in A$$

$$\lambda_l = \max\{0,\ \lambda_l - \theta P_l\}, \qquad \forall S_l \in L_A$$

Passo 8. Retirar possíveis DGES's inativas.

para todo $S_l \in L_A$ **faça**

se $\lambda_l = 0$

então $L_A = L_A \setminus \{S_l\}$

Passo 9. Retornar ao passo 2 e resolver o problema dual Lagrangeano com o novo conjunto de multiplicadores (obtidos no passo 7), caso o número de iterações desejadas não tiver sido atingido.

Algumas observações são necessárias a respeito do algoritmo. A primeira delas é que os critérios de parada que adotamos foram o número máximo de iterações permitido (variou para determinados tamanhos de instâncias) e aquele indicado no passo 5. Esse último critério, só é possível porque todas as instâncias tratadas têm custos inteiros. Uma outra observação é que é importante utilizar uma estrutura de dados eficiente para manipular e manter L_A. Uma questão crucial é verificar se um DGES violada numa certa iteração, já não está presente em L_A. A utilização adequada de uma tabela de *hashing* [66] pode agilizar esse processo. O fator β é utilizado como sugerido em Beasley [12]. Uma outra questão importante é a maneira como reduzimos o valor de α ao longo das iterações (também variou quando instâncias de tamanhos diferentes foram tratadas). E, por fim, a obtenção de um novo limite superior (passo 4) em cada iteração é realizada através da execução de uma Heurística Lagrageana, objeto de estudo do capítulo 7. Detalharemos melhor as variações de algumas desses parâmetros (número de iterações permitido, redução do valor de α), quando formos tratar dos resultados computacionais que obtivemos.

3.5 Redução do Problema

Nesta seção, apresentaremos algumas formas de reduzir o "o tamanho de entrada" de uma instância do PAGCM através da fixação de variáveis. Os conjuntos de variáveis que são considerados nas duas formas de redução são distintos. Na primeira forma de redução trabalhamos com arestas, e na segunda, com arcos.

A primeira redução é realizada como um pré-processamento, antes mesmo da execução de qualquer algoritmo e é baseada em algumas características

do grafo original não-direcionado. A segunda é uma forma dinâmica (ao longo das iterações do Algoritmo *Relax and Cut* para o PAGCM) de reduzir o problema. Aqui, o grafo considerado é direcionado. Para isso, utilizamos as informações dos custos reduzidos das variáveis $\{x_{ij} : (v_i, v_j) \in A\}$.

3.5.1 Redução Inicial [55]

Proposição 3.1 *Podemos fixar uma variável* x_{ij} *em* 0, *se*

$$c_{ij} \geq \text{máximo}\{c_{1j},\ c_{1i}\} \tag{3.19}$$

Prova. Assuma que $[v_i, v_j]$ está na solução ótima. Inicialmente, verificamos que as arestas $[v_1, v_i]$ e $[v_1, v_j]$ não podem simultaneamente fazer parte dessa solução pois isso acarretaria um ciclo. E se por exemplo, $[v_1, v_j]$ não estiver na solução ótima, ao substituirmos a aresta $[v_i, v_j]$ pela aresta $[v_1, v_j]$, não alteramos a viabilidade da solução, pois apenas dividimos a subárvore original (à qual a aresta $[v_i, v_j]$ pertence) em duas subárvores. Olhando para (3.19), verificamos que o custo total para a nova solução é necessariamente igual ou menor ao da subárvore original. ∎

Vale lembrar mais uma vez, que o grafo considerado nessa redução, é o não-direcionado, ou seja, o conjunto de variáveis utilizado é $\{e = [v_i, v_j],\ i, j \in I, i < j\}$. Essa redução é aplicada no passo 1 do nosso Algoritmo *Relax and Cut* para o PAGCM.

3.5.2 Redução baseada na Análise dos Custos Reduzidos

Para economizar tempo de CPU, a fixação de variáveis através da análise dos custos reduzidos é realizada somente quando ocorre um aumento global do melhor limite inferior até então conhecido. Os parâmetros para a fixação são o

limite inferior da iteração em que a melhora ocorreu e o melhor limite superior obtido até então. Os custos reduzidos, como mencionado anteriormente, são fornecidos pelo algoritmo proposto em [25], após o cálculo da arborescência de custo mínimo.

Seja T_A uma arborescência de custo mínimo e $\zeta(i,j)$ o custo reduzido associado a um arco $(v_i, v_j) \in A$.

Proposição 3.2 *Podemos fixar a variável x_{ij} em 0, se*

$$Z_{lb} + \zeta(i,j) \geq Z_{ub} \tag{3.20}$$

Prova. Considere o problema de encontrar uma arborescência viável para o PAGCM que contenha o arco (v_i, v_j) e tenha o menor custo possível. Claramente, tal problema é, pelo menos tão difícil quanto nosso problema original. No entanto, $Z_{lb} + \zeta(i,j)$, pela definição de custo reduzido em Programação Linear, é um limite inferior válido para o novo problema. Sendo assim, se $Z_{lb} + \zeta(i,j) > Z_{ub}$, fica estabelcido que qualquer solução viável para o PAGCM, que contenha (v_i, v_j), tem um custo que ultrapassa o da solução viável Z_{ub}. O arco (v_i, v_j) *não* pode então fazer parte de uma solução ótima para o PAGCM. ∎

Além da fixação acima, verificamos que também poderíamos fixar em zero algumas variáveis de fluxo, $\{z_{ij}^k : (v_i, v_j) \in A, \forall k \in \overline{I}\}$, através de um teste semelhante. Esse teste se aplica apenas a variáveis z_{ij}^k com um custo Lagrangeano $\overline{e}_{ijk} \geq 0$ (ou seja, o custo f_{ij} não possui contribuição de $\overline{e}_{ijk}$).

Proposição 3.3 *Podemos fixar uma variável z_{ij}^k em 0 se*

$$Z_{lb} + \zeta(i,j) + \overline{e}_{ijk} \geq Z_{ub} \tag{3.21}$$

Prova. A prova é análoga a anterior, observando apenas que se uma solução é ótima para um Problema de Programação Linear (arborescência de custo

mínimo no nosso caso), se aumentarmos em $\epsilon > 0$, o custo f_{ij} de uma variável não-básica x_{ij} qualquer, a solução ótima para o problema e o custo reduzido de x_{ij} ficaria acrescido de ϵ. Se x_{ij} for básica, e dessa forma $\zeta_{ij} = 0$, $Z_{lb} + \epsilon$ seria um limite inferior para o custo de uma solução ótima do problema em que $x_{ij} = 1$ é imposto, completando assim a prova. ∎

Capítulo 4

A Heurística Esau-Williams para o PAGCM

Neste capítulo, definiremos uma notação que será utilizada ao longo desse e dos próximos capítulos. Além disso, apresentaremos a heurística Esau-Williams [23], uma das heurísticas mais conhecidas e utilizadas da literatura.

4.1 Notação

A notação que será definida agora, é bastante similar aquela proposta em Ahuja *et al.* [8]. Nesse contexto, algumas notações da teoria dos grafos também serão utilizadas, tais como: subárvores, caminhos, ciclos, etc. Além disso, algumas definições de termos conhecidos serão omitidas. Maiores detalhes sobre essas definições poderão ser encontrados em [4].

Seja T uma árvore geradora capacitada viável para o PAGCM com raiz em v_1. Definindo-se uma orientação para esta árvore a partir de v_1, todo nó v_j tal que $[v_1, v_j] \in T$ é denominado um filho de v_1. De maneira análoga, v_1 será o pai de v_j. Essa convenção de parentesco (pai-filho) pode ser generalizada da seguinte forma. Para toda aresta $[v_i, v_j] \in T$, v_i será o pai de v_j se estiver mais próximo de v_1 (em termos do número de arestas em T) e v_j será o pai de v_i em caso contrário. Por exemplo, na figura 4.1, v_2 é o pai dos nós v_7 e v_{10} e v_{13} é filho de v_7.

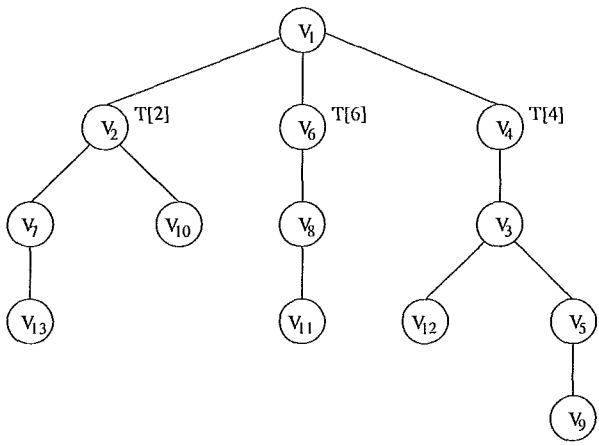

Figura 4.1: Um exemplo de uma Árvore Geradora Capacidada de Custo Mínimo. Nesse exemplo, todas as demandas são unitárias e $Q = 5$.

Para cada nó v_i em T, denominaremos por T_i, a subárvore de T com raiz em v_i. Denominaremos ainda por S_i, os descendentes de v_i, ou seja, os filhos de v_i, os filhos de seus filhos e assim por diante. Como exemplo, considere a figura 4.1, onde T_3 denota a subárvore com raiz v_3 e $S_3 = \{v_3, v_5, v_9, v_{12}\}$.

Além disso, denominaremos de subárvore de nível 1, uma subárvore onde a raiz é um filho de v_1. Para qualquer nó v_i em T, denotaremos por $T[i]$ a subárvore de nível 1 à qual v_i pertence. Denominaremos ainda por $S[i]$, o conjunto de nós contidos na subávore de nível 1 $T[i]$ à qual v_i pertence. Por exemplo, na figura 4.1, para cada nó $v_i = v_7, v_{10}, v_{13}$, $T[i] = T[2]$ e $S[i] = \{v_2, v_7, v_{10}, v_{13}\}$. Pode acontecer, no entanto, que venhamos a chamar uma subárvore de nível 1 apenas como subárvore. Porém, ficará óbvio para o contexto onde isso acontecer que a referência é para uma subárvore de nível 1.

E, finalmente, para um dado subconjunto de nós $S \subseteq \overline{V}$, $d(S) = \sum_{v_i \in S} q_i$. Se $d(S) \leq Q$, então S é viável. Denotaremos ainda por $c(S)$, o custo de uma árvore geradora de custo mínimo para o conjunto de nós $S \cup \{v_1\}$.

4.2 A Heurística Esau-Williams

Como foi mencionado anteriormente, muitas heurísticas diferentes foram propostas para o PAGCM [9, 31]. Entre elas, podemos citar aquelas de Esau-Williams [23], Kershenbaum [45], Kershenbaum e Chou [48], Kershenbaum, Boorstyn e Oppenheim [47], Gavish e Altinkemer [32] e Gouveia e Paixão [41].

A heurística de Esau-Williams [23], denotada aqui Esau-Williams (EW), é uma das mais citadas na literatura e é frequentemente utilizada como base de computação em testes computacionais. Ela é simples de implementar e utiliza o conceito de *savings*, como no algoritmo proposto em Clarke e Wright [18] para o Problema de Roteamento de Veículos. Em Amberg *et al.* [9], EW é apontada como a heurística de melhor desempenho computacional (incluindo qualidade da solução) na média, quando comparada a outros procedimentos que gastam tempos computacionais similares a ela. Sua complexidade é da ordem $O(n^2 log\, n)$ [31]. Devido a tudo isso, optamos por utilizar EW como heurística básica em nosso esquema.

Podemos descrever a heurística EW da seguinte maneira. Dado um nó v_j, denote por g_j, a aresta $[v_1, v_q]$ e b_j seu custo, onde $T[j] = T[q]$ (v_q é um filho de v_1 e pertence a mesma subárvore de v_j) e compute a matriz assimétrica $S = s_{ij}$, onde $s_{ij} = c_{ij} - b_j$. Iniciando com uma topologia onde cada nó $v_p \in \overline{V}$ está conectado a v_1 por uma aresta $[v_1, v_p]$, a heurística EW repete, iterativamente, os seguintes passos:

Passo 1. Identificar dois nós v_i e v_j, onde $T[i] \neq T[j]$, que possuem o menor *saving* s_{ij} e tal que o somatório das demandas de todos os nós que estão nas subárvores $T[i]$ e $T[j]$ não exceda Q, isto é, $\sum_{v_w \in S[i] \cup S[j]} q_w \leq Q$. Se não existir mais nenhum *saving* s_{ij} negativo e viável, Pare.

Passo 2. Remover g_j. Introduzir a aresta $[v_i, v_j]$ para expandir $T[i]$. Atualizar $s_{ku} = s_{ku} + b_j - b_i$ ($v_k \in \overline{V} \setminus \{S[i] \cup S[j] \cup \{v_1\}\}, v_u \in S[j]$). Retornar ao Passo 1.

Para uma melhor compreensão, considere o seguinte exemplo. Sejam $V = \{v_1, v_2, v_3, v_4, v_5\}$, v_1 a raiz e $Q = 3$. Sejam ainda, c_{ij} (veja figura 4.2) a tabela (matriz) de custos das arestas e s_{ij} a tabela (matriz) de *savings*.

-	1	2	3	4	5
1	-	87	367	240	251
2	87	-	87	20	128
3	367	87	-	156	3
4	240	20	156	-	61
5	251	128	3	61	-

Tabela 4.1: Tabela de custos c_{ij}

As tabelas 4.2-4.4 mostram iteração por iteração, o *saving* escolhido (passo 1, $\boxed{s^*_{ij}}$) e os *savings* que foram atualizados (passo 2, valores em negrito).

Na iteração 1 do exemplo acima, a tabela 4.2 é utilizada na busca do melhor *saving* s^*_{ij} viável, no caso, $s^*_{ij} = s_{53} = -364$. Seguimos então para o passo 2 onde são atualizados os *savings* $s_{23} = -164$ e $s_{43} = -95$, levando a tabela 4.3. Na iteração 2 (realizada a partir dos *savings* s_{ij} atualizados), o passo 1 é realizado novamente e $s^*_{ij} = s_{24} = -220$. Isso leva novamente a alterações em s_{ij}, $s_{34} = 69$ e $s_{54} = -26$. Repare que cada vez que um s^*_{ij} é selecionado, ele não é mais considerado nas iterações seguintes. Para cada s^*_{ij} escolhido, associamos o símbolo $*$ tanto para s^*_{ij} quanto para s^*_{ji}. Por fim, na iteração 3 , repetimos o passo 1 (a partir dos s_{ij} atualizados) e verificamos que não existe mais nenhum $s_{ij} < 0$ e viável, o que nos leva ao final da execução da EW.

A solução do exemplo acima é $\{(v_1, v_2), (v_1, v_5), (v_2, v_4), (v_5, v_3)\}$ e seu custo é 361. É fácil ver que essa solução é ótima para o exemplo. A figura 4.3 mostra as muadanças na topologia da AGC que está sendo formada iteração por iteração. Na iteração 1, a topologia inicial é a presente na figura 4.3(1). Na iteração 2, a topologia corrente é 4.3(2) e, por fim, na iteração 3, a topologia corrente é 4.3(2) (topologia final obtida pela EW).

-	1	2	3	4	5
1	-	87	367	240	251
2	-	-	-280	-220	-123
3	-	0	-	-84	-248
4	-	-67	-211	-	-190
5	-	41	-364	-179	-

Tabela 4.2: Tabela de *savings* s_{ij} que serve de entrada (passo 1) para a iteração 1 da EW

-	1	2	3	4	5
1	-	87	367	240	251
2	-	-	**-164**	-220	-123
3	-	0	-	-84	*
4	-	-67	**-95**	-	-190
5	-	41	*	-179	-

Tabela 4.3: Tabela de *savings* s_{ij} do final (passo 2) da iteração 1 e que serve de entrada para a iteração 2 da EW

-	1	2	3	4	5
1	-	87	367	240	251
2	-	-	-164	*	-123
3	-	0	-	**69**	*
4	-	*	-95	-	-190
5	-	41	*	**-26**	-

Tabela 4.4: Tabela de *savings* s_{ij} do final da iteração 2 e que serve de entrada para a iteração 3 da EW

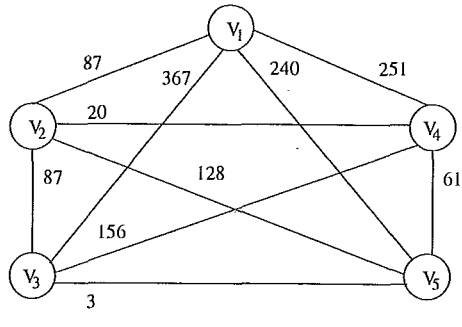

Figura 4.2: Grafo de entrada para a EW

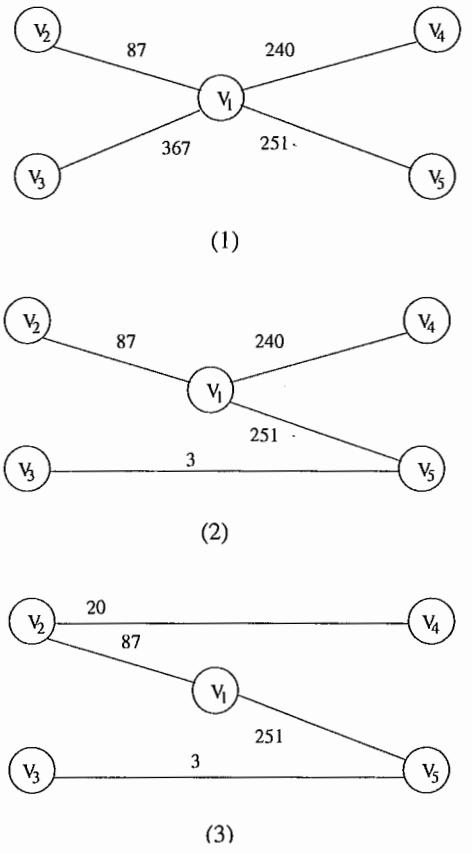

Figura 4.3: As modificações ocorridas a cada iteração da EW

Capítulo 5

Busca Local para o PAGCM

Neste capítulo, faremos uma revisão das principais vizinhanças de busca propostas na literatura para o PAGCM.

Em geral, uma busca local consiste na exploração e obtenção de novas soluções viáveis (preferivelmente melhores) a partir de uma solução viável inicial dada (veja Aarts e Lenstra [2] e Ahuja *et al.* [6]). O sucesso de uma busca local depende fundamentalmente da vizinhança de busca a ser explorada, ou seja, da estrutura que permite a obtenção dessas novas soluções. Mais adiante, quando apresentarmos as nossas Heurísticas Lagrangeanas, mostraremos, detalhamente, quais as estratégias de busca local que utilizamos. Muitas das vizinhanças de busca que serão apresentadas foram utilizadas na composição de nossas Heurísticas Lagrangenas.

