Juliana Pontes de Lima

Universidade Federal do Rio de Janeiro
Instituto de Matemática - Núcleo de
Computação Eletrônica

Um Algoritmo Branch-And-Cut para o Problema de Roteamento de Veículos Capacitado Assimétrico

Dissertação de mestrado

Rio de Janeiro, RJ - Brasil

2005

Juliana Pontes de Lima

Um Algoritmo Branch-And-Cut para o Problema de Roteamento de Veículos Capacitado Assimétrico

Dissertação submetida ao Corpo Docente do Instituto de Matemática - Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para a obtenção do grau de Mestre em Informática.
Área de Concentração: Otimização

Orientadores: Prof[a]. Dr[a]. Marcia Helena Costa Fampa
Prof. Dr. Eduardo Uchoa Barboza

Rio de Janeiro, RJ - Brasil
2005

L732 Lima, Juliana Pontes.

Um Algoritmo branch-and-cut para o problema de roteamento de veículos capacitado assimétrico / Juliana Pontes de Lima. - Rio de Janeiro, 2005.

xii, 100f.; il.

Dissertação (Mestrado em Informática) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Núcleo de Computação Eletrônica, 2005.

Orientadores: Márcia Helena Costa Fampa; Eduardo Uchoa Barboza

1. Branch-and Cut - Teses. 2. Roteamento de Veículos - Teses. 3. Logística - Teses. 4. Otimização Combinatória - Teses. I. Márcia Helena Costa Fampa (Orient.). II. Eduardo Uchoa Barbosa (Orient.). III. Universidade Federal do Rio de Janeiro. Instituto de Matemática. Núcleo de Computação Eletrônica. IV.Título

CDD

Um Algoritmo Branch-And-Cut para o Problema de Roteamento de Veículos Capacitado Assimétrico

Juliana Pontes de Lima

Dissertação submetida ao Corpo Docente do Instituto de Matemática - Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para a obtenção do grau de Mestre em Informática.
Área de Concentração: Otimização

Aprovada por:

Marcia Helena Costa Fampa - D.Sc - UFRJ.

Orientador

Eduardo Uchoa Barboza - D.Sc. - UFF

Orientador

Mauro Antônio Rincon - D.Sc. - UFRJ

Presidente

Carlos Alberto de J. Martinhon - D.Sc. - UFF

Rio de Janeiro, RJ - Brasil
2005

Aos meus pais.

Agradecimentos

À Deus.

À minha família, pelo apoio incondicional e irrestrito, por toda a paciência e compreensão.

À família Rutledge, por todo o apoio e carinho.

Ao Luis Augusto, pelo apoio, dedicação e incentivo, pelo companheirismo, por me encorajar em momentos de desânimo, por estar sempre presente quando precisei.

À Professora e orientadora Marcia Fampa, pela sua dedicação, apoio e incentivo em todos os momentos, pelos conhecimentos e segurança transmitidos, por sua generosidade. Agradeço sua confiança em mim.

Ao orientador Eduardo Uchoa, por todo conhecimento transmitido, pela grande contribuição dada a este trabalho, pela sua dedicação e disponibilidade.

Aos professores do mestrado Mauro Rincon, Fábio Protti, Maria Helena, Facó, Fred, Claudio B., Nelson Maculan, por todo conhecimento transmitido.

Aos professores da graduação, Celso Costa, Ana Isabel, Jorge Delgado, Pierre, Ricardo, Paulo Gusmão, Haroldo Clark, por terem me preparado para o Mestrado.

Ao Professor da graduação Francisco Fontenele Neto, pela amizade e dedicação, desde os tempos da graduação, pelos conselhos preciosos.

Aos amigos do NCE, pela convivência e experiências trocadas, em especial, Vinícius e Giseli (também pela grande força nos "últimos instantes"), Jorge, Michaelle, Rafael, Paula, Christiane, Sandro, Carlos, Marcelo, Maise. Ao Flávio, por esclarecer muitas das minhas dúvidas sobre programação.

Aos amigos da COPPE, pela convivência e experiências trocadas, em especial, Thayse (também pelo suporte dado nos "últimos instantes"), Elivelton, Alexandre, Rafael, Ana Lúcia, Rosa, Adria e Yuri. Ao Luidi, pela grande ajuda na implementação.