Um algoritmo de busca local para o PAGCM pode ser assim descrito. Uma solução viável T é obtida. Soluções vizinhas de T são definidas como sendo soluções que diferem de T de acordo com um padrão claramente definido (por exemplo, soluções que diferem de T em, no máximo, 3 variáveis). A seguir, um procedimento é utilizado para identificar a solução vizinha de T mais atraente. Uma substituição de soluções deve ser efetuada se o custo da solução encontrada encontrada for inferior ao custo de T. Esse processo deve continuar enquanto um critério de parada adotado não for atingido.

Como mencionado anteriormente, o desempenho de um algoritmo de busca local depende fundamentalmente da vizinhaça de busca a ser adotada. Faremos agora, uma revisão de algumas das vizinhanças de busca para o PAGCM que foram propostas na literatura. Nas próximas seções, apresentaremos as duas vizinhanças de busca propostas por Amberg *et al.* [9] e Sharaiha *et al.* [64] e as vizinhanças propostas por Ahuja *et al.* [8, 5]. Além disso, o conceito de Grafo de Melhoria também será apresentado. Por simplificação, assumiremos nas próximas seções que todos os nós possuem demandas idênticas. O caso onde as demandas são heterogêneas pode ser tratado de maneira similar.

5.1 Vizinhança de Busca Baseada em 2-Trocas de Nós

A vizinhança de busca proposta por Amberg *et al.* [9] considera a realização de dois tipos de movimentos:

Transferência de um Nó: consiste em remover um determinado nó da subárvore de nível 1 à qual ele pertence e levá-lo para outra subárvore de nível 1.

Troca de Nós: consiste na troca de nós (dois-a-dois) pertencentes à subárvores de nível 1 distintas.

Essa vizinhança é denominada *vizinhança baseada em 2-trocas de nós* devido ao fato de que apenas nós são movimentados e no máximo, duas subárvores de nível 1 distintas são afetadas pelos movimentos propostos.

Algumas observações importantes a respeito dessa vizinhança de busca devem ser feitas. Os nós que são selecionados para qualquer um dos movi-

mentos, não são, necessariamente, folhas (nós que não possuem filhos). Após qualquer transferência ou troca de nós, o ajuste da nova solução é feito através do cálculo de uma árvore geradora de custo mínimo a partir dos nós pertencentes à cada uma das subárvores de nível 1 (incluindo v_1) envolvidas no movimento. E, obviamente, somente as movimentações viáveis, ou seja, as que não violam a restrição de capacidade, são permitidas. A figura 5.1 ilustra os dois tipos de movimentos propostos por Amberg *et al.*. A figura 5.1(a) demonstra a transferência (viável) do nó v_{11} da subárvore $T[2]$ para a subárvore $T[3]$. A figura 5.1(b) demonstra a troca (viável) do nó v_6 da subárvore $T[2]$ pelo nó v_7 da subárvore $T[3]$.

É fácil verificar que existem no máximo $O(nK)$ transferências de nós e $O(n^2K)$ troca de nós, onde K é o número de subárvores de nível 1 existentes na solução T (isto é, solução viável que foi passada como entrada para a exploração da vizinhança). O custo da transferência de um determinado nó assim como o custo de uma troca de nós é calculado através da verificação do impacto que cada um desses movimentos tem na função objetivo. O custo da transferência de um nó v_i da subárvore $T[i]$ para a subárvore $T[j]$, $T[i] \neq T[j]$, é $c(S[i] \setminus \{v_i\}) - c(S[i]) + c(\{v_i\} \cup S[j]) - c(S[j])$ e pode ser determinado pela resolução (reotimização) de dois problemas da árvore geradora de custo mínimo. O custo de uma de troca de nós que envolve trocar um nó v_i da subárvore $T[i]$ por um nó v_j da subárvore $T[j]$, $T[i] \neq T[j]$, é $c(\{v_j\} \cup S[i] \setminus \{v_i\}) - c(S[i]) + c(\{v_i\} \cup S[j] \setminus \{v_j\}) - c(S[j])$ e assim como na transferência, pode ser determinado através da resolução de dois problemas da árvore geradora de custo mínimo. Cada árvore geradora de custo mínimo pode ser resolvida em $O(Q^2)$, quando todas as demandas são idênticas, pelo algoritmo de Prim [61]. Consequentemente, esse procedimento leva $O(n^2Q^2)$ para determinar a melhor (de maior impacto na função objetivo) transferência ou troca de nós a ser efetuada.

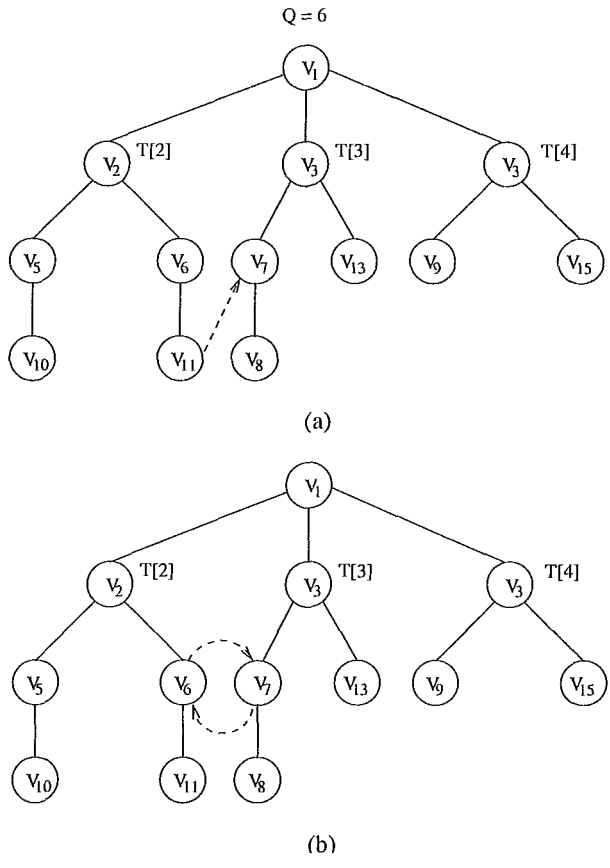

Figura 5.1: Ilustração da Vizinhança de Busca proposta por Amberg *et al.* (a) Transferência de um nó (b) Troca de nós

5.2 Vizinhança de Busca Baseada em 2-Trocas de Subárvores

A vizinhança de busca proposta por Saraiha *et al.* [64] consiste na realização de operações denominadas *cut* e *paste*. Uma operação de *cut* e *paste* consiste em desconectar (*cut*) uma subárvore contida numa subárvore de nível 1 e conectá-la (*paste*) a raiz ou a alguma outra subárvore. A figura 5.2 ilustra essa operação. Na figura 5.2(a), a subárvore T_6, parte da subárve $T[2]$ com raiz em v_6, é desconectada de $T[2]$ e conectada a raiz pela aresta de menor custo entre T_6 e a raiz. Já na figura 5.2(b) a subárvore T_6 é desconectada de $T[2]$ e conectada a subárvore $T[3]$ pela aresta de menor custo entre T_6 e $T[3]$. Se uma operação de *cut* e *paste* é efetuada e uma aresta $[v_i, v_j]$ é desconectada levando a inclusão de uma nova aresta $[v_p, v_q]$, a redução do custo da solução que resulta dessa operação é $c_{pq} - c_{ij}$.

Essa vizinhança de busca é denominada *vizinhança de busca baseada em 2-trocas de subárvores* uma vez que os movimentos envolvidos são realizados com subárvores e no máximo duas subárvores de nível 1 são afetadas a cada realização de uma operação *cut* e *paste*.

Existem $O(n^2)$ possibilidades de realizar operações *cut* e *paste*. A identificação de cada uma dessas operações leva $O(Q^2)$. Consequentemente, para determinar a melhor operação *cut* e *paste* a ser realizada, o tempo gasto é da ordem $O(n^2Q^2)$.

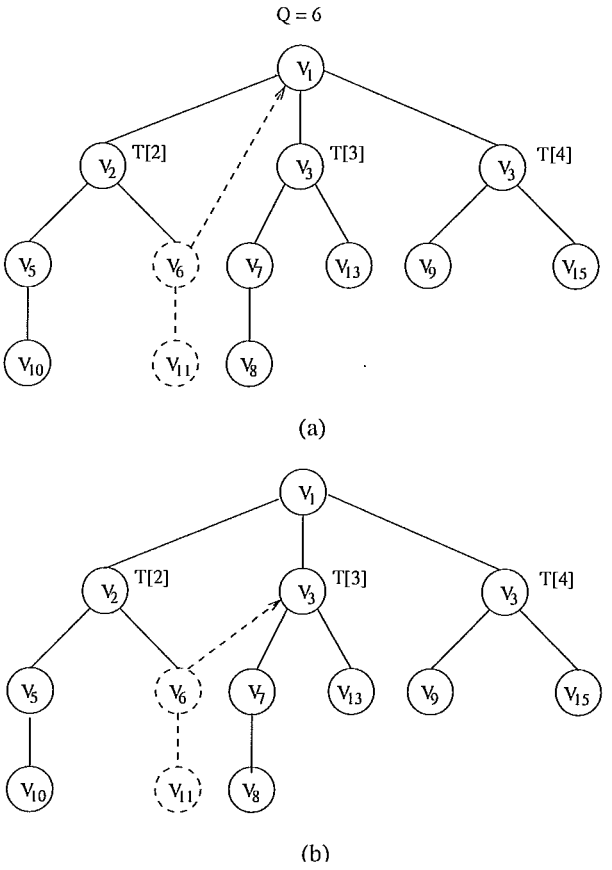

Figura 5.2: Ilustração da Vizinhança de Busca proposta por Saraiha *et al.* (a) Uma operação de *cut* e *paste* onde a subárvore T_6 é conectada ao nó central (b) Uma operação de *cut* e *paste* onde a subárvore T_6 é conectada a outra subárvore (T[3])

5.3 Vizinhança de Busca Baseada na Troca-Múltipla de Nós

Essa vizinhança de busca foi proposta por Ahuja *et al.* [8]. Ela pode ser vista como uma generalização da vizinhança de busca proposta por Amberg *et al.* [9]. Porém, nessa nova vizinhança, é possível, agora, a movimentação de vários nós (todos pertencentes à subárvores de nível 1 distintas). Ou seja, é possível que mais de duas subárvores de nível 1 sejam afetadas a cada movimento realizado, ao contrário das vizinhanças anteriormente citadas. Essa nova vizinhança é denominada *vizinhança de busca baseada na troca-múltipla de nós*. Ela pode ser obtida através da realização de *trocas-cíclicas* e *trocas-caminho*. A seguir, detalharemos como se dá cada uma dessas trocas, quando estamos movimentando nós.

5.3.1 Troca-Cíclica (TC) para Movimentar Nós (TCN)

Uma TCN pode ser definida como uma sequência de nós $v_2 - v_3 - v_4 - ... - v_r - v_2$, onde os nós $v_2 - v_3 - v_4 - ... - v_r$ pertencem à subárvores de nível 1 distintas, isto é, $T[p] \neq T[q]$, para $p \neq q$. A TCN $v_2 - v_3 - v_4 - ... - v_r - v_2$ representa as seguintes mudanças: O nó v_2 é movido da subárvore $T[2]$ para a subárvore $T[3]$, o nó v_3 é movido da subárvore $T[3]$ para a subárvore $T[4]$, e assim por diante, e finalmente, o nó v_r é movido da subárvore $T[r]$ para a subárvore $T[2]$. Uma TCN é *viável* se tal troca satisfaz a restrição de capacidade. Mais especificamente, a TCN $v_2 - v_3 - v_4 - ... - v_r - v_2$ é uma troca viável se e somente se o conjunto de nós $\{v_{p-1}\} \cup S[p] \setminus \{v_p\}$ é viável para $p = 2, 3, ..., r$, onde $v_1 = v_r$. Somente as TCN's viáveis serão consideradas. Seja T' uma árvore geradora capacitada viável obtida por uma TCN a partir de uma solução também viável T. O custo dessa TCN pode ser definido como $c(T') - c(T)$, ou seja:

$$c(T') - c(T) = \sum_{p=2}^{r} [\, c(\{v_{p-1}\} \cup S[p] \setminus \{v_p\}) - c(S[p]) \,] \tag{5.1}$$

onde $v_1 = v_r$.

Ao longo de uma TCN, vários problemas da árvore geradora de custo mínimo são resolvidos a partir de cada um dos novos subconjuntos de nós que são formados. Cada um desses subconjuntos é formado a partir dos nós de cada uma das subárvores envolvidas na troca (retirado o nó que irá para outra subárvore), acrescido do nó que foi movido de outra subárvore. O custo de uma TCN, na realidade, é diferença entre o somatório dos custos das novas subávores obtidas após a troca e somatório dos custos das subárvores anteriores. É fácil ver que somente as trocas-cíclicas onde $c(T') < c(T)$ são interessantes. A figura 5.3 ilustra uma TCN.

É importante ressaltar que uma TCN pode acarretar o aumento do número de subárvores. Quando está sendo realizada a computação $c(\{v_{p-1}\} \cup S[p] \setminus \{v_p\})$ a partir do conjunto de nós $\{v_1, v_{p-1}\} \cup S[p] \setminus \{v_p\})$, é possível que duas ou mais arestas partindo de v_1 apareçam na nova solução; nesse caso, uma simples subárvore pode transformar-se em duas ou mais subárvores. Porém, é fácil verificar que, o contrário, ou seja, a diminuição do número de subárvores após uma TCN, não é possível.

5.3.2 Troca-Caminho (TCA) para Movimentar Nós (TCAN)

Uma TCAN pode ser definida como uma sequência de nós $v_2 - v_3 - v_4 - ... - v_r$, onde os nós $v_2 - v_3 - v_4 - ... - v_r$ pertencem à subárvores de nível 1 distintas, isto é, $T[p] \neq T[q]$, para $v_p \neq v_q$. A TCAN $v_2 - v_3 - v_4 - ... - v_r$ representa as seguintes mudanças: O nó v_2 é movido da subárvore $T[2]$ para a subárvore $T[3]$, o nó v_3 é movido da subárvore $T[3]$ para a subárvore $T[4]$, e assim por diante, e finalmente, o nó v_{r-1} é movido da subárvore $T[r-1]$ para a subárvore $T[r]$. Essa troca difere de uma TCN apenas no último movimento onde o nó

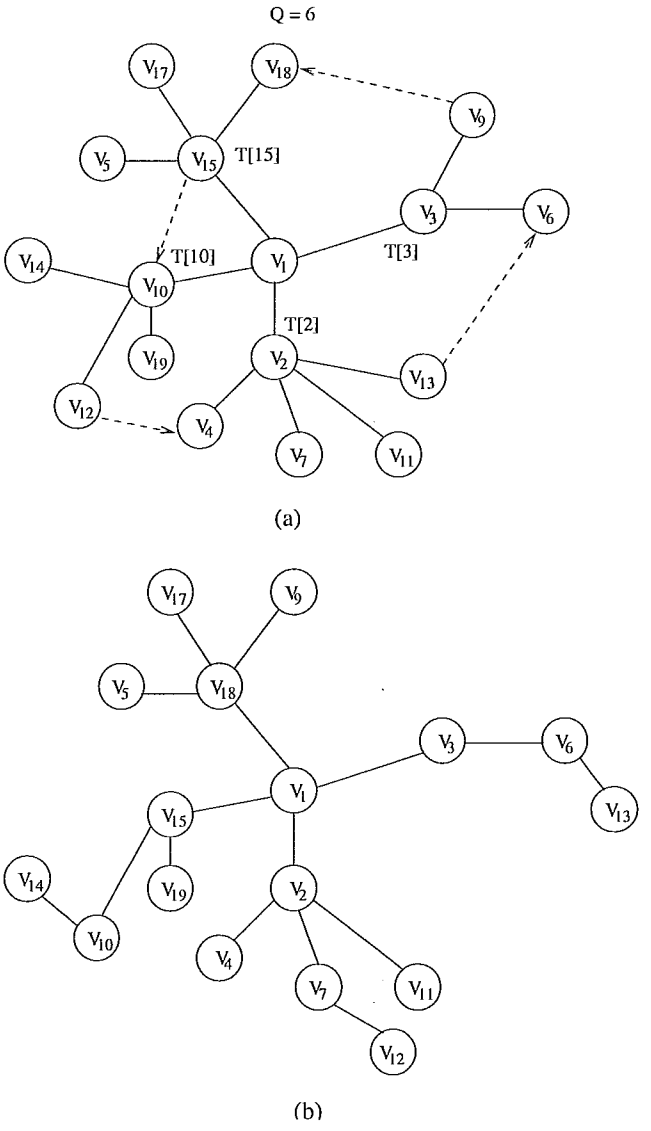

Figura 5.3: Ilustração de uma TCN (a) Uma AGC antes da TCN (b) A AGC depois da TCN

v_r não vai para a subárvore $T[2]$. Dessa forma, a subárvore $T[2]$ perde um nó e a subárvore $T[r]$ ganha um nó. Uma TCAN é *viável* se ela satisfaz a restrição de capacidade. Mais especificamente, a TCAN $v_2 - v_3 - v_4 - ... - v_r$ é uma troca viável se e somente se o conjunto de nós $\{v_{p-1}\} \cup S[p] \setminus \{v_p\}$ é viável para $p = 3, ..., r$. Somente as TCAN's viáveis serão consideradas. Seja T' uma árvore geradora capacitada viável obtida por uma TCAN a partir de uma solução também viável T. O custo dessa TCAN pode ser definido como $c(T') - c(T)$, ou seja:

$$c(T') - c(T) = c(S[2] \setminus \{v_2\}) + \sum_{p=3}^{r-1}(c(\{v_{p-1}\} \cup S[p] \setminus \{v_p\})) + c(\{v_{r-1}\} \cup S[r]) - \sum_{p=2}^{r}(c(S[p]) \quad (5.2)$$

É fácil ver que uma TCAN é interessante somente quando $c(T') < c(T)$. A figura 5.4 ilustra uma TCAN. Assim como ocorre em uma TCN, uma TCAN também pode aumentar o número de subárvores em T. Porém, ao contrário de uma TCN, uma TCAN pode, também, reduzir o número de subárvores em T. Por exemplo, na TCAN $v_2 - v_3 - v_4 - ... - v_r$, a subárvore $T[2]$ perde um nó. Se a subárvore $T[2]$ consistir apenas de um único nó, o número de subárvores na solução T' após essa troca, irá diminuir.

Em relação às trocas acima (TCN e TCAN), podemos observar o seguinte. Se T tem K subárvores para algum K fixo, então existem $O(n^K)$ possibilidades de TCN's e TCAN's. Note que $O(n^K)$ é substancialmente maior do que $O(n^2)$ das vizinhanças propostas por Amberg *et al.* e Saiaha *et al.*. Note ainda que uma TCAN pode ser transformada em uma TCN. Elas diferem apenas no último movimento da troca, veja [8].

5.4 Vizinhança de Busca Baseada na Troca-Múltipla de Subárvores

A vizinhança de busca que será apresentada agora, foi também proposta por Ahuja *et al.* [8]. Ela pode ser vista como uma generalização da vizinhança de busca proposta por Saraiha *et al.* [64]. Porém, nessa nova vizinhança, é possível, agora, a movimentação de várias subárvores (todos pertencentes à subárvores de nível 1 distintas). Ou seja, é possível que mais de duas subárvores de nível 1 sejam afetadas a cada movimento realizado,

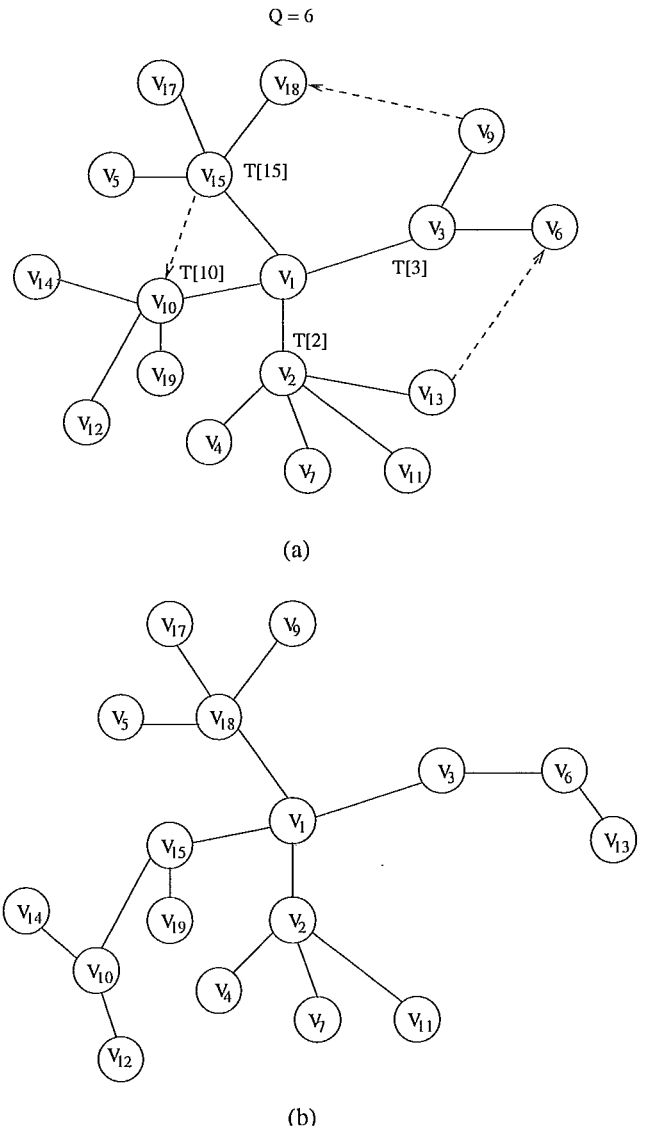

Figura 5.4: Ilustração de uma TCAN (a) Uma AGC antes da TCAN (b) A AGC depois da TCAN

ao contrário da vizinhaça de busca proposta por Saraiha *et al.* [64]. Essa nova vizinhança é denominada *vizinhança de busca baseada na troca-múltipla de subárvores*. Ela pode ser obtida, assim como acontece com a vizinhança de busca baseada na troca-múltipla de nós, através da realização de TC's e TCA's. A seguir, detalharemos como se dá cada uma dessas trocas, agora, movimentando subárvores.

5.4.1 Troca-Cíclica para Movimentar Subárvores (TCS)

Uma TCS pode ser definida como uma sequência de nós $v_2 - v_3 - v_4 - ... - v_r - v_2$, onde os nós $v_2 - v_3 - v_4 - ... - v_r$ pertencem à subárvores de nível 1 distintas, isto é, $T[p] \neq T[q]$ para $p \neq q$. A TCS $v_2 - v_3 - v_4 - ... - v_r - v_2$ representa as seguintes mudanças: Os nós da subárvore T_2 são movidos da subárvore de nível 1, $T[2]$, para a subárvore de nível 1, $T[3]$, os nós da subárvore T_3 são movidos da subárvore de nível 1, $T[3]$, para a subárvore de nível 1, $T[4]$, e assim por diante, e finalmente, os nós da subárvore T_r são movidos da subárvore de nível 1, $T[r]$, para a subárvore de nível 1, $T[2]$. Uma TCS é *viável* se a mesma satisfaz a restrição de capacidade. Mais detalhadamente, a TCS $v_2 - v_3 - v_4 - ... - v_r - v_2$ é uma troca viável se e somente se:

$$\sum_{k \in T[p]} q_k + \sum_{k \in T_{p-1}} q_k - \sum_{k \in T_p} q_k \leq Q, \text{ para todo } p = 2, 3, 4, ..., r, \qquad (5.3)$$

onde $T_1 = T_r$. Somente as TCS's viáveis serão consideradas. Seja T' uma árvore geradora capacitada viável obtida por uma TCS a partir de uma solução também viável T. O custo dessa TCS pode ser definido como $c(T') - c(T)$, ou seja:

$$c(T') - c(T) = \sum_{p=2}^{r} (c(\{S_{p-1}\} \cup S[p] \setminus \{S_p\}) - \sum_{p=2}^{r} c(S[p])) \qquad (5.4)$$

onde $S_1 = S_r$.

É fácil ver, novamente, que uma TCS é interessante somente quando $c(T') < c(T)$. E, além disso, todas as observações feitas em relação às TCN's também valem para as TCS's.

5.4.2 Troca-Caminho para Movimentar Subárvores (TCAS)

Uma TCAS pode ser definida como uma sequência de nós $v_2 - v_3 - v_4 - ... - v_r$, onde os nós $v_2 - v_3 - v_4 - ... - v_r$ pertencem à subárvores de nível 1 distintas, isto é, $T[p] \neq T[q]$ para $p \neq q$. A TCAS $v_2 - v_3 - v_4 - ... - v_r$ representa as seguintes mudanças: Os nós da subárvore T_2 são movidos da subárvore de nível 1, $T[2]$, para a subárvore de nível 1, $T[3]$, os nós da subárvore T_3 são movidos da subárvore de nível 1, $T[3]$, para a subárvore de nível 1, $T[4]$, e assim por diante, e finalmente, os nós da subárvore T_{r-1} são movidos da subárvore de nível 1, $T[r-1]$, para a subárvore de nível 1, $T[r]$. Uma TCAS é *viável* se tal troca satisfaz a restrição de capacidade. Mais especificamente, a TCAS $v_2 - v_3 - v_4 - ... - v_r$ é uma troca viável se e somente se:

$$\sum_{k \in T[p]} q_k + \sum_{k \in T_{p-1}} q_k - \sum_{k \in T_p} q_k \leq Q, \text{ para todo } p = 3, 4, ..., r, \tag{5.5}$$

Somente as TCAS's viáveis serão consideradas. Seja T' uma árvore geradora capacitada viável obtida por uma TCAS a partir de uma solução também viável T. O custo dessa TCAS pode ser definido como $c(T') - c(T)$, ou seja:

$$\begin{aligned} c(T') - c(T) = {} & c(S[2] \setminus \{S_2\}) + \\ & \textstyle\sum_{p=3}^{r-1} (c(\{S_{p-1}\} \cup S[p] \setminus \{S_p\})) + \\ & c(\{S_{r-1}\} \cup S[r]) - \\ & \textstyle\sum_{p=2}^{r} (c(S[p]) \end{aligned} \tag{5.6}$$

Mais uma vez, é fácil ver que uma TCAS é interessante somente quando $c(T') < c(T)$. E, além disso, todas as observações feitas em relação às TCAN's também valem para as TCAS's.

Assim como ocorre com a de vizinhança busca baseada na troca-múltipla de nós, se T tem K subárvores de nível 1 para algum K fixo, então existem $O(n^K)$ possibilidades de TCS's e TCAS's. Note ainda que assim como acontece com a de vizinhança de busca anterior, uma TCAS pode ser transformada em uma TCS, veja [8].

5.5 Vizinhança de Busca Composta

Essa nova vizinhança de busca que denominaremos de *vizinhança de busca composta*, pode ser vista como uma unificação da vizinhança de busca baseada na troca-múltipla de nós com a vizinhança baseada na troca-múltipla de subárvores. Ela foi proposta por Ahuja *et al.* [5]. A exploração dessa nova vizinhança também se dá através de TC's e TCA's. Denominaremos de TCC e TCAC, uma TC e uma TCA, respectivamente, para essa vizinhança.

Uma TCC pode ser definida como uma sequência de nós $v_2 - v_3 - v_4 - ... - v_r - v_2$, onde os nós $v_2 - v_3 - v_4 - ... - v_r$ pertencem a subárvores de nível 1 distintas, isto é, $T[p] \neq T[q]$ para $p \neq q$. A TCC $v_2 - v_3 - v_4 - ... - v_r - v_2$ representa as seguintes mudanças: ou o nó v_2 ou a subárvore T_2 é movido(a) da subárvore de nível 1, $T[2]$, para a subárvore de nível 1, $T[3]$, ou o nó v_3 ou a subárvore T_3 é movido(a) da subárvore de nível 1, $T[3]$, para a subárvore de nível 1, $T[4]$, e assim por diante, e finalmente, ou o nó v_r ou a subárvore T_r é movido(a) da subárvore de nível 1, $T[r]$, para a subárvore de nível 1, $T[2]$. Dessa forma, para cada nó v_k em uma TCC, é possível movimentar o próprio nó v_k ou a subárvore com raiz em v_k, T_k. Podemos definir uma TCAC de forma similar a um TCC. A diferença é que o último nó v_r ou a subárvore T_r, não é movido(a) da subárvore de nível 1, $T[r]$, para subárvore de nível 1, $T[2]$. Assim como acontece com as vizinhanças de busca que possibilitam trocas-múltiplas de nós ou subárvores, uma TCAC pode ser transformada em uma TCC. Os detalhes das computações dos custos de uma TCC e de uma TCAC foram omitidos pelo fato deles serem facilmente extendidos a partir das computações dos custos de uma TC e de uma TCA para as duas últimas

vizinhanças apresentadas. Além disso, não utilizamos essa vizinhança em nossa busca local, como será melhor detalhado mais na frente.

A vizinhança de busca composta possibilita uma exploração de soluções vizinhas substancialmente maior do que as vizinhanças anteriores, pois permite novas possibilidades de movimentos. Para cada nó v_k na TC $v_2 - v_3 - v_4 - ... - v_r - v_2$, uma TCN permite mover um nó da subárvore de nível 1 à qual ele pertence para outra subárvore. Uma TCS permite mover uma subárvore da subárvore de nível 1 à qual ela pertence para outra subárvore de nível 1. Tanto uma TCN quanto uma TCS permite uma única possibilidade de movimento. Porém, uma TCC permite duas possibilidades. Ou o nó v_k, ou a subárbore T_k é movido(a) da subárvore de nível 1 à qual ele(a) pertence para uma outra subárvore de nível 1. Ahuja *et al.* [5] fazem uma análise comparativa entre os tamanhos das vizinhanças que podem ser exploradas pelas últimas três vizinhanças de busca apresentadas.

5.6 Grafo de Melhoria

O número de soluções vizinhas obtidas a partir das vizinhanças de busca baseadas ou na troca-múltipla de nós, ou na troca-múltipla de subárvores, ou ainda da vizinhaça de busca composta, é bastante grande para ser enumerado explicitamente. Ahuja *et al.* [8, 5] propuseram um método heurístico que não enumera toda a vizinhança, mas na prática, é bastante eficiente para identificar soluções mais atraentes dessa vizinhança (ou seja, as de menor custo no caso do PAGCM) a partir de uma solução inicial (dada como entrada). Segundo Ahuja *et al.* [8], esse procedimento heurístico gasta, para encontrar uma solução vizinha melhor que a atual, um tempo computacional comparável ao das vizinhanças de busca propostas por Amberg *et al.* e Saraiha *et al.*. A idéia central desse método é o conceito de *Grafo de Melhoria.*

O conceito de *Grafo de Melhoria* (GM) foi proposto originalmente por Thompson e Orlin [67] e Thompson e Psaraftis [68]. Esse conceito pode

ser utilizado para obter soluções vizinhas de problemas tais quais o Problema de Roteamento de Veículos, Problemas Generalizados de Associação, etc (veja Ahuja *et al.* [7]). Em especial, no caso do PAGCM, o conceito de GM pode ser utilizado para obter, isoladamente, a vizinhança de busca baseada na troca-múltipla de nós, a vizinhança baseada na troca-múltipla de subárvores e a vizinhança de busca composta. A obtenção de um GM é feita sempre a partir de uma solução inicial viável dada como entrada. Um GM para qualquer uma dessas vizinhanças possui algumas características em comum. A primeira delas é que ele é direcionado. Além disso, existe uma correspondência de um-para-um entre os nós em G e os nós desse grafo. E, por fim, uma TC ou uma TCA em relação à uma determinada solução (dada como entrada para a computação do grafo) define um ciclo ou um caminho direcionado em um GM. O custo desse ciclo ou caminho é igual ao custo da troca correspondente. Como uma TCA, seja para movimentar nós ou subárvores ou ainda ambos, pode ser transformada em uma TC, não há perda de generalidade em considerarmos, a título de explicação, somente o caso das TC's. A seguir, mostraremos, detalhadamente, a forma de obter o GM correspondente a cada uma das demais vizinhanças de busca apresentadas.

Seja T uma solução viável para o PAGCM que é passada como entrada para computação de cada um dos GM's dos próximos itens.

5.6.1 GM para a Vizinhança de Busca Baseada na Troca-Múltipla de Nós

O GM $\mathbf{G^1(T)}$ para a vizinhança de busca baseada na troca-múltipla de nós é construído da seguinte forma. Existe uma correspondência de um-para-um entre os nós em G e os nós em $\mathbf{G^1(T)}$, ou seja, o conjunto de nós de $\mathbf{G^1(T)}$ é o mesmo de G. Um arco (v_i, v_j) em $\mathbf{G^1(T)}$ significa que o nó v_i sai da subárvore de nível de 1, $T[i]$, e junta-se a subárvore de nível 1, $T[j]$, e simultaneamente, o nó v_j sai da subárvore de nível 1, $T[j]$. Para construir $\mathbf{G^1(T)}$, todos os pares de nós v_i e v_j em V são considerados e um arco (v_i, v_j)

é adicionado se e somente se (*i*) $T[i] \neq T[j]$ e (*ii*) $S[j] \cup \{v_i\} \setminus \{v_j\}$ é um conjunto de nós viável (ou seja, $d(\{v_i\} \cup S[j] \setminus \{v_j\}) \leq Q$). O custo α_{ij} de um arco (v_i, v_j) em $\mathbf{G^1(T)}$ é definido como $\alpha_{ij} = c(\{v_i\} \cup S[j] \setminus \{v_j\}) - c(S[j])$. Um ciclo direcionado $v_2 - v_3 - v_4 - ... - v_r - v_2$ no GM $\mathbf{G^1(T)}$ é um *subconjunto disjunto* se as subárvores de nível 1, $T[2] - T[3] - T[4] - ... - T[r]$ são distintas, ou seja, $T[p] \neq T[q]$ para $p \neq q$.

Para enumerar os arcos em $\mathbf{G^1(T)}$ devemos analisar a expressão que calcula o custo de cada arco, ou seja, $\alpha_{ij} = c(\{v_i\} \cup S[j] \setminus \{v_j\}) - c(S[j])$. Primeiramente, assumiremos que o custo $c(S[j])$ de cada uma das subárvores de nível 1 em T, é sempre conhecido. Por conseguinte, para computar α_{ij} é necessário calcular $c(\{v_i\} \cup S[j] \setminus \{v_j\})$. Esse cáculo envolve a retirada do nó v_j de $T[j]$ e adição do nó v_i nessa mesma subárvore para então determinar o custo de uma árvore geradora de custo mínimo. Para resolver um problema da árvore geradora de custo mínimo a partir de $\{v_i\} \cup S[j] \setminus \{v_j\}$, é necessário considerar o conjunto de arcos obtido pela união dos arcos da árvore geradora de custo mínimo obtida a partir de $T[j] \setminus \{v_j\}$ com os arcos (v_i, v_w) para todo $v_w \in S[j] \setminus \{v_j\}$. Para tanto, é necessário resolver um problema da árvore geradora de custo mínimo a partir desse subgrafo que possui $O(Q)$ nós. Isso pode ser feito utilizando o algoritmo de Kruskal [4] em $O(Q \, log \, Q)$. A resolução de uma árvore geradora de custo mínimo a partir de $T[j] \setminus \{v_j\}$ leva $O(Q^2)$. Feito isso, um problema da árvore geradora de custo mínimo é resolvido a partir do conjunto de nós $\{v_i\} \cup S[j] \setminus \{v_j\}$ para cada $V \setminus T[j]$ e então é determinado o custo do arco correspondente em $\mathbf{G^1(T)}$. Desta forma, é possível determinar o custo de todos os arcos que entram no nó v_j em $O(Q^2) + O(Q(n - Q)) = O(nQ)$. Consequentemente, os custos de todos os arcos podem ser obtidos em $O(n^2Q)$. Vale lembrar que para a análise da complexidade acima, foi assumido que todas as demandas são idênticas.

5.6.2 GM para a Vizinhança de Busca Baseada na Troca-Múltipla de Subárvores

O GM $\mathbf{G^2(T)}$ para a vizinhança de busca baseada na troca-múltipla de subárvores é construído da seguinte forma. Existe uma correspondência de um-para-um entre os nós em G e os nós em $\mathbf{G^2(T)}$, ou seja, o conjunto de nós de $\mathbf{G^2(T)}$ é o mesmo de G. Um arco (v_i, v_j) em $\mathbf{G^2(T)}$ significa que a subárvore T_i sai da subárvore de nível de 1, $T[i]$, e junta-se a subárvore de nível 1, $T[j]$, e simultaneamente, a subárvore T_j sai da subárvore de nível 1, $T[j]$. Para construir $\mathbf{G^2(T)}$, todos os pares de nós v_i e v_j em V são considerados e um arco (v_i, v_j) é adicionado se e somente se (i) $T[i] \neq T[j]$ e (ii) $S[j] \cup \{T_i\} \setminus \{T_j\}$ é um conjunto de nós viável (ou seja, $d(\{T_i\} \cup S[j] \setminus \{T_j\}) \leq Q$). O custo α_{ij} de um arco (v_i, v_j) em $\mathbf{G^2(T)}$ é definido como $\alpha_{ij} = c(\{T_i\} \cup S[j] \setminus \{T_j\}) - c(S[j])$. Um ciclo direcionado $v_2 - v_3 - v_4 - ... - v_r - v_2$ em $\mathbf{G^2(T)}$ é um *subconjunto disjunto* se as subárvores de nível 1, $T[2] - T[3] - T[4] - ... - T[r]$ são distintas, ou seja, $T[p] \neq T[q]$ para $p \neq q$.