Aos professores Carlos Alberto de J. Martinhon e Mauro Rincon pela participação na banca examinadora desta tese.

Ao Instituto de Matemática e ao Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro pela oportunidade de desenvolver este trabalho.

À Capes, pela bolsa de estudos.

Enfim, agradeço à todos aqueles que me ajudaram, direta ou indiretamente.

Resumo

Esta tese propõe procedimentos de separação das Desigualdades de Capacidade para o Problema de Roteamento de Veículos Capacitado Assimétrico (ACVRP). Tais procedimentos surgiram de adaptações daqueles já existentes para a versão simétrica do mesmo (SCVRP). São eles: a heurística das componentes conexas, a heurística de contração e o método de fluxo máximo para a separação das Desigualdades de Capacidade Fracionárias. Implementamos o primeiro deles e o inserimos em um algoritmo *Branch-And-Cut*, onde os limites inferiores em cada nó da árvore de enumeração são determinados pelo algoritmo de planos de cortes, que incorpora Desigualdades de Capacidade válidas para o problema original. Os resultados obtidos, com instâncias previamente utilizadas na literatura, foram comparados com os obtidos por dois algoritmos *Branch-And-Bound*. Apesar de sua importância em casos em que todo trajeto possui sentido obrigatório, o ACVRP tem recebido pouca atenção dos pesquisadores. Desta forma, os algoritmos *Branch-And-Bound* ainda representam o estado da arte com relação aos métodos exatos e, então, os procedimentos de *bounding* são o enfoque principal do estudo do ACVRP. A qualidade dos limites inferiores fornecidos por estes procedimentos é, em geral baixa, com excessão do método aditivo proposto em 94. Nosso algoritmo de planos de corte gerou limites inferiores de boa qualidade que, para a maioria das instâncias testadas, superaram aqueles gerados pelos procedimentos mencionados acima. Isto resultou, muitas vezes, numa menor quantidade de subproblemas gerados.

Palavras-chave:

Branch-and-Cut, Roteamento de Veículos, Logística, Otimização Combinatória.

Abstract

This thesis proposes separation procedures of the Capacity Inequalities for the Asymmetric Capacitated Vehicle Routing Problem (ACVRP). These procedures were extensions of their symmetric version. The procedures considered are: the connected component heuristic, the shrinking heuristic and the maximum flow method for the fractional capacity inequality's separation. We implemented the first of them and inserted it in a Branch-And-Cut algorithm, where lower bounds in each node of enumeration tree are determined by the cutting planes algorithm, which includes valid capacity inequalities to the original problem. The computational results, obtained with instances already used in the literature, were compared to the ones obtained by two Branch-And-Bound Algorithms. In spite of its importance, in cases where every way can be traversed in only one direction, the ACVRP has received few attention from researchers. In this way, the Branch-And-Bound algorithms still represent the state of art with respect to exacts methods and, then, the bounding procedures represent the main focus of the ACVRP study. The lower bounds quality, provided by the existent procedures, is generally poor, with exception of the additive method proposed in 94. Our cutting planes algorithm produced excellent lower bounds that, for the most instances tested, overcame those generated by the procedures mentioned above. This resulted, in many times, in a lower number of subproblems generated.

Palavras-chave:

Branch-and-Cut, Vehicle Routing, Logistic, Combinatorial optimization.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A Logística é um setor que cuida da obtenção, estocagem, distribuição e armazenamento dos produtos de uma determinada companhia. Além disso, é responsável pelo controle do processo produtivo na busca da redução dos custos e da otimização da produção.

Uma etapa de extrema importância em um sistema logístico é, sem duvida, a distribuição das mercadorias uma vez que ela, por si só, engloba elevados custos, influenciando diretamente no preço final das mercadorias transportadas. Além das questões econômicas, podemos citar, também, a segurança como uma forte razão para uma distribuição eficiente. Alguns exemplos, onde a questão da segurança é muito importante, são as situações onde os produtos transportados são alimentos, medicamentos, combustíveis ...