Assim como acontece no cáculo dos custos dos arcos de $\mathbf{G^1(T)}$, para computar os custos dos arcos de $\mathbf{G^2(T)}$ devemos analisar a expressão que define o custo de cada arco, ou seja, $\alpha_{ij} = c(\{T_i\} \cup S[j] \setminus \{T_j\}) - c(S[j])$. O segundo termo, $c(S[j])$, novamente, é assumido ser conhecido. O primeiro termo, $c(\{T_i\} \cup S[j] \setminus \{T_j\})$, envolve o cálculo de uma árvore geradora de custo mínimo a partir do conjunto de nós $\{S_i\} \cup S[j] \setminus \{S_j\})$. Para tanto, é necessário remover a subárvore T_j com raiz em v_j (obtemos, com isso, uma árvore geradora de custo mínimo $T[j] \setminus T_j$ a partir do conjunto de nós $S[j] \setminus S_j$). Repare que não é necessário recalcular uma AGM, ao removermos T_j, obtemos imediatamente a AGM $T[j] \setminus T_j$). Em seguida, é adicionado a $T[j] \setminus T_j$, o conjunto de nós S_i e todos os arcos entre S_i e $S[j] \setminus S_j$ e resolvido um problema da árvore geradora de custo mínimo. O resultado desse procedimento é uma árvore geradora de custo mínimo a partir do conjunto de nós $\{S_i\} \cup S[j] \setminus \{S_j\})$. Assumindo que todas as demandas são idênticas, podemos resolver uma árvore geradora de custo mínimo a partir de um subgrafo com Q nós e Q^2 arcos pelo algoritmo de Prim [61] em $O(Q^2)$. Portanto, computar

um simples α_{ij}, leva $O(Q^2)$. Consequemente, a computação de todos os α_{ij}, leva $O(n^2Q^2)$.

5.6.3 GM para a Vizinhança de Busca Composta

O GM $\mathbf{G^3(T)}$ para a vizinhança de busca composta pode ser obtido combinando as idéias das obtenções de $\mathbf{G^1(T)}$ e $\mathbf{G^2(T)}$. A computação detalhada de $\mathbf{G^3(T)}$ pode ser encontrada em Ahuja *et al.* [5]. Não entraremos nos detalhes dessa computação porque utilizamos em nosso trabalho, como será melhor detalhado mais na frente, somente os GM's $\mathbf{G^1(T)}$ e $\mathbf{G^2(T)}$.

O seguinte resultado mostra a correspondência entre as TC's e os ciclos nos GM's $\mathbf{G^1(T)}$ e $\mathbf{G^2(T)}$.

Lema 5.1 *Existe uma correspondência de um-para-um entre as trocas-cíclicas baseadas em nós em relação à uma árvore T e os ciclos direcionados em subconjuntos disjuntos no GM $\mathbf{G^1(T)}$, e ambos têm o mesmo custo. Da mesma forma, existe uma correspondência de um-para-um entre as trocas-cíclicas baseadas em subárvores em relação à uma árvore T e os ciclos direcionados em subconjuntos disjuntos no GM $\mathbf{G^2(T)}$, e ambos têm o mesmo custo.*

A prova de (5.1) e um outro lema nessa mesma linha, porém em relação ao $\mathbf{G^3(T)}$, podem ser encontrados em Ahuja *et al.* [8, 5].

5.6.4 Computação e Atualização de um Grafo de Melhoria

Na construção de um GM é necessário determinar o custo do arco (v_i, v_j) para cada arco (v_i, v_j). A princípio, pode parecer que o tempo necessário para construir um Grafo de Melhoria é $\Omega(n^2)$ vezes o tempo necessário para

calcular o custo de um arco (v_i, v_j). Porém, é bem provável que ao computarmos o custo de um arco (v_i, v_j), seja possível extrair alguma informação interessante sobre a computação de um arco (v_i, v_k) e isso talvez possa reduzir o tempo total de criação de um GM (no entanto, a identificação dessas informações é, atualmente, tema de pesquisa, veja Ahuja *et al* [7]).

A atualização de um GM após uma TC, pode ser feita sem que seja necessário a recomputação de todos os arcos. Se dois nós v_i e v_j pertencem à duas subárvores de nível 1 distintas que não foram afetados pela troca, então os custos dos arcos formados por nós pertencentes às subárvores $T[i]$ e $T[j]$, $i \neq j$, permanecem inalterados. De fato, após uma TC, é necessário atualizar somente os arcos incidentes aos nós pertencentes às subárvores de nível 1 que foram afetadas na troca (Isso vale apenas para o GM da vizinhança de busca baseada na troca-múltipla de nós. Para os outros GM's é necessário atualizar também os arcos que saem dos nós pertencentes às subárvores de nível 1 que foram afetadas na troca). Essa observação diminui substancialmente o tempo de atualização de um GM.

5.6.5 Identificação de Ciclos Negativos em Subconjuntos Disjuntos

De acordo com o lema (5.1), uma TC (que leva a uma solução de menor custo) em relação à uma solução viável T, corresponde a um ciclo direcionado de custo negativo em *subconjuntos disjuntos* (CDSD) em qualquer um dos GM's mencionados acima. Embora existam algoritmos de tempo polinomial para identificar ciclos negativos (sem a restrição dos elementos pertencerem à conjuntos distintos) (Ahuja *et al.* [4]), o problema de encontrar um CDSD é NP-Completo (Thompson e Orlin [67]). Por comodidade, denominaremos de *ciclo válido* um CDSD.

Em Ahuja *et al.* [8], um método heurístico foi proposto para identificar ciclos válidos. Sua complexidade é da ordem de $\Omega(n^3Q)$. Apesar de ter sido

utilizado com sucesso, esse método não garante a identifição de um ciclo válido se o mesmo existir. Para suprir essa deficiência, Ahuja *et al.* [5] propuseram um método exato para identificar ciclos válidos. Tal método pode identificar o CDSD de custo mais negativo (possivelmente, o que acarretaria o maior impacto na função objetivo). Esse método foi implementado por nós, porém, o seu alto custo computacional inviabializou sua utilização nesse estudo, como será melhor detalhado mais na frente. Um estudo detalhado sobre a identificação do CDSD de custo mais negativo (com propostas de solução heurísticas e exatas) pode ser encontrado em Ahuja *et al.* [3].

5.7 Um Algoritmo de uma Busca Local que utiliza o conceito de GM

Após termos apresentado as vizinhanças de busca que permitem trocas-múltiplas de nós e /ou subárvores e a forma como podemos explorá-las (o conceito de GM e os procedimentos que podemos utilizar para identifiçar ciclos válidos), apresentaremos agora, um algoritmo de busca local que utiliza todos esses conceitos. Seja $\mathbf{G(T)}$ um dos GM's apresentados anteriormente($\mathbf{G^1(T)}$ ou $\mathbf{G^2(T)}$ ou $\mathbf{G^3(T)}$) obtido a partir de uma solução inicial dada, T. O algoritmo pode então ser descrito por:

Algoritmo Busca-Local-PAGCM
Início
Obter uma solução viável T
Construir o Grafo de Melhoria $\mathbf{G(T)}$
Enquanto $\mathbf{G(T)}$ possuir ciclos válidos **faça**
Início
Obter um ciclo válido W em $\mathbf{G(T)}$;
Utilizar a troca-múltipla corresponde a W e atualizar a solução T;
Atualizar $\mathbf{G(T)}$
Fim

Fim

Vale enfatizar que no algoritmo acima, a computação e atualização do GM é realizada da forma como foi descrito na seção 5.6. O mesmo acontece em relação à identifição de ciclos válidos.

No próximo capítulo, apresentaremos algumas das Metaheurísticas que foram propostas na literatura para o PAGCM. A maioria das Metaheurísticas que serão apresentadas, têm suas buscas locais baseadas em alguma(s) das vizinhanças de busca apresentadas anteriormente.

Capítulo 6

Metaheurísticas para o PAGCM

Como mencionado anteriormente, algumas Metaheurísticas diferentes foram propostas na literatura para resolver o PAGCM. Mais a frente, quando formos tratar dos resultados computacionais obtidos, mostraremos que muitos dos limites superiores obtidos por algumas dessas Metaheurísticas, são, na realidade, soluções ótimas. Nas próximas seções, apresentaremos as principais Metaheurísticas que foram propostas.

6.1 Busca Tabu e *Simulated Annealing* com 2-Trocas de Nós

Amberg *et al.* propõem uma Busca Tabu [34] e um *Simulated Annealing* (*A*arts e Korst [1]) para o PAGCM. Ambas as estratégias iniciam com uma solução obtida pela heurística Esau-Williams [23]. A estratégia de busca local adotada e que é comum às duas Metaheurísticas é baseada na vizinhança de busca apresentada na seção 5.1 (transferência de um nó e troca de nós dois-a-dois). Somente as instâncias (mantidas em [14]) que possuem demandas idênticas foram consideradas. Muitos dos limites superiores, aqui obtidos, foram provados, em outras referências, como sendo soluções ótimas. Mais a frente, detalharemos melhor o procedimento utilizado para provar essas

otimalidades.

6.2 Busca Tabu com 2-Trocas de Subárvores

Saraiha *et al.* [64] propõem uma Busca Tabu para o PAGCM. Nessa proposta, a solução inicial é obtida através de uma adaptação do algoritmo de Prim [61] para o problema. A estratégia de busca local adotada é baseada na vizinhança que foi apresentada na seção 5.2 (operações *cut* e *paste*). Aqui, foram tratadas tanto instâncias com demandas idênticas quanto aquelas com demandas heterogêneas. Muitos dos limites superior obtidos, também foram provados, em outras referências, como sendo soluções ótimas. Em especial, houve, para a época, uma melhoria significativa na qualidade dos limites superiores, até então conhecidos, para quase todas as instâncias que possuem demandas heterogêneas.

6.3 Busca Tabu e GRASP com Grafo de Melhoria

Ahuja *et al.* [8] propõem uma Busca Tabu e um GRASP ([24]) para o PAGCM. A solução inicial, tanto para a Busca Tabu como para o GRASP, é obtida através de um versão aleatorizada da heurística EW. Nessa versão, em cada iteração são determinados os p *savings* mais negativos para algum valor pequeno de p. Um número aleotório k é gerado entre 1 e p e então o k-ésimo *saving* é escolhido (passo 1 da EW). As estratégias de busca local utilizadas são baseadas nas vizinhanças de busca apresentadas nas seções 5.4 (troca-múltipla de subárvores) e 5.3 (troca-múltipla de nós). Foram implementadas duas versões de cada uma das Metaheurísticas. Uma versão do GRASP e da Busca Tabu tendo como estratégia de busca local, a troca-múltipla de nós. E uma outra versão do GRASP e da Busca Tabu tendo como estratégia de busca local, a troca-múltipla de subárvores. Para definir as vizinhanças adotadas foi utilizado o conceito de GM (seção 5.6). Esse conceito foi utilizado tanto

para obter a vizinhança proporcionada pela troca-múltipla de nós quanto a proporcionada pela troca-múltipla de subárvores. Para identificar TC's e/ou TCA's, foi utilizada uma heurística também proposta em Ahuja *et al.* [8]. Os resultados computacionais obtidos nesse trabalho foram muito bons. Foram obtidos ou melhorados os melhores limites superiores até então conhecidos para todas as instâncias da literatura (foram tratadas tanto instâncias com demandas idênticas quanto com demandas heterogêneas) mantidas em [14]. Muitos dos limites superiores obtidos para as instâncias com demandas homogêneas, foram provados como sendo soluções ótimas em outras referências (mais a frente, detalharemos melhor como foi realizado esse processo de provar otimalidades).

6.4 GRASP com Grafo de Melhoria Generalizado

Ahuja *et al.* [5] propõem um novo GRASP para o PAGCM. A solução inicial é obtida através de um versão aleatorizada da heurística EW (aleatorização essa, feita da mesma forma como foi explicado no trabalho anterior). A estratégia de busca local utilizada é baseada na vizinhança de busca composta (seção 5.5). Assim como no trabalho anterior, foi utilizado o conceito de GM para explorar a vizinhança proporcionada pela vizinhança de busca composta. Ao contrário do outro trabalho, um método exato, também proposto em Ahuja *et al.* [5], foi utilizado para identificar TC's. Mais uma vez, os resultados aqui obtidos foram muito bons. Em relação as instâncias com demandas idênticas, somente uma instância teve seu limite superior melhorado em relação aos resultados obtidos em [8]. Nas demais instâncias com demandas idênticas, foram obtidos os mesmos limites superiores encontrados em [8]. Porém, em relação as instâncias com demandas heterogêneas foram melhorados muitos dos limites superiores em [8], principalmente de instâncias com $n = 200$. Onde não houve melhoria, foram obtidos os mesmos resultados encontrados em [8].

6.5 GRASP com *Path-Relinking*

Souza *et al.* [65] propõem mais um GRASP para o PAGCM. A solução inicial é obtida através de um versão aleatorizada da heurística EW (um pouco diferente da aleatorização proposta no trabalho citado anteriormente). A estratégia de busca local utilizada é baseada em uma vizinhança de busca denominada *path-relinking* proposta por Glover [33]. Somente instâncias com demandas idênticas foram tratadas. Muitos dos limites superiores aqui obtidos, são iguais aos obtidos em [8, 5]. Não houve nenhuma melhoria em relação aos resultados obtidos em [8, 5].

Capítulo 7

Heurísticas Lagrangeanas para o PAGCM

Apresentaremos nesse capítulo, as Heurísticas Lagrangeanas que implementamos para o PAGCM. As Heurísticas aqui apresentadas, terão enfoque mais algorítmico do que de implementação, ou seja, não haverá preocupação com os detalhes de implementação bem como com a linguagem de programação utilizada. O mesmo acontecerá em relação as estruturas de dados. Porém, vale enfatizar que é essencial uma estrutura que mantenha, de forma eficiente, os descendentes de cada nó e o peso acumulado (somatório das demandas) de um determinado nó e seus descendentes em uma solução viável.

Ao longo desse capítulo, defineremos o que vem a ser uma Heurística Lagrangeana, mostraremos os seus componentes e as estratégias de busca local que utilizamos. E, por fim, os algoritmos de cada uma das Heurísticas Lagrangeanas que implementamos.

7.1 Heurísticas Lagrangeanas

Uma Heurística Lagrangeana é um procedimento que transforma soluções dual viáveis (obtidas através de Relaxação Lagrangeana) em soluções primal viáveis [12]. Ela é utilizada no contexto de um algoritmo de Otimização por Subgradientes onde os multiplicadores de Lagrange estão disponíveis, as-

sim como a solução ótima do Subproblema Lagrangeano associado a esses multiplicadores. Nesse trabalho, a noção do que seja uma Heurística Lagrangeana será expandida para incorporar (a exemplo das Metaheurísticas) uma heurística construtiva inicial. A idéia básica seria então utilizar soluções dual viáveis para influenciar a determinação de soluções primal viáveis pela heurística construtiva. Mais especificamente, soluções dual viáveis serão utilizadas para modificar os custos das variáveis originais quando passadas à heurística de construção. Nesse estudo, a heurística EW será utilizada como heurística de construção. Heurísticas Lagrangeanas, da forma como definimos, foram aplicadas com sucesso ao Problema de Steiner em Grafos [50], Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau nos Vértices em [10], Problema de Roteamento de Veículos [57] e ao Problema de Partição Ótima de Retângulos [16] entre outros.

As Heurísticas Lagrangeanas que introduziremos aqui, são compostas, basicamente, de quatro etapas. Numa primeira etapa, uma solução viável inicial é obtida através da execução da heurística EW. Em um contexto mais amplo, o Algoritmo *Relax and Cut* para o PAGCM é aplicado vizando obter $\overline{Z}_{lb}(u,v)(3.14)$. Ao longo das iterações desse processo, um conjunto de multiplicadores de Lagrange é gerado. Para ums certa iteração r, os multiplicadores $\{t_{ik}^{r-1} : \forall i \in I, \forall k \in \overline{I}, u_{ij}^{r-1} : (v_i, v_j) \in A\}$ servem de entrada para a geração de custos Lagrangeanos $\{f_{ij} : (v_i, v_j) \in A\}$. A segunda etapa consiste na resolução do Subproblema Lagrangeano que gera uma solução $\{\overline{x}_{ij} : (v_i, v_j) \in A\}$. Numa terceira etapa, a heurística básica é aplicada novamente, porém ela utiliza, agora, como entrada, ou os custos Lagrangeanos, ou os custos Reduzidos (de Programação Linear) associados à solução do Subproblema Lagrangeano ou, ainda, os custos Complementares, a serem definidos posteriormente. Por fim, numa quarta etapa, a solução obtida pela aplicação da heurística básica é submetida a um processo de busca local. O processo de busca local não é mais efetuado a partir dos custos que serviram de entrada para a aplicação da heurística básica e sim a partir dos custos originais $\{c_e : e \in E\}$. Mais adiante, detalharemos as nossas estratégias

de busca local. A maneira pela qual os custos dos arcos são introduzidos na terceita etapa de uma de nossas Heurísticas Lagrangeanas, define o tipo de Heurística Lagrangeana que está sendo aplicada. É importante enfatizar que nossas Heurísticas Lagrangeanas são acionadas em toda iteração do Algoritmo *Relax and Cut* para o PAGCM (passo 4).

No próximo item, mostraremos, mais detalhadamente, todos os componentes das Heurísticas Lagrangeanas que introduziremos. Apresentaremos, ainda, os motivos pelos quais adotamos certas estratégias nas composições das mesmas.

7.1.1 Componentes da Heurística Lagrangeana

A primeira etapa de nossa Heurística Lagrangeana, ou seja, determinação de uma solução inicial para o PAGCM, é feita pela heurística EW. A segunda etapa é a resolução do Subproblema Lagrangeano. Como já foi mencionado, utilizamos o algoritmo proposto em [25] para resolvê-lo. Na terceira etapa, fizemos alguns experimentos computacionais. Inicialmente, utilizamos os custos Lagrangeanos como entrada para a terceira fase da Heurística Lagrangeana. Posteriormente, fizemos a experiência de utilizar não mais esses custos e sim os custos Lagrangeanos Reduzidos $\{\zeta(i,j) : (v_i, v_j) \in A\}$ fornecidos por um algoritmo também proposto em [25]. E, por fim, testamos os custos Complementares, ou seja, se $\overline{x}$ é a solução ótima do problema Lagrangeano, os custos de entrada para a EW são dados por $\{c_{ij}(1 - \overline{x}_{ij} - \overline{x}_{ji}) : [v_i, v_j] \in E\}$. Note que essa estrutura de custos torna mais atraentes, para a solução primal, variáveis assumindo valor 1 na solução Lagrangeana. Após analisarmos os resultados dessas experiências, verificamos que a heurística EW tendo como entrada os custos Complementares forneceu limites superiores melhores do que quando executada tendo como entrada os custos Lagrangeanos ou Reduzidos.

Além dos componentes citados acima, existem outros componentes importantes em uma Heurística Lagrangeana. Os critérios de parada do Al-

goritmo (*Relax and Cut* para o PAGCM) e a forma pela qual se reduz o valor de α ao longo das iterações do algoritmo, são partes importantes de uma Heurística Lagrangeana (veja seção 3.4). Uma outra componente importante é a fixação de variáveis de decisão e de variáveis de fluxo (seção 3.5). Antes de incorporarmos os testes de fixação de variáveis da seção 3.5, fizemos alguns experimentos computacionais com e sem esses testes. Após analisarmos os resultados, verificamos que as Heurísticas Lagrangeanas acionadas em conjunto com os testes de fixação, produziram melhores limites superiores do que quando acionadas sem os mesmos. Além disso, verificamos que a qualidade do limite inferior obtido também melhorou. Uma outra observação importante decorrente desses experimentos foi que quando a diferença entre o limite superior e o limite inferior era pequena (até 3%), o número de variáveis de decisão restantes após o término do algoritmo era de um pouco menos de 10% do número de variáveis de decisão original. Como mencionado anteriormente, os testes de fixação são acionados somente quando ocorre um aumento global do melhor limite inferior conhecido ao longo das iterações do algoritmo *Relax and Cut* para o PAGCM, pois é necessário um certo esforço computacional para executar esses testes. Diante de um saldo tão postivo, esses testes foram mantidos.

E, por fim, a última componente de uma Heurística Lagrangeana, como mencionado anteriormente, é a estratégia de busca local adotada (quarta etapa). A estratégia de busca local que utilizamos procura melhorar a solução inicial obtida na terceira etapa, tentando, isoladamente e em sequência, trocar subárvores duas-a-duas, trocar nós dois-a-dois e transferir nós. Essas estratégias serão detalhadas no próximo item.

7.1.2 Estratégias de Busca Local adotadas nas Heurísticas Lagrangeanas

A definição da estratégia de busca que adotamos foi baseada em alguns experimentos computacionais que realizamos. Nossa idéia inicial era aplicar

o algoritmo da seção 5.7 utilizando o GM $\mathbf{G^1(T)}$ e o algoritmo exato proposto em Ahuja *et al.* [5] para identificar ciclos válidos. Porém, verificamos que o custo computacional requerido por essa estratégia era muito alto. Em particular, o algoritmo utlizado para identificação de ciclos válidos gastava um tempo computacional bastante considerável. Além disso, analisando os experimentos computacionais realizados, verificamos que não ocorreram melhorias do limite superior tão significativas que justificasse a utilização dessa estratégia. Isso descartou, de cara, a opção de usarmos a mesma estratégia, porém utilizando o GM $\mathbf{G^2(T)}$, já que o problema não era a construção e atualização do GM utilizado e sim o algoritmo para identifição de ciclos válidos que era muito lento. Procuramos, então, uma alternativa que fosse ao mesmo tempo forte (no sentido de produzir boas melhorias no limite superior) e de custo computacional barato. Vale lembrar que nossas estratégias são baseadas na utilização dos GM's $\mathbf{G^1(T)}$ e $\mathbf{G^2(T)}$.

Direcionamos nosso estudo então, no sentido de encontrar uma forma de aproveitarmos a vizinhança proporcionada pelos GM's ($\mathbf{G^1(T)}$ e $\mathbf{G^2(T)}$), sem que fosse necessário utilizarmos um algoritmo computacionalmente caro, para identificar soluções vizinhas de boa qualidade. Pensamos, então, em um algoritmo para identificar ciclos válidos com no máximo dois arcos (tanto em $\mathbf{G^1(T)}$ quanto em $\mathbf{G^2(T)}$). A identifição desses ciclos, uma vez computado um GM, é fácil de implementar e não é tão dispendiosa computacionalmente, como será mostrado mais adiante. Além disso, tentamos, também, transferências de nós (veja seção 5.1).

Nossa estratégia ficou então articulada em três passos. Os dois primeiros passos têm como base o algoritmo da seção 5.7 (**Busca-Local-PAGCM**). A única restrição é que os ciclos válidos devem possuir somente dois arcos. Seja T uma solução viável obtida na terceira etapa de uma de nossas Heurísticas Lagrangeanas. Essa solução é então submetida aos seguintes passos. Num primeiro passo, o algoritmo **Busca-Local-PAGCM** é acionado com $\mathbf{G(T) = G^2(T)}$. O segundo passo é similar ao primeiro, as diferenças são que o GM utilizado é o $\mathbf{G^1(T)}$ e a solução que é passada como entrada

é a obtida após o primeira passo. E, por fim, num terceiro passo, tentamos fazer transferências de nós como foi apresentado na seção 5.1 (com a solução obtida após o segundo passo). Essa estratégia é sequencial, ou seja, primeiro submetemos T ao primeiro passo (obtendo T'), depois submetemos T' ao segundo (obtendo T'') e finalmente, submetemos T'' ao terceiro passo (obtendo T'''). Essa sequência de passos é realizada uma única vez para evitarmos que seja gasto um tempo computacional considerável. Essa escolha foi feita após realizarmos alguns experimentos computacionais. Fizemos experimentos variando a ordem dessa sequência e verificamos que a sequência apresentada acima, foi a que produziu os melhores limites superiores.

É fácil ver que para identificar um ciclo válido com dois arcos em $\mathbf{G^1(T)}$ ou $\mathbf{G^2(T)}$, basta verificarmos se $(\alpha_{ij} + \alpha_{ji} < 0)$, onde α_{ij} é o custo de um arco (v_i, v_j) em $\mathbf{G^1(T)}$ ou $\mathbf{G^2(T)}$. Heuristicamente, resolvemos fazer a identificação desses ciclos da seguinte forma. Para cada nó v_i, $i = 2, 3, 4, ..., n$, identificamos (nessa ordem crescente dos índices i) o melhor par de troca (de maior impacto na função objetivo) para o nó v_i, digamos (v_i, v_j). Porém, esse processo tem uma restrição. Caso identifiquemos, por exemplo, o par de troca (v_i, v_j) como sendo o melhor par para o nó v_i, automaticamente, mais nenhum par de troca poderá ser formado por nós pertencentes às duas subárvores às quais os nós v_i e v_j pertencem ($T[i]$ e $T[j]$). É possível ainda que sejam realizadas mais de uma troca a cada iteração, basta que os pares de trocas identificados atendam a restrição mencionada acima, ou seja, pertençam à subárvores distintas.

Como foi mencionado, a diferença entre as duas primeiras etapas está no GM utilizado. Se o GM utilizado for $\mathbf{G^1(T)}$, uma troca (v_i, v_j) significa que o nó v_i é movido da subárvore de nível 1, $T[i]$, para a subárvore de nível 1, $T[j]$, e nó v_j é movido da subárvore de nível 1, $T[j]$, para a subárvore de nível 1, $T[i]$. Caso o GM utilizado seja o $\mathbf{G^2(T)}$, um par de troca (v_i, v_j) significa que a subárvore T_i é movida da subárvore de nível 1, $T[i]$, para a subárvore de nível 1, $T[j]$, e a subárvore T_j é movida da subárvore de nível 1, $T[j]$, para a subárvore de nível 1, $T[i]$.

A seguir, definiremos formalmente o algoritmo que foi utilizado nas dois primeiros passos de nossa busca local. Seja L_T uma lista dos possíveis pares de troca (v_i, v_j) a serem realizadas em cada iteração de nosso algoritmo. Sejam ainda *melhoria* uma variável booleana (que diz se ainda existe alguma troca que leva a um decréscimo da função objetivo), e L_S, a lista de subárvores que possuem um nó (caso o GM utilizado seja $\mathbf{G^1(T)}$) ou uma subárvore (caso o GM utilizado seja $\mathbf{G^2(T)}$) quando ainda estamos procurando por pares de troca que levam a uma melhoria da função objetivo. Introduza ainda, as variáveis v_i^t e v_j^t que poderão armazenar possíveis pares de troca, e a variável ct_{troca} que é utilizada para armazenar o custo da melhor troca (a mais negativa) para cada nó v_i, $i = 2, 3, 4, ..., n$. O nosso algoritmo de busca local que identifica ciclos válidos de dois arcos em $\mathbf{G(T)}$, onde $\mathbf{G(T)} = \mathbf{G^1(T)}$ ou $\mathbf{G(T)} = \mathbf{G^2(T)}$, pode ser assim descrito.

Algoritmo Busca-Local-Ciclo-2

(1) **início**
(2) Obter uma solução viável T
(3) Construir o Grafo de Melhoria $\mathbf{G(T)}$
(4) *melhoria* = verdadeiro;
(5) $L_S = \emptyset$;
(6) $L_T = \emptyset$;
(7) **enquanto** melhoria **faça**
(8) **início**
(9) *melhoria* = falso;
(10) **para** i=2 até n **faça**
(11) **início**
(12) $v_i^t = v_i$;
(13) $ct_{troca} = 0$;
(14) $v_j^t = 0$;
(15) **para** j=2 até n **faça**
(16) se $((T[i] \neq T[j])$ e $(T[i] \notin L_S)$ e $(T[j] \notin L_S)$

e $(\alpha_{ij} + \alpha_{ji} < ct_{troca}))$

então início

(17) $ct_{troca} = \alpha_{ij} + \alpha_{ji}$;

(18) $v_j^t \quad = v_j$;

(19) **fim-se**

(20) **se** $(v_j^t \neq 0)$ // Existe alguma troca que proporciona melhoria

então início

(21) *melhoria* = verdadeiro;

(22) $L_S \quad = L_S \cup \{T[i], T[j]\}$;

(23) $L_T \quad = L_T \cup \{(v_i^t, v_j^t)\}$;

(24) **fim-se**

(25) **fim-para**

(26) **se** *melhoria*

então início

(27) Efetuar a lista de trocas L_T e atualizar a solução T;

(28) Atualizar $\mathbf{G(T)}$;

(29) $L_S = \emptyset$;

(30) $L_T = \emptyset$;

(31) **fim-se**

(32) **fim**

Os passos (2)-(6) servem para inicializar o algoritmo (obtenção de uma solução viável, construção de $\mathbf{G(T)}$ e inicialização de algumas variáveis). Os passos (10)-(19) são utilizados para identificar o melhor par de troca para cada nó v_i. Por exemplo, em uma certa iteração, o algoritmo está procurando um par de troca para um nó v_i. Digamos que existam alguns pares de troca possíveis e que o melhor par de troca é (v_i^t, v_j^t), onde $v_i^t = v_i$. Após isso, como existe um par de troca que leva a uma melhoria, devemos acrescentar em L_S (22), as subárvores às quais os nós v_i^t e v_j^t pertencem, ou seja, $T[i]$ e $T[j]$. Isso serve para evitar que os nós (se $\mathbf{G(T)} = \mathbf{G^1(T)}$) ou subárvores (se $\mathbf{G(T)} = \mathbf{G^2(T)}$) de $T[i]$ e $T[j]$ possam fazer parte de algum outro par de

troca. Além disso, o par de troca (v_i^t, v_j^t) é incluído em L_T (23). Em (26) um teste é utilizado para verificar se existe alguma troca em L_T (isso é sinal de que é possível melhorar a solução atual). Caso esse teste seja positivo, a lista de trocas L_S é efetuada, levando a uma nova solução (27). Após isso, $\mathbf{G(T)}$ é atualizado e L_S e L_T são reinicilizadas. Todo o processo de identifição de pares de trocas é realizado novamente, agora, a partir de $\mathbf{G(T)}$ atualizado. Caso não exista mais nenhum par de troca que leve a uma melhoria da função objetivo, o algoritmo pára.

O terceiro passo de nossa estratégia, como foi mencionado anteriormente, é a transferência de nós. Basicamente, a identificação de transferências válidas (que não violam a restrição de capacidade e diminuem o valor da função objetivo) é realizada da mesma forma que identificamos trocas duas-a-duas (de nós ou subárvores). Procuramos para cada nó $v_i \in \overline{V}$, a melhor transferência e então evitamos que as subárvores envolvidas nessa transferência, participem de qualquer outra transferência, na iteração corrente, já que é possível mais de uma transferência por iteração. Vale lembrar aqui, que também tentamos transferência de subárvores, porém tais movimentos, quase não melhoraram os nossos limites superiores.

Seja T uma solução viável que é passada como entrada para a etapa de transferências. Para agilizar o processo de identificação de transferências válidas, utilizamos uma idéia proposta em Ahuja *et al.* [8]. Porém, aqui, não a usamos exatamente como sugerido em [8], ou seja, como uma forma de transformar uma TCA em uma TC. A adéia aqui, é montarmos um grafo, que denominaremos *Grafo de Transferência*, $\mathbf{G^T}$. Antes de construirmos $\mathbf{G^T(T)}$, precisamos criar um nó denominado *pseudono*, para cada uma das subárvores de T. Além desses nós, $\mathbf{G^T(T)}$ possui ainda, todos os nós do grafo original, G. Os arcos de $\mathbf{G^T(T)}$ serão definidos de modo que o custo de cada um deles, seja o custo da transferência de um nó da subárvore à qual ele pertence para uma outra subárvore, caso a restrição de capacidade não seja violada. O custo da transferência de um nó $v_i \in \overline{V}$ da subárvore $T[i]$ para o pseudono (subárvore) $T[h]$, $T[i] \neq T[h]$ e $\sum_{k \in T[j]} q_k + q_i \leq Q$ (a capacidade

não é violada), é calculado da seguinte forma:

$$\beta_{ih} = c(S[i] \setminus \{v_i\}) - c(S[i]) + c(\{v_i\} \cup S[h]) - c(S[h]), \qquad (7.1)$$

e pode ser determinado pela resolução (reotimização) de dois problemas da árvore geradora de custo mínimo, como foi mostrado na seção 5.1.

É fácil ver que para identificarmos uma transferência válida, basta verificarmos se $\beta_{ih} < 0$. Caso isso seja verdade, β_{ih} identifica uma tranferência válida. Portanto, uma vez computado um $\mathbf{G^T(T)}$, a adaptação do **Algoritmo Busca-Local-Ciclo-2** para encontrar transferências válidas em $\mathbf{G^T(T)}$ é bem simples. A atualização desse grafo é realizada da mesma forma que atualizamos um GM. Somente os arcos incidentes nos pseudonos que foram afetados pelas trocas realizadas serão recomputados.

Após detalharmos cada uma das três etapas de nossa busca local (Busca-Local-Lag), podemos definir, algoritmicamente, nossa estratégia da seguinte forma:

Algoritmo Busca-Local-Lag

(1) **início**

(2) Obter uma solução viável T

(3) Aplicar o algoritmo Busca-Local-Ciclo-2 utilizando $\mathbf{G^2(T)}$, obtendo T' (uma nova solução viável)

(4) Aplicar o algoritmo Busca-Local-Ciclo-2 utilizando $\mathbf{G^1(T')}$, obtendo T'' (uma nova solução viável)

(5) Procurar Transferências de Nós Válidas utilizando $\mathbf{G^T(T'')}$, obtendo T''' (a solução viável obtida no final do algoritmo)

(6) **fim**

7.1.3 Heurística Lagrangena Básica

Chamaremos de Heurística Lagrangeana Básica (HeurLagBasica), a Heurística Lagrangeana que, na terceira etapa, recebe como entrada, os custos Lagrangeanos. O custo Lagrangeano de uma aresta $[v_i, v_j] \in E$, é a soma dos custos Lagrangeanos dos arcos associados à $[v_i, v_j]$, ou seja, $\overline{d}_{ij} + \overline{d}_{ji}$. Como foi mencionado anteriormente, ela é acionado em toda iteração do nosso Algoritmo *Relax and Cut* para o PAGCM. Seu algoritmo pode ser assim descrito.

HeurLagBasica
Entrada: Grafo G
Saída: AGCM, limite inferior, limite superior
passo 1: aplicar a heurística EW para obtenção de uma solução viável para o PAGCM
passo 2: resolver o Subproblema Lagrangeano para os custos Lagrangeanos da iteração corrente
passo 3: se nenhuma das duas condições de parada for satisfeita, aplicar a heurística EW tendo como entrada os custos Lagrangeanos (obtendo então, uma nova solução viável, T). Submeter T a Busca-Local-Lag (repare que o passo (2) desse algoritmo é descartado, pois já temos a solução viável T) para os custos orginais.
passo 4: se nenhuma das duas condições de parada for satisfeita, atualizar os multiplicadores de Lagrange e voltar ao passo 2.
fim

7.1.4 Heurística Lagrangena com Custos Reduzidos

Quando, na terceira etapa da Heurística Lagrangeana, aplicarmos os custos reduzidos das variáveis de decisão, obteremos a Heurística Lagrangeana de Custos Reduzidos (HeurLagReduz). O custo reduzido de uma aresta

$[v_i, v_j] \in E$, é a soma dos custos reduzidos dos arcos associados à $[v_i, v_j]$, ou seja, $\zeta_{ij} + \zeta_{ji}$. Assimo como ocorre com a **HeurLagBasica**, ela é acionada em toda iteração do nosso Algoritmo *Relax and Cut* para o PAGCM. Seu algoritmo pode ser assim descrito.