O problema da distribuição constitui um subproblema de um dos mais importantes e estudados problemas de otimização combinatória, denominado, genericamente, Problema de Roteamento de Veículos ou VRP (sua sigla em inglês). O VRP refere-se a uma família de problemas e possui uma quantidade enorme de aplicações práticas tais como a

distribuição de bebidas, de jornais, de derivados de petróleo, coleta de lixo, roteamento de helicóptero, transporte escolar, entre outros.

No VRP, o que se deseja é determinar um conjunto ótimo de rotas a serem percorridas por uma dada frota de veículos para atender um conjunto de clientes. Estas rotas devem estar satisfazendo a uma série de restrições. Devido a sua alta complexidade, nenhuma técnica para a solução exata do VRP com todas as restrições do "mundo real" foi proposta na literatura.

As principais variantes do VRP são aquelas que possuem restrições associadas à capacidade dos veículos. Podemos citar, entre elas, o VRP com janelas de tempo(VRP *with Time Windows*), VRP com entrega e coleta em cada cliente (VRP *with Pickup and Delivery*), VRP com entrega na ida e coleta na volta (VRP *with Backhauls*), entre outras. Em sua versão mais básica, denominada Problema de Roteamento de Veículos Capacitado (CVRP), é disponibilizada uma frota de K veículos idênticos para atender a um conjunto de n clientes, cada qual com uma demanda específica associada. Deve-se determinar K rotas a serem percorridas pelos veículos, todas elas partindo e retornando ao depósito, de modo que cada cliente seja visitado exatamente uma vez. O objetivo é obter custo total mínimo no atendimento aos clientes, nunca excedendo a capacidade de cada veículo. O CVRP é um problema *NP*-difícil uma vez que este generaliza o Problema do Caixeiro Viajante (ou TSP, sua sigla em inglês), onde podemos assumir que a capacidade do veículo é maior que a soma das demandas de todos os n clientes, sendo necessária, portanto, a utilização de apenas um veículo.

A partir do CVRP, podemos, ainda, fazer distinção de dois problemas, de acordo com o grafo representativo da rede viária (composta de estradas e pontos de conexão) associada ao problema, utilizada para o transporte das mercadorias. A não-orientação do referido grafo nos leva a definir o Problema de Roteamento de Veículos Capacita-

do Simétrico (SCVRP) enquanto que a orientação do mesmo, produz o Problema de Roteamento de Veículos Capacitado Assimétrico (ACVRP), sendo este último, objeto de estudo do nosso trabalho. Durante toda a dissertação, no entanto, estaremos tratando também do problema simétrico uma vez que grande parte da teoria por nós utilizada no tratamento do ACVRP foi previamente proposta para o mesmo.

O SCVRP tem recebido muita atenção da comunidade de Otimização desde que foi proposto em 1959 por Dantzig e Ramser [15]. A partir daí, muitas foram as técnicas desenvolvidas para tentar solucioná-lo. Entre elas, podemos citar os métodos de Geração de Colunas, Relaxação Lagrangeana e Relax-And-Cut, que estão entre as melhores abordagens do problema nos anos 80 e 90 [8, 3, 11, 32]. Com o desenvolvimento dos algoritmos *Branch-And-Cut*, em meados dos anos 90 (1995), a maioria das pesquisas concentram-se na descrição da envoltória convexa de tal problema uma vez que melhores resultados foram obtidos no tratamento do problema em questão [31, 38, 9].

Em [37], foi apresentado um algoritmo *Branch-And-Cut-and-Price* Robusto (o termo "robusto"significa que a adição de cortes não alteram a estrutura do subproblema Pricing) que aplicado ao SCVRP [21] gerou grandes resultados resolvendo, inclusive, instâncias que estavam em aberto a décadas e que não se esperava estarem próximos de serem solucionadas.

Enquanto que o SCVRP tem sido objeto de estudo de muitos pesquisadores, pouco se fala sobre o ACVRP, apesar de sua importância em casos em que todo trajeto possui sentido obrigatório. O que se tem até agora são trabalhos implementando métodos *Branch-and-Bound* puros [19, 27]. Tais algoritmos ainda representam o estado da arte com relação aos métodos exatos e, desta forma, procedimentos de *bounding* representam o enfoque principal do estudo do ACVRP.

Nesta dissertação, propomos um algoritmo *Branch-And-Cut* para resolver o ACVRP.

Os resultados computacionais obtidos na implementação deste algoritmo, serão comparados com aqueles de [19] e [27].