HeurLagReduz
Entrada: Grafo G
Saída: AGCM, limite inferior, limite superior
passo 1: aplicar a heurística EW para obtenção de uma solução viável para o PAGCM
passo 2: resolver o Subproblema Lagrangeano para os custos Lagrangeanos da iteração corrente
passo 3: se nenhuma das duas condições de parada for satisfeita, aplicar a heurística EW tendo como entrada os custos Reduzidos (obtendo então, uma nova solução viável, T). Submeter T a Busca-Local-Lag (repare que o passo (2) desse algoritmo é descartado, pois já temos a solução viável T) para os custos orginais.
passo 4: se nenhuma das duas condições de parada for satisfeita, atualizar os multiplicadores de Lagrange e voltar ao passo 2.
fim

7.1.5 Heurística Lagrangena com Custos Complementares

Antes de definirmos nosso próximo tipo de Heurística Lagrangena, precisamos definir o que são custos complementares. Os custos complementares são os custos obtidos quando modificamos os custos originais da seguinte forma. Cada aresta do grafo original terá custo zero se na solução do Subproblema Lagrangeano aparecer algum arco relacionado à ela. Mais formalmente, seja $\overline{x}_{ij}$ a solução corrente do Subproblema Lagrangeano, os custos complementares $\{\overline{c}_{ij} : [v_i, v_j] \in E\}$ são obtidos da seguinte forma:

$$\overline{c}_{ij} = c_{ij}(1 - \overline{x}_{ij} - \overline{x}_{ji}), \quad \forall [v_i, v_j] \in E \tag{7.2}$$

A Heurística Lagrangeana de Custos Complementares (HeurLagComp) é obtida quando utilizamos os custos complementares como entrada para a heurística EW. Ela também é acionada em toda iteração do nosso Algoritmo *Relax and Cut* para o PAGCM. Segue seu algoritmo

HeurLagComp

Entrada: Grafo G

Saída: AGCM, limite inferior, limite superior

passo 1: aplicar a heurística EW para obtenção de uma solução viável para o PAGCM

passo 2: resolver o Subproblema Lagrangeano para os custos Lagrangeanos da iteração corrente

passo 3: se nenhuma das duas condições de parada for satisfeita, obter os custos complementares. Aplicar a heurística EW tendo como entrada os custos complementares (obtendo então, uma nova solução viável, T). Submeter T a Busca-Local-Lag (repare que o passo (2) desse algoritmo é descartado, pois já temos a solução viável T) para os custos orginais.

passo 4: se nenhuma das duas condições de parada for satisfeita, atualizar os multiplicadores de Lagrange e voltar ao passo 2.

fim

Capítulo 8

Resultados Computacionais

Neste capítulo, discutiremos o desempenho computacional da Heurística Lagrangeana com Custos Complementares. Como foi mencionado anteriormente, a HeurLagComp apresentou um desempenho computacional melhor (em termos de qualidade das soluções obtidas) do que o das outras Heurísticas Lagrangeanas. Devido a isso, analisaremos somente os resultados obtidos pela HeurLagComp. Confrontaremos aqui, os nossos resultados com os melhores resultados da literatura, tanto em termos de limites inferiores quanto de limites superiores.

8.1 Classes de Problemas Encontradas na Literatura

Na literatura, foram propostas algumas classes de problemas. Essas classes foram utilizadas como *benckmark* na maioria dos mais recentes algoritmos propostos para o PAGCM. As primeiras três classes que serão apresentadas abaixo, estão disponíveis eletronicamente na página http://www.ms.ic.ac.uk/info.html ([14]). A última classe, na realidade, é composta de algumas instâncias propostas originalmente para o Problema de Roteamento de Veículos (PRV) que foram adaptadas para o PAGCM (veja Hall [42]).

classe de problemas tc: grafos completos para $n = 41$, 81 e 121. A matriz de custos de cada problema foi gerada ao se tomar a parte inteira da distância euclidiana entre as coordenadas dos n pontos. Os pontos foram gerados aleatoriamente em um quadrado de dimensão 100×100. A raiz está localizada no centro desse quadrado. Para $n = 41$, existem 10 diferentes instâncias, onde Q (capacidade) pode assumir valor 5 ou 10. Para $n = 81$ existem 5 instâncias diferentes e Q pode assumir os seguintes valores: 5, 10 e 20. Todas as demandas são idênticas (unitárias). Existem, portanto, 35 instâncias.

classe de problemas te: possui as mesmas características da classe de problemas **tc**, exceto pelo fato de que a raiz, desta vez, está localizada em alguma extremidade do quadrado. Os problemas dessa classe são considerados mais difíceis do que os da classe **tc**. Ela possui a mesma quantidade de instâncias, ou seja, 35.

classe de problemas cm: grafos completos para $n = 50, 100$ e 200. Ao contrário das classes anteriores, os problemas dessa classe não são euclidianos. A matriz de custos e as demandas são geradas aletoriamente no intervalo $[0, 100]$. Claramente, os problemas dessa classe não satisfazem a desigualdade triangular. Para cada valor de n existem 5 diferentes instâncias onde a capacidade pode variar da seguinte forma: $Q = 200, 400$ e 800. Essa classe, possui, portanto, 45 instâncias.

classe de problemas propostas por Hall [42] (ch): essa classe é composta de alguns problemas que foram originalmente propostos para o PRV. Isso deve-se a grande semelhança entre o PRV e o PACGM (veja Araque *et al.* [11]). Os problemas **eil*** e **att48** foram selecionados por Reinelt [63] e (exceto o problema **att48**) apareceram originalmente no livro de Eilon *et al.* [22]. Esses problemas estão disponíveis via **ftp** (anônimo) em elib.zib-berlin.de no

diretório /pub/mp-testdata/tsp. As instâncias **vrpnc*** estão documentadas em [20] e são parte de uma biblioteca de problemas teste descritas em [14]. Elas também estão disponíveis na página http://www.ms.ic.ac.uk/info.html assim como os problemas **eil***. Todos esses problemas, exceto o problema **att48** (que possui demandas idênticas), possuem demandas heterogêneas. Além disso, todos são euclidianos. Para computar a matriz de custos foi seguida uma convenção da literatura do PRV, ou seja, a matriz de custos foi computada se tomando o maior inteiro não maior do que a distância euclidiana entre os nós. Essa classe possui 16 instâncias.

As classes de problemas **tc** e **te** foram inicialmente relatadas em Gouveia [35]. Elas foram utilizadas em muitos trabalhos como por exemplo, Amberg *et al.* [9], Hall [42], Gouveia e Martins [39, 40], Saraiha *et al.* [64], Patterson *et al.* [60], Ahuja *et al.* [8, 5] e Souza *et al.* [65]. A classe **cm** também foi utilizada em muitos trabalhos, entre eles podemos citar Saraiha *et al.* [64] e Ahuja *et al.* [8, 5]. A classe **ch** foi utilizada somente em Hall [42].

8.2 Apresentação dos Resultados

Visando uma melhor apresentação dos resultados, todos os programas foram executados em uma mesma máquina que possui a seguinte configuração: **Pentium III, 933 MHz**, com **512MB** de memória **RAM**. Além disso, todos os códigos foram implementados na linguagem de programação **C**, compilador **gcc**, versão **2.95**, para **Linux**.

Como foi mencionado, a Heurística Lagrageana com Custos Complementares (HeurLagComp) foi a que apresentou os melhores resultados em comparação com as outras Heurísticas Lagrangeanas. Devido a isso, toda análise de resultados se dará ao confrontarmos os resultados que obtivemos com a HeurLagComp com os melhores resultados da literatura.

Antes de apresentarmos os nossos resultados, vamos fazer uma revisão dos principais resultados da literatura tanto em termos de limites inferiores,

quanto de limites superiores obtidos para o PAGCM. Para todas as instâncias com demandas idênticas (classes **tc** e **te**) onde $n = 41$, a solução ótima é conhecida. O mesmo acontece em relação às instâncias onde $n = 81$ e $Q = 20$. Para todas essas instâncias, os limites superiores obtidos em Amberg *et al.* [9] e Ahuja *et al.* [8] foram provados como sendo soluções ótimas. Saraiha *et al.* [64] e Patterson *et al.* [60] também obtiveram alguns limites superiores que são soluções ótimas para algumas dessas instâncias. A prova de otimalidade da maioria desses limites superiores foi feita através da combinação do método baseado em planos-de-corte proposto por Hall [42] com um pacote programação inteira mista. Os casos remanescentes, ou seja, os casos onde $n = 41$, a raiz está localizada em alguma extremidade e $Q = 5$, a otimalidade foi provada através da combinação do pacote de programação inteira mista CPLEX, com o método baseado em planos-de-corte proposto por Gouveia e Martins [39].

Os melhores limites inferiores para instâncias com demandas idênticas foram obtidos ou por Gouveia e Martins [40] ou por Hall [42]. Para uma revisão dos melhores resultados da literatura para essas instâncias, veja [40].

Em relação à classe **cm**, muito foi feito em relação à obtenção de limites superiores. Os recentes trabalhos de Ahuja *et al.* [8, 5] igualaram ou melhoraram os melhores limites superiores da literatura. Alguns desses limites também foram obtidos em Saraiha *et al.* [64]. Até onde se sabe, o trabalho proposto por Frangioni *et al.* [27] foi o único que produziu limites inferiores para os problemas dessa classe. E, até onde se sabe também, a otimalidade de qualquer um dos problemas dessa classe *não* foi provada.

A classe de problemas **ch**, como foi mencionado, foi proposta e utilizada somente em Hall [42]. Porém, como ainda existem 6 instâncias onde a otimalidade não é conhecida, resolvemos incluir essa classe em nossos testes computacionais. As otimalidades das outras 10 instâncias foram obtidas, através da combinação do método baseado em planos-de-corte proposto em [42] com alguns pacotes de programação inteira mista. Os limites superiores

para as instâncias que não tiveram suas otimalidades provadas, foram obtidos inicialmente pela heurística proposta em [46] e possivelmente melhoradas ao longo de um algoritmo *Branch and Cut* também prosposto em [42].

As tabelas 8.1-8.5 que serão apresentadas nas próximas seções, conterão as seguintes informações: o nome da instância (Instância), o número de nós (N), a capacidade (Q), o limite inferior obtido pela HeurLagComp (LI), o limite superior obtido pela HeurLagComp (LS), o número de DGES's ativas ao final da HeurLagComp (N_DGES), o melhor limite inferior conhecido na literatura (M_LI), o melhor limite superior conhecido na literatura (M_LS), a solução ótima do problema (OPT), quando ela for conhecida, e o tempo de CPU gasto pela HeurLagComp (CPU), em segundos.

Além das tabelas mencionadas anteriormente, ainda existem mais três tabelas. A tabela 8.6 compara os tempos de CPU gastos pela HeurLagComp com os gastos pelos métodos em Ahuja *et al.* [8, 5]. A primeira coluna dessa tabela é o número de nós do problema (N), a segunda (Ahu) é o tempo gasto pelos métodos em [8, 5]. De [8], somente os tempos de CPU para as instâncias onde $n = 41$, foram extraídos, pois somente para essas instâncias o método mais recente ([5]) não foi testado. O restante dos tempos de CPU foram extraídos de [5]. A terceira coluna (HeurLagComp) representa o tempo de CPU gasto pela nossa heurística. Vale ressaltar que a máquina utilizada em [8] foi uma **DEC ALPHA**. E aquela utilizada em [5] tem configuração: **Pentium 4, 512 RAM, 256 L2 CACHE** (mais rápida do que aquela que utilizamos, porém é a que tem configuração mais parecida). Portanto, não dá para fazer uma comparação fiel dos tempos de CPU, porém, a título de ilustração, resolvemos fazê-la, sempre com a ressalva citada acima. Vale lembrar ainda que que nosso algoritmo produz tanto limites inferiores quanto limites superiores (portanto, é esperado um gasto considerável de tempo), enquanto os demais, produzem unicamente ou limites inferiores ou limites superiores. Evitamos comparar os nossos tempos de CPU com os tempos dos algoritmos que obtêm limites inferiores para o PAGCM, pois as máquinas utilizadas para a execução de tais algoritmos, eram bastante diferentes daquela que

aqui utilizamos.

As outras duas tabelas (8.7 e 8.8) contêm, basicamente, as mesmas informações: o nome da instância (Instância), o número de nós (N), a capacidade (Q), GAP (diferença, em percentagem, do limite inferior ao limite superior), o número de variáveis x_{ij} (binárias) inicial do problema (NVI), o número de variáveis x_{ij} fixadas em zero com teste de fixação do item 3.5.1 (NVFI), o número de variáveis x_{ij} fixadas em zero com teste baseado na análise dos custos reduzidos (item 3.5.2) (NVFCR), o número de variáveis restantes ao final do término da HeurLagComp (NVR), o número de variáveis de fluxo (z_{ij}^k), relacionadas às variáveis binárias não fixadas em zero, fixadas em zero (com o teste do item 3.5.2) (NVZF). Além dessas informações, a tabela 8.7 possui a coluna PVF que representa o percentual de variáveis binárias fixadas. E a tabela 8.8 possui mais duas colunas: OPT que representa o custo da solução ótima do problema e CPU que representa o tempo de CPU gasto pelo CPLEX [19] para resolver de forma exata os modelos multifluxos gerados a partir do conjunto de variáveis restantes após o final da HeurLagComp. Vale ressaltar que quando qualquer coluna, para qualquer uma das tabelas anteriormente citadas, apresentar os mesmos valores, eles não serão repetidos em cada linha da tabela, aparecerão apenas uma única vez.

Após analisarmos alguns experimentos computacionais que realizamos, e também para uma análise mais fiel dos resultados, estabelecemos a seguinte regra para execução da HeurLagComp. Para problemas onde $n < 50$ ou $n = 80$, utilizamos um número máximo de iterações de 5000 e a redução de α foi feita a cada 250 iterações sem melhoras do LI global (à exceção da instância **eil51** que foi tratada com esses parâmetros). Para a maioria dos problemas restantes ($n \geq 50$ e $n \neq 80$) utilizamos um número máximo de iterações de 10000 e a redução de α foi feita a cada 500 iterações sem melhoras do LI global. As excessões aqui são os problemas **vrpnc*** da classe **ch**. Para esses problemas, utilizamos um número máximo de iterações de 15000 e a redução de α foi feita a cada 750 iterações sem melhoras do LI global. Além

disso, utilizamos o mesmo β (= 0.03) em todas as execuções. Nas próximas seções, analisaremos os resultados que obtivemos para cada uma das classes de problemas mencionadas anteriormente.

8.3 Problemas das Classes TC e TE

Na tabela 8.1, apresentamos os resultados para os problemas da classe **tc** ($n = 41, 81$). Para todas as instâncias onde $n = 41$ e $Q = 5$, observamos que os limites superiores obtidos pela HeurLagComp são todos soluções ótimas e que os limites inferiores não se mostraram tão competitivos quanto os melhores limites da literatura. Em relação às instâncias onde $n = 41$ e $Q = 10$, os limites superiores que obtivemos são, também, todos soluções ótimas. Os limites inferiores, nesse caso, se mostraram mais competitivos, tanto que a própria HeurLagComp, através do teste de otimalidade $Z_{ub} - Z_{lb} < 1$, conseguiu provar a otimalidade de 6 das 10 instâncias. À excessão da instância **tc40-10**, os *gaps* das outras instâncias foram muitos pequenos.

Ainda em relação às instâncias onde $n = 41$, observamos que o N_DGES é consideralmente maior, na maioria dos casos, quando $Q = 5$. Isso era esperado, uma vez que quanto mais apertada é a capacidade, maior é a probabilidade de se encontrar violações de DGES's. É interessante observar ainda que para todas as instâncias onde HeurLagComp provou a otimalidade, o tempo de CPU gasto foi consideravelmente menor quando comparado com os tempos de outras instâncias onde a otimalidade não foi provada. Isso se explica pelo fato de que a prova de otimalidade, nesses casos, aconteceu antes das 5000 iterações previstas da HeurLagComp. Todas essas instâncias são consideradas fáceis, principalmente pelo fato da raiz está localizada no centro do quadrado onde as coordenadas dos nós foram geradas. Além disso, o número de nós é razoalvemente pequeno.

Para $n = 81$, ainda na tabela 8.1, observamos o seguinte. Igualamos os melhores limites superiores em 8, das 15 instâncias consideradas. Em

somente uma (**tc80-1**, $Q = 10$), das quatro instâncias que não tiveram ainda suas otimalidades provadas, não obtivemos a melhor solução viável da literatura. Porém, com outros parâmetros (número máximo de iterações permitido e a forma como reduzimos o valor de α ao longo das iterações), obtivemos o mesmo valor da literatura, ou seja, 888. Nossos limites inferiores se mostraram mais competitivos quando $Q = 20$. Nesse caso, a própria HeurLagComp provou a otimalidade de duas instâncias (**tc80-3** e **tc80-4**) através do teste $Z_{ub} - Z_{lb} < 1$. Assim como ocorreu com as instâncias onde $n = 41$, o N_DGES é maior quanto mais apertada é a capacidade. A média do N_DGES quando $Q = 5$ é 66, 41 para $Q = 10$ e 20 para $Q = 20$.

Em relação aos problemas da classe **te**, podemos observar o seguinte pela tabela 8.2. Para $n = 41$ e $Q = 5$, os nossos limites superiores são todos soluções ótimas. Já os nossos limites inferiores assim como ocorreu na classe **tc** ($n = 41$ e $Q = 5$) não se mostraram muito competitivos em relação aos melhores da literatura. O mesmo aconteceu quando ($n = 41$ e $Q = 10$), nossos limites superiores são soluções ótimas e os limites inferiores pouco competitivos, porém, com *gaps* menores (em relação às mesmas instâncias, porém para $Q = 5$).

Pela tabela 8.2 e ainda para $n = 41$, podemos observar que o N_DGES é maior quando $Q = 5$. Além disso, tivemos até uma surpresa, a instância **te40-1**, para $Q = 10$, teve o N_DGES maior do que quando testada para $Q = 5$. Nos demais casos, o N_DGES para $Q = 5$ foi maior do que quando $Q = 10$, como era esperado.

Em relação às instâncias onde $n = 81$, pela tabela 8.2, podemos observar que obtemos os melhores limites superiores da literatura em 11, das 15 instâncias consideradas. Nossos limites superiores não se mostratam tão competitivos, assim como ocorreu para os problemas dessa classe onde $n = 41$. O N_DGES para essas instâncias tem um comportamento semelhante ao das instâncias anteriores. Ele é maior, quanto mais apertada é a capacidade. Na média, o N_DGES quando $Q = 5$ é 70, quando $Q = 10$ é 57 e 40 quando

$Q = 20$. Porém, note que essa média é maior para essa classe do que para a classe **tc** para o mesmo valor de $n(81)$. O mesmo aconteceu em relação às instâncias de 41 nós.

O tempo médio de CPU para todas as instâncias (classes **tc** e **te**) onde $n = 41$ foi de 156s, um pouco maior do que o gasto em (Ahu) (veja a tabela 8.6). Já em relação às instâncias de 81 nós, o tempo médio de CPU gasto pela HeurLagComp foi menor ($1185s$) do que o gasto pelos métodos em (Ahu)(1800s).

8.4 Problemas da Classe CM

A tabela 8.3 apresenta os resultados obtidos para os problemas da classe **cm**. Vale lembrar aqui que a otimalidade de qualquer um desses problemas é desconhecida. Os valores que aparecem em negrito são os LI's que a HeurLagComp conseguiu melhorar. Já os valores que aparecem sublinhados, são os valores (M_LI e/ou M_LS) que são melhores que os nossos.

Pela tabela 8.3, para $n = 50$, observamos que só não obtivemos os melhores limites superiores da literatura em três instâncias. Vale aqui mencionar que o limite superior 564 da instância **cm50r4**, $Q = 400$, foi por nós obtido, porém, o número máximo de iterações permitido usado, assim como a forma como α foi reduzido ao longo das iterações, era diferente daquele utilizado na computação desses resultados. Porém, os limites inferiores que obtivemos para esses problemas são consideravelmente melhores do que os anteriormente conhecidos. Observe que essas melhorias são mais significativas quanto mais apertada é a capacidade (maior quando $Q = 200$, depois quando $Q = 400$ e, por fim, quando $Q = 800$). Observe ainda que N_DGES é maior também quanto mais apertada for a capacidade. A média do N_DGES quando $Q = 200$ é 65, 42 quando $Q = 400$ e 27 quando $Q = 800$.

Ainda pela tabela 8.3, porém em relação às instâncias onde $n = 100$, observamos que somente obtivemos os melhores limites superiores da literatura

para 4 instâncias. Vale lembrar aqui que da mesma forma que obtivemos o limite superior 564 para a instância **cm50r4**, $Q = 400$, obtivemos o limite superior 186 para a instância **cm100r5**, $Q = 800$, porém, com um número máximo de iterações permitido, assim como a forma como o α foi reduzido, diferentes dos estabelecidos para a computação desses testes. Porém, conseguimos melhorar 12 dos 15 limites inferiores para essas instâncias (perdemos em duas e empatamos em outra). Assim como ocorre para $n = 50$, verificamos que as melhorias mais significativas ocorreram nas instâncias onde a capacidade é mais apertada. Observe ainda que N_DGES é maior também quanto mais apertada for a capacidade. A média do N_DGES quando $Q = 200$ é 74, 37 quando $Q = 400$ e 26 quando $Q = 800$.

A tabela 8.4 mostra os resultados das instâncias onde $n = 200$. Observamos nessa tabela que o desempenho da HeurLagComp, tanto em termos de limites inferiores quanto em termos de limites superiores, foi pior do que quando $n = 50, 100$. Não conseguimos obter nenhum dos melhores limites superiores conhecidos e melhoramos 8 dos melhores limites inferiores conhecidos (perdemos em 7). Em geral, o comportamento da HeurLagComp é bem parecido com o comportamento da mesma quando está tratando as instâncias anteriores. As melhorias mais significativas ocorreram quanto mais apertada é a capacidade. Da mesma forma, o N_DGES é maior também quanto mais apertada for a capacidade. A média do N_DGES quando $Q = 200$ é 144, 63 quando $Q = 400$ e quando $Q = 800$ é 17. O que é bastante curioso aqui é o pequeno número de DGES's ativas na última iteração quando $Q = 800$. Talvez, por isso, não tenhamos conseguido melhorar os limites inferiores dessas instâncias. E, possivelmente, ocorreram outros tipos de violações nas subarborescências (da solução do Subproblema Lagrangeano) que não violaram as DGES's.

Pela tabela 8.6 verificamos que os tempos médios de CPU gastos pela HeurLagComp crescem bastante a medida que o tamanho da instância aumenta. O tempo médio de CPU gasto em (Ahu) foi bem menor do que o gasto pela nossa heurística, para $n = 100, 200$. Porém, para $n = 50$, nossa

heurística teve um tempo médio de CPU menor.

A tabela 8.7 mostra o desempenho dos testes de fixação de variáveis da seção 3.5. Observe que o percentual do número total de variáveis binárias fixadas foi superior a 90%. Em relação à instância de 50 nós, observe que o número de variáveis (binárias) que foram fixadas em zero com o teste de redução inicial (NVFI) foi maior do que o de fixadas com os custos reduzidos (NVFCR). O contrário ocorreu em relação às instâncias com 100 nós. O número de variáveis fixadas através da análise dos custos reduzidos foi maior do que aquele fixado com teste de redução inicial. Além disso, verificamos que o número de variáveis de fluxo fixadas em zero não foi tão considerável em todas as instâncias dessa tabela.

Ainda pela tabela 8.7 observamos que apesar de mais de 90% das variáveis de decisão terem sido fixadas em zero, o número de variáveis de decisão restantes (para $n = 100$) ainda é considerável. Porém, em relação à instância **cm50r4**, observe que sobraram poucas variáveis de decisão. Devido a isso, conseguimos provar a otimalidade (até então desconhecida) dessa instância, como será melhor detalhado mais adiante.

8.5 Problemas da Classe CH

A tabela 8.5 apresenta os resultados obtidos para os problemas da classe **ch**. Entre as instâncias que possuem suas otimalidades conhecidas, a própria HeurLagComp conseguiu provar as otimalidades de duas (**eil22** e **eil30**).

Ainda pela tabela 8.5 observamos que, em geral, os nossos limites inferiores foram competitivos (não melhores, para as instâncias **eil***) com os obtidos em [42] (melhores da literatura). Enquanto que os limites inferiores que obtivemos para as instâncias **vrpnc*** não se mostraram tão competitivos. Em relação aos limites superiores, a HeurLagComp conseguiu melhorar os limites superiores de 5 das 6 instâncias que possuem otimalidades desconhecidas (**eila76**, **eilb76**, **eila76**, **vrpnc5**, **vrpnc11**). E mais, obtivemos as

soluções ótimas de todas as instâncias que possuem otimalidades conhecidas.

Porém, aconteceram algumas surpresas. Conseguimos obter, para 2 instâncias, soluções viáveis de custos menores do que os custos das supostas soluções ótimas. Isso aconteceu com as instâncias **eil33**, **att48** (os nomes delas estão sublinados na tabela). Obtivemos para a instância **eil33**, um limite superior de custo 494, quando o suposto ótimo é 503. Para a instância **att48**, obtivemos um limite superior de 28935 quando suposto ótimo tem custo 29154.

Em relação ao N_DGES, o comportamento da HeurLagComp é parecido com o que a mesma teve quando tratou as instâncias anteriores (com demandas heterogêneas). O N_DGES é maior quanto mais apertada é a capacidade. Como o número de nós e a capacidade são sempre diferentes, fica difícil fazer uma análise mais detalhada.

Em relação aos tempo de CPU, vale destacar que quando a HeurLagComp provou a otimalidade, os tempos foram bem pequenos. Novamente, isso deve-se ao fato que a prova de otimalidade ocorreu bem antes do número de iterações previsto.

8.6 Provas de Otimalidade

Para ilustrar como é possível provar a otimalidade através do método proposto nesse estudo, a tabela 8.8 exemplifica as provas de otimalidade de 4 instâncias (com *gaps* de dualidade pequenos), entre elas, existe uma (**cm50r4**, $Q = 800$) que, até então, não possuía otimalidade conhecida.

A idéia é gerar um Problema de Programação Linear Inteira (PPLI) a partir das variáveis (binárias e de fluxo) restantes ao final da HeurLagComp. Ou seja, ao término da execução de nossa heurística, nós geramos o modelo multifluxos (capítulo 2) para o problema correspondente a partir das variáveis restantes e ainda acrescentamos, a esse modelo, as DGES's ativas ao final do

nosso algoritmo.

Esse tipo de prova depende fundamentalmente dos testes de fixação de variáveis. Para essas instâncias, como foram fixadas muitas variáveis de decisão, foi possível gerar um PPLI de tamanho razoável, o que permitiu a resolução exata desses problemas.

Para resolver os PPLI's gerados, utilizamos o pacote CPLEX 7.0[19] com seus parâmetros *default*. Pela tabela 8.8 verificamos que o tempo de CPU gasto para resolver os PPLI's gerados é bem pequeno. Isso deve-se, mais uma vez, ao pequeno número de variáveis binárias restantes. Apesar de um número razoável de variáveis de fluxo (NVZF), associadas às variáveis binárias não fixadas em zero, terem sido fixadas em zero também.

O resultado mais interessante desses testes foi provar a otimalidade (até então desconhecida) da instância **cm50r4**. Tentamos ainda provar a otimalidade de outras instâncias (**cm100***, $Q = 800$), porém, apesar dessas instâncias possuírem *gaps* de dualidade pequenos, o número de variáveis restantes (NVR) ainda é bastante considerável e isso impossibilitou a resolução dos PPLI's.

Instância	N	Q	LI	LS	N_DGES	M_LI	M_LS	OPT	CPU
	41	5							
tc40-1			576.9	586	38	582	586	586	135
tc40-2			567.6	578	44	574	578	578	136
tc40-3			565.4	577	30	574	577	577	140
tc40-4			603.7	617	33	613	617	617	144
tc40-5			586.3	600	37	593	600	600	143
tc40-6			584.8	590	40	590	590	590	142
tc40-7			602.9	609	35	608	609	609	137
tc40-8			548.6	553	42	552	553	553	130
tc40-9			594.3	599	31	598	599	599	135
tc40-10			591.9	600	49	593	600	600	144
	41	10							
tc40-1			497.1	498	14	498	498	498	85
tc40-2			488.6	490	28	490	490	490	147
tc40-3			499.3	500	18	500	500	500	65
tc40-4			511.3	512	15	512	512	512	100
tc40-5			503.1	504	30	504	504	504	148
tc40-6			497.1	498	18	498	498	498	74
tc40-7			507.1	508	20	508	508	508	76
tc40-8			483.3	485	26	484	485	485	153
tc40-9			513.3	516	24	514	516	516	152
tc40-10			508.9	517	21	516	517	517	159
	81	5							
tc80-1			1058.7	1099	58	1097	1099	1099	894
tc80-2			1065.0	1102	75	1095	1100	1100	899
tc80-3			1033.7	1073	61	1069	1073	1073	866
tc80-4			1028.3	1084	71	1073	1080	1080	865
tc80-5			1230.6	1288	76	1278	1287	1287	862
	81	10							
tc80-1			872.1	890	48	878	888	-	1053
tc80-2			870.4	877	44	876	877	-	1005
tc80-3			864.8	878	25	870	878	-	1001
tc80-4			857.3	868	45	864	868	-	1033
tc80-5			994.0	1012	44	999	1002	1002	1045
	81	20							
tc80-1			830.9	834	20	834	834	834	1208
tc80-2			815.7	824	24	820	820	820	1281
tc80-3			827.1	828	11	828	828	828	785
tc80-4			819.1	820	17	820	820	820	889
tc80-5			913.8	922	29	916	916	916	1260

Tabela 8.1: Problemas da classe **tc** para $n = 41, 81$

Instância	N	Q	LI	LS	N_DGES	M_LI	M_LS	OPT	CPU
	41	5							
te40-1			780.9	830	33	830	830	830	176
te40-2			756.8	792	41	790	792	792	169
te40-3			752.5	797	46	792	797	797	174
te40-4			779.7	814	36	812	814	814	166
te40-5			758.8	784	36	780	784	784	169
te40-6			776.0	818	51	817	818	818	172
te40-7			788.0	820	37	820	820	820	170
te40-8			793.0	827	34	822	827	827	159
te40-9			746.2	779	41	778	779	779	176
	41	10							
te40-1			579.8	596	38	590	596	596	206
te40-2			566.0	573	34	571	573	573	202
te40-3			547.6	568	30	556	568	568	228
te40-4			578.7	596	28	585	596	596	182
te40-5			563.9	572	25	565	572	572	203
te40-6			564.8	576	36	573	576	576	224
te40-7			576.7	591	32	586	591	591	198
te40-8			588.9	610	30	599	610	610	197
te40-9			551.0	562	30	558	562	562	205
	81	5							
te80-1			2343.2	2544	69	2536	2544	2544	1086
te80-2			2349.8	2560	77	2533	2551	-	1042
te80-3			2413.9	2615	67	2596	2612	-	1027
te80-4			2342.6	2564	70	2537	2558	-	1068
te80-5			2301.7	2469	66	2463	2469	2469	1053
	81	10							
te80-1			1541.0	1657	59	1641	1657	-	1354
te80-2			1525.5	1639	60	1607	1639	-	1320
te80-3			1573.1	1687	54	1664	1687	-	1373
te80-4			1555.9	1635	56	1624	1629	1629	1392
te80-5			1516.4	1603	55	1592	1603	-	1332
	81	20							
te80-1			1252.2	1275	27	1256	1275	-	1813
te80-2			1197.6	1224	47	1207	1224	-	2053
te80-3			1242.4	1267	46	1257	1267	-	1611
te80-4			1244.0	1265	42	1249	1265	-	1508
te80-5			1227.0	1240	38	1231	1240	-	1584

Tabela 8.2: Problemas da classe te para $n = 41, 81$

Instância	N	Q	LI	LS	N_DGES	M_LI	M_LS	OPT	CPU
	50	200							
cm50r1			**1039.4**	1098	70	998.6	1098	-	562
cm50r2			**913.1**	981	66	876.7	974	-	560
cm50r3			**1131.1**	1194	66	1079.8	1186	-	590
cm50r4			**768.9**	800	64	709.0	800	-	509
cm50r5			**867.7**	928	59	829.1	928	-	563
	50	400							
cm50r1			**645.4**	681	43	608.6	681	-	606
cm50r2			**608.8**	631	45	591.5	631	-	628
cm50r3			**704.3**	732	44	658.5	732	-	671
cm50r4			**543.5**	567	39	509.3	564	-	549
cm50r5			**586.5**	611	41	557.0	611	-	609
	50	800							
cm50r1			**482.6**	495	29	450.7	495	-	774
cm50r2			**502.4**	513	30	488.3	513	-	739
cm50r3			**522.3**	532	28	483.1	532	-	810
cm50r4			**468.9**	471	23	458.4	471	-	604
cm50r5			**477.4**	492	26	462.9	492	-	759
	100	200							
cm100r1			**452.8**	553	80	441.7	516	-	3115
cm100r2			**544.2**	638	67	525.3	596	-	3031
cm100r3			**478.0**	582	63	469.4	541	-	3439
cm100r4			**379.4**	459	82	372.3	437	-	3111
cm100r5			**373.3**	465	77	356.8	425	-	3470
	100	400							
cm100r1			**244.4**	262	49	232.9	252	-	3469
cm100r2			254.1	283	25	256.6	278	-	3705
cm100r3			216.0	243	34	219.0	236	-	3745
cm100r4			**205.6**	223	41	204.6	219	-	4384
cm100r5			**212.6**	228	38	211.5	223	-	3605
	100	800							
cm100r1			**176.9**	182	42	176.0	182	-	4951
cm100r2			**176.8**	179	22	169.8	179	-	4073
cm100r3			171.2	175	27	171.2	175	-	4074
cm100r4			**180.3**	183	16	179.4	183	-	4813
cm100r5			**182.9**	187	24	181.7	186	-	4754

Tabela 8.3: Problemas da classe **cm** para $n = 50, 100$

Instância	N	Q	LI	LS	N_DGES	M_LI	M_LS	OPT	CPU
	200	200							
cm200r1			**883.5**	1117	137	878.0	1017	-	27727
cm200r2			**1082.9**	1378	129	1071.6	1221	-	28787
cm200r3			**1199.6**	1487	141	1186.9	1365	-	27068
cm200r4			**831.0**	1041	158	821.8	927	-	26674
cm200r5			**845.3**	1083	153	835.0	965	-	28589
	200	400							
cm200r1			350.0	424	65	364.7	397	-	29669
cm200r2			**451.7**	510	60	451.0	478	-	28502
cm200r3			**530.3**	590	71	528.4	560	-	28182
cm200r4			**368.7**	413	50	368.5	392	-	27845
cm200r5			390.1	445	67	395.6	420	-	27769
	200	800							
cm200r1			245.0	257	16	248.0	254	-	32507
cm200r2			281.8	296	18	282.6	294	-	31624
cm200r3			347.0	362	17	347.5	361	-	31647
cm200r4			265.7	278	18	268.8	275	-	31282
cm200r5			281.0	297	15	282.6	292	-	33467

Tabela 8.4: Problemas da classe **cm** para $n = 200$

Instância	N	Q	LI	LS	N_DGES	M_LI	M_LS	OPT	CPU
eil22	22	6000	245.1	246	18	246	246	246	9
eil23		4500	391.3	395	19	392.5	395	395	41
eil30		4500	311.2	312	11	312	312	312	22
eil33		8000	**484.9**	**494**	31	496.1	503	503	110
eil51		160	374.1	380	30	376.0	380	380	290
	76								
eila76		100	510.3	**526**	56	514.1	531	-	1583
eilb76		140	563.4	**594**	62	568.7	601	-	1532
eilc76		180	488.3	497	49	490.7	497	497	1743
eild76		220	475.7	482	41	481.1	482	482	1737
	101								
eila101		200	566.0	570	52	568.9	570	570	3990
eilb101		112	635.7	**662**	72	639.9	675	-	3863
att48	48	15	**28715.7**	**28935**	15	28604	29154	29154	261
vrpnc4	151	200	637.4	**660**	85	657.3	677	-	14870
vrpnc5	200	200	747.0	**797**	77	768.2	807	-	33266
vrpnc11	121	200	569.3	**613**	54	583.2	614	-	9952
vrpnc12	101	200	493.7	512	66	510.0	512	512	3582

Tabela 8.5: Problemas da classe **ch**

N	Ahu	HeurLagComp
40	100	156
50	1000	636
80	1800	1185
100	1800	3849
200	3600	29405

Tabela 8.6: Tempos de CPU (em segundos) Ahuja *et al.* [8, 5] (Ahu) × HeurLagComp

Instância	N	Q	GAP (%)	NVI	NVFI	NVFCR	NVR	NVZF	PVF (%)
	50	800		2401					
cm50r4			0.004		1224	980	197	205	91.8
	100	800		9801					
cm100r1			0.028		3382	5568	851	242	91.3
cm100r2			0.012		2740	6544	517	105	94.7
cm100r3			0.022		3324	5762	715	264	92.7
cm100r4			0.014		3516	5722	563	135	94.3
cm100r5			0.022		4000	5026	775	244	92.1

Tabela 8.7: Desempenho dos testes de fixação de variáveis

Instância	N	Q	GAP (%)	NVI	NVFI	NVFCR	NVR	NVZF	OPT	CPU
	40	10		1600						
tc40-2			0.003		970	478	197	140	490	73.8
tc40-8			0.004		513	980	117	195	485	87.2
eil23	23	4500	0.010	484	256	142	86	38	395	9.46
cm50r4	50	800	0.004	2401	1224	980	197	205	**471**	179.7

Tabela 8.8: Provas de Otimalidade

Capítulo 9

Conclusões e Sugestões para Trabalhos Futuros

Neste trabalho, introduzimos uma Heurística Lagrangeana para o PAGCM. Ela é utilizada no contexto de um algoritmo de Otimização por Subgradientes onde os multiplicadores de Lagrange estão disponíveis, assim como a solução ótima do Subproblema Lagrangeano associado à esses multiplicadores. No caso particular da Heurística Lagrangeana proposta nesse estudo (HeurLagComp), ela foi utilizada num contexto de algoritmo *Relax and Cut* (um esquema dinâmico onde desigualdades (DGES) que violam o Subproblema Lagrangeano corrente são dualizadas e mantidas assim enquanto *ativas*). A idéia de utilizar a HeurLagComp nesse contexto deve-se ao fato de que os limites inferiores obtidos nesse esquema são melhores do que os obtidos no contexto habitual de um algoritmo de Otimização por Subgradientes.