No segundo capítulo, apresentamos o Problema de Rotemento de Veículos e suas principais variantes, como o Problema de Roteamento de veículos com restrição de Janelas de tempo e aquela com restrições de capacidade. Para esta segunda, fazemos uma descrição detalhada de cada um dos dois casos já mencionados anteriormente: o Simétrico e o Assimétrico. Porém, cabe salientar que o enfoque principal de nosso estudo é o segundo caso.

No terceiro capítulo, faremos um estudo dos métodos de solução de problemas inteiros que possuem ligação direta com a abordagem desta dissertação (*Branch-And-Cut*). Estes métodos são: Algoritmos de Planos de Cortes, *Branch-And-Bound* e *Branch-And-Cut*. No quarto capítulo, é feito um resumo dos métodos de solução do SCVRP e do ACVRP. Falaremos do ACVRP com mais detalhes.

No quinto Capítulo, descreveremos o nosso algoritmo *Branch-And-Cut* para a resolução do ACVRP. E no sexto capítulo, faremos um estudo de três procedimentos de separação das Desigualdades de Capacidade para o Problema de Roteamento de Veículos Capacitado Assimétrico. Tais procedimentos constituem extensões daqueles já existentes para a versão simétrica do mesmo. São eles: a heurística das componentes conexas, a heurística de contração e o procedimento exato de separação das desigualdades de Capacidade Fracionárias. O procedimento implementado foi o primero mencionado.

No sétimo capítulo, faremos comparações de nossos resultados computacionais com os melhores resultados encontrados na literatura [19, 27]. E, por fim, no oitavo capítulo, apresentaremos nossas conclusões sobre os resultados obtidos e deixaremos sugestões para trabalhos futuros.

Capítulo 2

O Problema de Roteamento de Veículos (VRP)

O Problema de Roteamento de Veículos (VRP) consiste de uma família de problemas combinatórios que visam a determinação de um conjunto de rotas ótimas, a serem percorridas por uma frota de veículos, para servir um conjunto de clientes. As rotas devem estar satisfazendo à restrições operacionais que dependem de fatores, como a natureza das mercadorias a serem transportadas, as características dos clientes e veículos, entre outros.

Ao VRP corresponde uma série de aplicações práticas relacionadas à indústria, ao comércio, ao setor de serviços, ao lazer, entre outros. Podemos destacar as seguintes aplicações:

- Distribuição de jornais;
- Distribuição de derivados de Petróleo;
- Roteamento de Helicópteros;

- Sistema de transportes coletivos urbanos;
- Recolhimento de lixo;
- Patrulhamento policial;
- Distribuição e Recolhimento de leite, etc;

Nesta dissertação, estamos interessados no problema de distribuição de mercadorias, subproblema de grande importância, dentro do VRP, devido ao fato de que ele engloba elevados custos encarecendo o preço final do produto. Este pode ser definido como o problema de determinar rotas em uma rede viária apropriada, cada uma realizada por um veículo, que começa e termina em um depósito, tal que todas as demandas dos clientes são satisfeitas, todas as restrições operacionais são respeitadas e o custo global de transporte é minimizado.

As principais variantes do VRP são aquelas que levam em conta a capacidade dos veículos. Algumas delas são: VRP com restrições de capacidade (*Capacitated* VRP), VRP com janelas de tempo (VRP *with Time Windows*), VRP com entrega e coleta em cada cliente (VRP *with Pickup and Delivery*) e o VRP com entrega na ida e coleta na volta (VRP with *Backhauls*). Nesta tese, estaremos tratando do CVRP, em especial do ACVRP , que será visto com detalhes na próxima seção. Todas as variantes mencionadas anteriormente constituem uma extensão deste problema e terão suas particularidades descritas, brevemente, nas seções seguintes.