Além disso, foram utilizadas (como parte de nossa Heurística Lagrangeana), algumas das mais recentes e eficientes estratégias de Busca Local da literatura, veja [8, 5].

Em geral, nossos resultados em termos de limites superiores foram dominados somente pelo trabalho recente de Ahuja *et al.* [5]. Porém, em relação aos limites inferiores conhecidos para uma determinada classe de problemas (**cm**), conseguimos melhorar sensivelmente quase todos.

Verificamos que a qualidade do limite inferior obtido influencia diretamente na qualidade do limite superior obtido. Em geral, quanto melhor for o limite inferior, melhor será o limite superior. Isso se explica pelo fato de que a Heurística Lagrangeana proposta nesse estudo é guiada, inicialmente, pelas soluções dos Subproblemas Lagrangeanos resolvidos aos longo das iterações. Portanto, quanto mais a solução do Subproblema Lagrangeano se aproximar de uma solução viável para o problema (geralmente acontece quando o *gap* de dualidade é pequeno), melhor será guiada a Heurística Lagrangeana na obtenção de limites superiores.

De forma geral, para as classes **tc** e **te** obtivemos as soluções ótimas de quase todas as instâncias onde as otimalidades eram conhecidas e para aquelas onde isso não acontecia, obtivemos a maioria dos melhores limites superiores da literatura.

Em relação à classe de problemas **cm**, verificamos que o desempenho de nossa Heurística foi melhor na obtenção de limites inferiores do que na obtenção de limites superiores. E que, o desempenho na obtenção de limites superiores, foi melhor para instâncias de 50 nós.

Os resultados para os problemas da classe **ch** foram os mais expressivos de nossa Heurística Lagrangeana. Melhoramos todos os limites superiores das instâncias que possuem otimalidades desconhecidas (6) e ainda encontramos soluções viáveis de custos menores do que as supostas soluções ótimas em duas instâncias.

E, por fim, um resultado bastante interessante de nosso trabalho foi provar a otimalidade (desconhecida até então) de uma das instâncias da classe **cm** (**cm50r4**, $Q = 800$). Enfim, a Heurística Lagrangeana proposta neste trabalho se mostrou uma ferramenta interessante para tratar o PAGCM, principalmente, quando o *gap* de dualidade é pequeno. Muito provavelmente, uma melhoria na qualidade dos limites inferiores pode levar, além de melhorias na qualidade do limite superior, a prova de otimalidade de outras instâncias, principalmente, da classe **cm**.

9.1 Trabalhos Futuros

Um sugestão para trabalhos futuros imediata, a partir das idéias aqui apresentadas e também pelos resultados obtidos, é melhorar a qualidade dos limites inferiores. Pois, como foi mencionado, uma melhoria dos limites inferiores provavelmente melhorará a qualidade dos limites superiores além de possibilitar a prova de otimalidade de outras instâncias, principalmente das instâncias **cm100***, $Q = 800$.

Para aumentar os nossos limites inferiores acreditamos que a inclusão das desigualdades *estrelas* (veja Hall [42]) possa ter um impacto bastante possitivo nesse sentido. Além disso, podemos procurar novas desigualdades para o PAGCM analisando a solução da relaxação linear (da formulação multifluxos) de instâncias pequenas.

Uma outra sugestão interessante é tratarmos do problema quando o número de subárvores é fixo (veja [69]). Ou seja, o número de arcos saindo da raiz é fixo em M (o que leva à restrição $\sum_{(v_1,v_j)\in A} x_{1j} = M$). Outra desigualdade que pode ser usada é que agora um limite inferior para o fluxo que sai por qualquer arco da raiz, digamos (v_1, v_j), tem que ser maior ou igual ao somatório das demandas de todos os nós - $(M-1) * Q$, ou seja, $\sum_{k\in\bar{I}} z_{1j}^k q_k \geq [\sum_{k\in\bar{I}} q_k - (M-1) * Q]x_{1j}$, $\forall (v_1, v_j) \in A$. Essas desigualdades devem tornar a formulação que aqui utilizamos, mais forte para resolver o PAGCM quando o número de subárvores é fixo.

Bibliografia

[1] Aarts, E., Korst, J. *Simulated Annealing and Boltzman Machines.* Wiley, Chichester, 1989.

[2] Aarts, E., Lenstra, J. K. *Local Search in Combinatorial Optimization.* Wiley, New York, USA, 1997.

[3] Ahuja, R., Boland, N., Dumitrescu, I. "Exact and Heuristic Algorithms for the Subset Disjoint Minimum Cost Cycle Problem", 2001. Working Paper - http://www.ise.ufl.edu/ahuja/vlsn/Papers/Pap-ABD.pdf.

[4] Ahuja, R. K., Magnanti, T.L., Orlin, J.B. *Network Flows: Theory, Algorithms and Applications.* Prentice Hall, New Jersey, USA, 1993.

[5] Ahuja, R.K., Ergun, O., Orlin, J.B., Punnen, A.P. "A Composite Very Large-Scale Neighborhood Structure for the Capacitated Minimum Spanning Tree Problem", 2001. Submited to Operations Research Letters.

[6] Ahuja, R.K., Ergun, O., Orlin, J.B., Punnen, A.P. "A Survey of Very Large-Scale Neighborhood Search Techniques", 2002. To appear in Discrete Applied Mathematics.

[7] Ahuja, R.K., Orlin, J.B., Sharma, D. "Very Large-Scale Neighborhood Search". *International Transactions in Operations Research*, v. 7, pp. 301–317, 2000.

[8] Ahuja, R.K., Orlin, J.B., Sharma, D. "Multi-Exchange Neighborhood Search Algorithms for the Capacitated Minimum Spanning Tree Problem". *Mathematical Programming*, v. 91, pp. 71–97, 2001.

[9] Amberg, A., Domschke, W., Voß, S. "Capacitated Minimum Spanning Trees: Algorithms Using Intelligent Search". *Combinatorial Optimization: Theory and Pratice*, v. 1, pp. 9–40, 1996.

[10] Andrade, R. C. *Heurísticas Lagrangeanas para o Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau nos Vértices.* Tese de mestrado, Programa de Engenharia de Sistemas e Computação - UFRJ, Rio de Janeiro, Brasil, 1999.

[11] Araque, J. R., Hall, L., Magnanti, T.L. *Capacitated Trees, Capacitated Routing, and Associated Polyhedra.* Technical Report SOR-90-12, Program in Statistics and Operations Research, Princeton University, Princeton, NJ, 1990.

[12] Beasley, J. *Lagrangean Relaxation.* Personal Notes, Imperial College, London, England, 1992.

[13] Beasley, J. E. "An Algorithm for the Steiner Problem in Graphs". *Networks*, v. 14, pp. 147–159, 1984.

[14] Beasley, J. E. "Or-Library: Distribuiting Test Problems by Eletronic Mail". *Journal of Operational Research Society*, v. 41, pp. 1069–1072, 1990.

[15] Belloni, A., Lucena, A. *Lagrangian Heuristics to Linear Ordering.* Relatório Técnico, Laboratório de Métodos Quantitativos, Departamento de Administração, Universidade Federal do Rio de Janeiro, 2001.

[16] Calheiros, F., Lucena, A., Souza, C. *Optimal Rectangular Partitions.* Relatório Técnico, Laboratório de Métodos Quantitativos, Departamento de Administração, Universidade Federal do Rio de Janeiro, 2001.

[17] Chandy, K., Lo, T. "The Capacitated Minimum Spanning Tree Problem". *Networks*, v. 3, pp. 173–182, 1973.

[18] Clarke, G., Wright, J.W. "Scheduling of Vehicles from a Central Depot to a Number Delivery Points". *Operations Research*, v. 12, pp. 568–581, 1964.

[19] CPLEX 7.1 ILOG. *Inc. CPLEX Division*, 2001.

[20] Cristofides, N., Mingozzi, A., Toth, P., Sandi, C. *Combinatorial Optimization.* Wiley & Sons, New York, 1979.

[21] Edmonds, J. "optimum branchings". *Journal of Research of The National Bureau of Standars*, v. 71B, pp. 233–240, 1967.

[22] Elion, S., Watson-Gandy, C.D.T, Cristofides, N. *Distribuition Management: Mathematical Modelling and Practical Analysis.* Hafner, New York, USA, 1994.

[23] Esau, L. R., Williams, K. C. "On Teleprocessing System Design. Part II-A Method for Approximating the Optimal Network". *IBM Systems Journal*, v. 5, pp. 142–147, 1966.

[24] Feo, T. A., C., M. G. "Greedy Randomized Adaptive Search Procedures". *Journal of Global Optmization*, v. 6, pp. 109–125, 1995.

[25] Fischetti, M., Toth, Paolo. "An Efficient Algorithm for the Min-Sum Arborescence Problem on Complete Digraphs". *ORSA Jornal on Computing*, v. 5, pp. 426–434, 1993.

[26] Fisher, M. L. "The Lagrangean Relaxation Method for Solving Integer Programming Problems". *Management Science*, v. 27, pp. 1–18, 1981.

[27] Frangioni, A., Pretolani, D., Scutella, M. G. *Fast Lower Bounds for the Capacitated Minimum Spanning Tree Problem.* Technical Report TR-99-05, Dipartimento di Informatica, Università di Pisa, 1999.

[28] Gavish, B. "Topological Design of Centralized Computer Networks: Formulations and Algorithms". *Networks*, v. 12, pp. 355–377, 1982.

[29] Gavish, B. "Formulations and Algorithms for the Capacitated Minimum Spanning Tree Problem". *JACM*, v. 30, pp. 118–132, 1983.

[30] Gavish, B. "Augmented Lagragean Basead Algorithms for Centralized Network Design". *IEEE Trans. Commun. Com.*, v. 33, pp. 1247–1257, 1985.

[31] Gavish, B. "Topological Design Telecommunications Networks - Local Access Design Methods". *Annals of Operations Research*, v. 33, pp. 17–71, 1991.

[32] Gavish, B., Altinkemer, K. "A Parallel Savings Heuristic for the Topological Design of Local Access Networks". In *Proceedings of IEEE INFOCOM*, 1986.

[33] Glover, F. "Ejection Chains, Reference Structures and Altering Path Methods for Traveling Salesman Problems". *Discrete Applied Mathematics*, v. 65, pp. 223–253, 1996.

[34] Glover, F., Laguna, M. *Tabu Search.* Kluwer Academic Publishers, Norwell, MA, 1997.

[35] Gouveia, L. "A 2n-Constraint Formulation for the Capacitated Minimal Spanning Tree Problem". *Operations Research*, v. 43, pp. 130–141, 1995.

[36] Gouveia, L. "Multicommodity Flow Models for Spanning Tree with Hop Constraints". *European Journal of Operational Research*, v. 95, pp. 178–190, 1996.

[37] Gouveia, L. *Using Variable Redefinition for Computing Minimum Spanning and Steiner Trees with Hop Constraints.* Working paper 2-96, Centro de Investigação Operacional, Faculdade de Ciências, Universidade de Lisboa, 1996.

[38] Gouveia, L., Lopes, M. J. "Valid Inequalities for Non-Unit Demand Capacitated Spanning Tree Problems with Flow Costs". *European Journal of Operational Research*, v. 121, pp. 394–411, 2000.

[39] Gouveia, L., Martins, P. "The Capacitated Minimal Spanning Tree Problem: An Experiment with a Hop-Index Model". *Annals of Operations Research*, v. 86, pp. 271–294, 1999.

[40] Gouveia, L., Martins, P. "A Hierarchy of Hop-Indexed Models for the Capacitated Minimum Spanning Tree Problem". *Networks*, v. 35, pp. 1–16, 2000.

[41] Gouveia, L., Paixão, J. "Dynamic Programming Basead Heuristic for the Topological Design Access Networks". *Annals of Operations Research*, v. 33, pp. 305–327, 1991.

[42] Hall, L. "Experience with a Cutting Plane Algorithm for the Capacitated Spanning Tree Problem". *INFORMS Jounal on Computing*, v. 8, pp. 219–234, 1996.

[43] Held, M., Wolfe, P. "Validation of Subgradiente Optimization". *Mathematical Programming*, v. 6, pp. 62–88, 1974.

[44] Hunting, M., Faigle, U., Kern, W. "A Lagrangian Relaxation Approach to the Edge-Weighted Clique Problem". *European Journal of Operational Research*, v. 131, pp. 119–131, 1992.

[45] Kershenbaum, A. "Computing Capacitated Minimal Spanning Trees Efficiently". *Networks*, v. 4, pp. 299–310, 1974.

[46] Kershenbaum, A., Boorstyn, R. R., Oppenheim, R. "Second Order Greedy Algorithms for Centralized Teleprocessing Network Design". *IEEE Transactions on Communications Com*, v. 28, pp. 1835–1838, 1980.

[47] Kershenbaum, A., Boorstyn, R. R., Oppenheim, R. "Centralized Teleprocessing Network Design". *Networks*, v. 13, pp. 279–293, 1983.

[48] Kershenbaum, A., Chou, W. "A Unified Algorithm for Designing Multidrop Teleprocessing Network". *IEEE Transactions on Communications*, v. 22, pp. 1762–1772, 1974.

[49] Laporte, G., Nobert, Y. "Comb Inequalities for the Vehicle Routing Problem". *Methods of Operations Research*, v. 51, pp. 271–276, 1984.

[50] Lucena, A. "Steiner Problem in Graphs: Lagrangean Relaxation and Cutting-Planes". *COAL Bulletin*, v. 21, pp. 2–8, 1992.

[51] Lucena, A. "Tight Bounds for the Steiner Problem in Graphs". In *Proceedings of NETFLOW93*, pp. 147–154, 1992.

[52] Maculan, N. "A New Linear Programming Formulation for Shortest s-directed Spanning Tree Problem". *Jounal of Combinatorics Information and System Sciences*, v. 31, pp. 53–56, 1986.

[53] Maculan, N. "The Steiner Problem in Graphs". *Annals of Discrete Mathematics*, v. 31, pp. 185–212, 1987.

[54] Magnanti, T., Wolsey, L. "Optimal Trees". In *Network Models, Handbooks in Operations Research and Management Science*, v. 7, pp. 503–615. North-Holland, 1995.

[55] Malik, K., Yu, G. "A Branch and Bound Algorithm for the Capacitated Minimum Spanning Tree Problem". *Networks*, v. 23, pp. 525–532, 1993.

[56] Martello, S., Toth, P. *Knapsack Problems: Algorithms & Computer Implementations.* Wiley, Chichester, England, 1990.

[57] Martinhon, C., Lucena, A., Maculan, N. *A Relax and Cut Algorithm for the Vehicle Routing Problem.* Relatório Técnico, Laboratório de

Métodos Quantitativos, Departamento de Administração, Universidade Federal do Rio de Janeiro, 2000.

[58] Padberg, M., Rinaldi, G. "A Branch and Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems". *SIAM Review*, v. 33, pp. 30–60, 1991.

[59] Papadimitriou, C. H. "The Complexity of the Capacitated Tree Problem". *Networks*, v. 8, pp. 217–230, 1978.

[60] Patterson, R., Rolland, E., Pirkul, H. "A Memory Adaptive Reasoning Technique for Solving the Capacitated Minimum Spanning Tree Problem". *Journal of Heuristics*, v. 5, pp. 159–180, 1999.

[61] Prim, R. C. "Shortest Connection Networks and Some Generalizations". *Bell System Technical Journal*, v. 36, pp. 1389–1401, 1957.

[62] Rardin, R., Choe, U. *Tighter Relaxations of Fixed Charge Network Flow Problems.* Technical Report 3-79-18, Industrial Systems Engineering, Georgia Institute of Technology, Atlanta, 1979.

[63] Reinelt, G. "TSPLIB: A Traveling Salesman Problem Library". *ORSA Journal on Computing*, v. 3, pp. 376–384, 1991.

[64] Saraiha, Y., Gendreau, M., Laporte, G., Osman, I. "A Tabu Search Algorithm for the Capacitated Shortest Spanning Tree Problem". *Networks*, v. 29, pp. 161–171, 1997.

[65] Souza, M. C., Duhamel, C., Ribeiro, C. C. "A GRASP Heuristic for the Capacitated Minimum Spanning Tree Problem Using a Memory-Based Local Search Strategy", 2002. Submetido para Publicação.

[66] Szwarcfiter, J. L., Markenzon, L. *Estruturas de Dados e seus Algoritmos.* LTC - Livros Técnicos e Científicos S.A., Rio de Janeiro, Brasil, 1994.

[67] Thompson, P. M., Orlin, J. B. *The Theory of Cyclic Tranfers.* Working paper OR200-89, Operations Research Center, MIT, Cambridge, MA, 1989.

[68] Thompson, P. M., Paraftis, H. N. "Cyclic Transfer Algorithms for Multi-Vehicle Routing and Scheduling Problems". *Operations Research*, v. 41, pp. 935–946, 1993.

[69] Toth, P., Vigo, D. "An Exact Algorithm for the Capacitated Shortest Spanning Arborescense". *Annals of Operations Research*, v. 61, pp. 121–141, 1995.

[70] Wong, R. T. "A Dual Ascent Approach for Steiner Tree Problems on a Directed Graph". *Mathematical Programming*, v. 28, pp. 271–287, 1984.

[71] Yu, G., Fisher, M. L. *A Lagrangean Optimization Algorithm for the Asymmetric Non-Uniform Fleet Vehicle Routing Problem.* Working paper 89-03-10, Decision Science Departament, The Wharton School, University of Pennsylvania, 1989.

[72] Zhang, N. *Facet-Defining Inequalities for Capacitated Spanning Trees.* Master's thesis, Princeton University, 1993.