2.1 O Problema de Roteamento de Veículos Capacitado

O Problema de Roteamento de Veículos Capacitado, conhecido por CVRP (*Capacitated Vehicle Routing Problem*), consiste da versão mais básica do VRP. Neste problema, são dados uma frota homogênea composta de K veículos com capacidade C, e um grafo $G = (V, A)$ completo, onde $V = \{0, 1, \ldots, n\}$ é o conjunto de vértices, que representam o depósito(vértice 0) e os clientes (demais vértices), e A é o conjunto de arcos, que são às conexões entre clientes e entre cliente e depósito. A cada nó $i \in V$ é associada uma demanda d_i, que será não-negativa se $i \neq 0$ e nula se $i = 0$, e a cada arco (i, j), um valor não-negativo c_{ij}, que representa o custo de ir do cliente i ao cliente j. O CVRP tem por objetivo encontrar um conjunto de K rotas (cada uma correspondendo a um veículo) de modo a minimizar a soma dos custos das rotas. Essas K rotas devem estar satisfazendo as seguintes restrições:

- Toda rota i inicia e termina no depósito;
- Cada cliente será servido exatamente uma vez;
- A soma das demandas em cada rota não excede C.

A figura 4.4, a seguir, ilustra uma solução para uma instância do ACVRP, onde a quantidade de nós do grafo é 9, com demanda unitária para os nós clientes, o número de veículos disponiveis é 3, cada um com capacidade de 3. O quadrado representa o depósito e os circulos, os clientes.

Se os custos das arestas são tais que $c_{ij} = c_{ji}$, $\forall (i, j) \in A$ então o grafo G é não direcionado e o problema correspondente é denominado Problema de Roteamento de

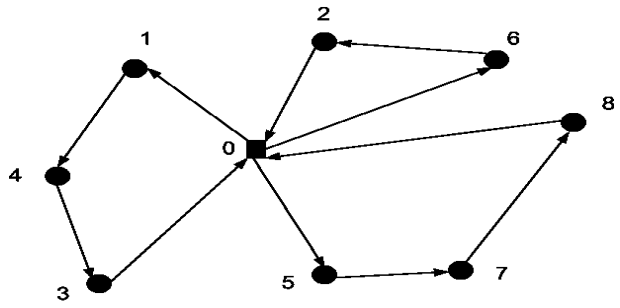

Figura 2.1: Solução viável para o ACVRP

Veículos Capacitado Simétrico (SCVRP). Caso contrário, teremos que G é direcionado e o problema correspondente é denominado Problema de Roteamento de Veículos Capacitado Assimétrico (ACVRP). Na versão simétrica do problema, o conjunto dos arcos A será substituído pelo conjunto de arestas E.

O CVRP generaliza o bem conhecido Problema do caixeiro viajante, ou TSP (sua sigla em inglês), que é o CVRP cuja capacidade dos veículos é maior que a soma das demandas de todos os clientes e, portanto, apenas um veículo é utilizado. Desta forma, o CVRP é um problema NP-difícil. Quando, no CVRP, desconsideramos o requerimento de capacidade dos veículos mas mantemos a sua quantidade, o problema será referido como k-TSP.

Para o que segue, seja $V_0 = \{1, ..., n\}$ o conjunto de vértices associados aos clientes. Para $S \subseteq V_0$ qualquer, define-se $d(S)$ como sendo a demanda total do conjunto S, onde $d(S) = \sum_{i \in S} d_i$. Utilizamos um limite inferior para o número mínimo de veículos necessário para servir todos os clientes em S. Este será denotado por $r(S)$, onde $r(S) = \lceil (d(S)/C) \rceil$. Outros valores para $r(S)$ são encontrados na literatura, por exemplo, ele pode ser determinado através da resolução de um problema denominado Bin Packing. Tal problema consiste na determinação de uma partição de S com um número mínimo de subconjuntos (bins), tal que a demanda total de cada subconjunto não excede a

capacidade C.

2.1.1 O Problema de Roteamento de Veículos Capacitado Simétrico

Para a formulação da versão simétrica do SCVRP, as arestas de E serão representadas por e . As variáveis de decisão x_e serão definidas como sendo o número de vezes que a aresta e é percorrida na solução. Além disso, $x_e \in \{0,1\}$ se $e \notin \delta(0)$ e $x_e \in \{0,1,2\}$ se $e \in \delta(0)$, onde $\delta(S)$ é o conjunto de arestas que possuem uma extremidade em S, quando $S = \{i\}$ escreveremos $\delta(i)$ e não $\delta(\{i\})$. Se x_e assume valor 1 ou 2 dizemos que e está na solução do SCVRP. Caso contrário, e não está na solução. A formulação do SCVRP é dada por:

$$min \sum_{e \in E} c_e x_e \tag{1.1}$$

sujeito a:

$$\sum_{e \in \delta(i)} x_e = 2, \forall i \in V_0, \tag{1.2}$$

$$\sum_{e \in \delta(0)} x_e = 2k, \tag{1.3}$$

$$\sum_{e \in \delta(S)} x_e \geq 2r(S), \forall S \subseteq V_0, S \neq \emptyset, \tag{1.4}$$

$$x_e \in \{0,1\}, \forall e \notin \delta(0), \tag{1.5}$$

$$x_e \in \{0,1,2\}, \forall e \in \delta(0), \tag{1.6}$$

As restrições de grau (1.2) e (1.3) asseguram que os clientes sejam visitados exatamente uma vez e que k veículos sejam utilizadas na solução. As desigualdades (1.4) são chamadas Desigualdades de Capacidade Simétricas (DCS) e impõem o requerimento de capacidade dos veículos e a conexidade das rotas. Se $r(S) = d(S)/C$ então a desigualdade $\sum_{e \in \delta(S)} x_e \geq 2r(S)$ será denominada Desigualdade de Capacidade Fracionária

Simétrica (DCFS) e, quando esta estiver sendo considerada (capítulo 6), a desigualdade (1.4) será referida como Desigualdade de Capacidade Arredondada Simétrica (DCAS). Repare que as DCAS dominam as DCFS.

No capítulo 4, faremos um resumo dos principais métodos de solução do CVRP simétrico utililizados nos últimos anos.

2.1.2 O Problema de Roteamento de Veículos Capacitado Assimétrico

Na formulação do CVRP assimétrico, os arcos direcionados de A serão denotados por a e, similarmente ao caso SCVRP, as variáveis de decisão x_a indicam a quantidade de vezes que o arco a é percorrido na solução. Além disso, $x_a \in \{0,1\}, \forall a \in A$. $\delta^-(S)$ é o conjunto dos arcos que entram em S e $\delta^+(S)$, o conjunto daqueles que saem de S. Diremos que o arco a está na solução caso $x_a = 1$. A formulação do ACVRP é a seguinte:

$$min \sum_{a \in A} c_a x_a \tag{2.1}$$

sujeito a:

$$\sum_{a \in \delta^-(i)} x_a = 1, \forall i \in V_0, \tag{2.2}$$

$$\sum_{a \in \delta^+(i)} x_a = 1, \forall i \in V_0, \tag{2.3}$$

$$\sum_{a \in \delta^-(0)} x_a = k, \tag{2.4}$$

$$\sum_{a \in \delta^+(0)} x_a = k, \tag{2.5}$$

$$\sum_{a \in \delta^-(S)} x_a \geq r(S), \forall S \subseteq V_0, S \neq \emptyset, \tag{2.6}$$

$$x_a \in \{0,1\}, \forall a \in A, \tag{2.7}$$

As restrições (2.2), (2.3), (2.4) e (2.5) garantem que os clientes serão visitados exata-

mente uma vez e que exatamente k veículos serão utilizadas na solução. As restrições (2.6), chamadas Desigualdades de Capacidade Assimétricas (DCA) estão garantindo que a capacidade dos veículos não é excedida e que as rotas são conexas com o depósito. Se $r(S) = d(S)/C$ então a desigualdade $\sum_{a \in \delta^-(S)} x_a \geq r(S)$ será denominada Desigualdade de Capacidade Fracionária Assimétrica (DCFA) e, quando esta estiver sendo considerada (capítulo6), a desigualdade (1.4) será referida como Desigualdades de Capacidade Arredondada Assimétricas (DCAA). Repare que as DCAA dominam as DCAF.
Note que as restrições (2.2)-(2.5) implicam que

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} = \sum_{i \in S} \sum_{j \notin S} x_{ij}, \quad \text{para} \quad \text{cada} \quad S \subset V, S \neq \emptyset. \tag{2.7}$$

O que quer dizer que no corte de qualquer conjunto de clientes $S \subseteq V \setminus \{0\}$ a quantidade dos arcos que entram em S é igual a dos arcos que saem.

No capítulo 4, descreveremos dois algoritmos *Branch-And-Bound* para o CVRP assimétrico. Tais algoritmos são considerados atuais uma vez que não são conhecidos outros algoritmos exatos propostos para o problema em questão.

2.2 O Problema de Roteamento de Veículos com Janelas de Tempo

Como uma das mais importantes variantes do VRP, podemos citar o Problema de Roteamento de Veículos com Janelas de tempo (VRPTW), que constitui uma extensão do CVRP. Além das características já mencionadas do CVRP, este problema inclui, para o depósito e para cada cliente i ($i = 0, \ldots, n$) uma janela de tempo $[a_i, b_i]$ que representa o intervalo de tempo dentro do qual o veículo deve começar o serviço em i, se este representa um nó cliente ($i = 1, \ldots, n$). A este é permitido chegar antes de a_i porém

terá que esperar até o instante a_i para serví-lo. Além disso, é proibido chegar depois de b_i. Quando $i = 0$, a janela de tempo significa que o veículo não poderá deixar o depósito antes de a_0 e terá que estar de volta ao mesmo até b_0. Podemos assumir que todos os veículos deixam o depósito no instante de tempo a_0, com $a_0 = 0$. São dados o tempo de viagem t_{ij} entre cada par de nós, $i, j \in V$, e um tempo de serviço s_i para cada cliente i. Então, o veículo gastará o tempo s_i em cada cliente i.

O VRPTW consiste em encontrar um conjunto de k rotas com custo mínimo satisfazendo:

- Cada rota inicia e termina no depósito;
- Cada cliente será servido exatamente uma vez;
- A demanda total dos clientes visitados em cada rota não excede a capacidade do veículo C;
- Para cada cliente i, o serviço inicia dentro da janela de tempo $[a_i, b_i]$ e o veículo gasta o tempo s_i servindo este cliente.

Repare que satisfazer o requerimento de janela de tempo para cada cliente induz uma orientação nas rotas mesmo que a matriz de custos do problema seja simétrica. Conseqüentemente, o VRPTW é normalmente formulado como um problema assimétrico.

O VRPTW é NP-difícil uma vez que generaliza o CVRP, para $a_i = 0$ e $b_i = +\infty, \forall i \in V \setminus \{0\}$. Alguns algoritmos exatos para este problema são encontrados em [28, 12, 25].

2.3 O Problema de Roteamento de Veículos com Entrega na Ida e Coleta na Volta

Outra extensão do CVRP de grande importância é o VRP com entrega na ida e coleta na volta (VRPB). Neste problema, o conjunto de clientes $V \setminus \{0\}$ é particionado em 2 subconjuntos. No primeiro, encontram-se os clientes que irão apenas receber mercadorias oriundas do depósito (clientes *Linehaul*) e, no segundo, aqueles que somente terão suas mercadorias coletadas (clientes *Backhaul*). Então, toda rota que serve clientes dos dois tipos devem estar respeitando uma regra de precedência entre eles: clientes *Linehaul* devem ser servidos antes de qualquer cliente *Backhaul*. Além disso, nenhuma rota poderá servir apenas clientes *Backhaul*. As quantidades d_i (associadas a cada cliente i) a serem coletadas ou distribuídas são conhecidas previamente. Assim como no CVRP, a frota de veículos é homogênea. Desta forma, o VRPB consiste em encontrar uma coleção de k rotas de menor custo tal que:

- Cada rota inicia e termina no depósito;
- Cada cliente será servido exatamente uma vez;
- A demanda total dos clientes *Linehaul* e *Backhaul* visitados por uma rota não excede, separadamente, a capacidade do veículo C;
- Em toda rota os clientes *Linehaul* são atendidos antes dos *Backhaul*.

Repare que a regra de precedência entre os dois tipos de clientes induz uma orientação nas rotas, independentemente da matriz de custos do problema. Quando a matriz de custos for assimétrica, o problema é denominado VRP assimétrico com entrega na ida e coleta na volta (AVRPB).

Note que o CVRP é o VRPB quando não existirem clientes *Backhaul*. Isto quer dizer que o VRPB generaliza o CVRP e portanto, é um problema NP-Difícil. Algoritmos exatos para este problema são dados nas referências [34, 41].

2.4 O Problema de Roteamento de Veículos com Coleta e Entrega em Cada Cliente

O Problema de Roteamento de Veículos com Coleta e Entrega (VRPPD) é outra extensão do CVRP, não menos importate do que as duas anteriores. Nela, além das características já vistas para o CVRP, teremos que, associado a cada cliente i, há dois valores d_i e p_i correspondentes às demandas a serem entregues e coletadas em i, respectivamente. Então, para cada cliente i, existem um nó destino D_i da demanda a ser coletada e um nó origem O_i da demanda a ser entregue neste cliente. Em cada cliente, a entrega é realizada antes da coleta. A carga atual do veículo é dada pela carga inicial menos a demanda total já distribuída mais toda a demanda já coletada. O VRPPD tem por objetivo encontrar um conjunto de k rotas com custo mínimo que satisfaça:

- Cada rota inicia e termina no depósito;
- Cada cliente será servido exatamente uma vez;
- A carga atual do veículo em cada ponto da rota deve ser não-negativa e não poderá exceder a capacidade do veículo;
- Para cada cliente i, o nó O_i deverá ser atendido na mesma rota que i e antes dele;
- Para cada cliente i, o nó D_i deverá ser atendido na mesma rota que i e depois dele;

A ordenação na visita dos nós de origem e de destino, para cada cliente, significa que toda rota está implicitamente orientada.

Quando, os nós O_i e D_i, $\forall i = 1, \ldots, n$, estão associados ao depósito ($O_i = D_i = 0$) e a demanda a ser coletada é nula, para todos os clientes i, o VRPPD recai no CVRP. Isto quer dizer que este problema generaliza o CVRP e portanto, é NP-difícil. Podemos encontrar algoritmos exatos aplicados ao VRPPD em [17], [16], [39] e [40].

Capítulo 3

Algoritmos em Programação Linear Inteira

Neste capítulo, faremos uma abordagem dos métodos exatos de soluções de problemas inteiros, diretamente relacionados com aquele utilizado nesta dissertação para a resolução do ACVRP. São eles: os algoritmos de planos de corte, *Branch-And-Bound* e *Branch-And-Cut* [44, 13, 45], que podem ser aplicados ao problema (P), definido abaixo:

$$\text{(P)} \qquad min\{cx : x \in X\}, \text{ onde } X = \{x : Ax \leq b, x \in \mathbb{Z}_+^n\},$$

3.1 Algoritmos de Planos de Cortes

Considerando o problema (P) definido acima, podemos reformulá-lo como o seguinte problema linear

$$\text{(PL)} \qquad\qquad min\{cx : x \in \tilde{X}\},$$

onde $\tilde{X} = conv(x) = \{x : \tilde{A}x \leq \tilde{b}, x \geq 0\}$ representa a envoltória convexa do conjunto de soluções viáveis de (P). Como cada ponto extremo do poliedro determinado por $\tilde{X}$ pertence a X, a solução ótima de (PL), para um dado valor de c, é uma solução ótima de (P). Então, teoricamente, podemos resolver o problema inteiro (P) como um problema de programação linear.

No entanto, para problemas NP-difíceis, como o ACVRP, a descrição completa de $\tilde{X}$ não é conhecida e, portanto, dada uma instância do problema (P), que pertence a esta classe, uma idéia é tentar obter ao menos uma boa aproximação para ele através da adição de desigualdades válidas para X, $Qx \leq q$ (uma desigualdade é dita válida para X se esta é satisfeita por todos os pontos de X). Caso sejam encontradas, tais desigualdades são incorporadas à formulação $F = \{x : Ax \leq b, x \geq 0\}$, $X = F \cap \mathbb{Z}^n$, do problema (P), obtendo formulação melhor $F' = \{x : Ax \leq b, Qx \leq q, x \geq 0\}$ com $X = F' \cap \mathbb{Z}^n$. Os algoritmos de Planos de Cortes funcionam desta forma e seu objetivo é que F' seja uma formulação mais próxima de $\tilde{X}$ nas redondezas do ótimo apenas e não que o descreva completamente.

Algoritmos de Planos de cortes para problemas inteiros gerais foram inicialmente propostos por Gomory [22] em 1958. Sua idéia original era gerar cortes a partir do arredondamento inteiro de uma restrição associada a uma variável básica fracionária, oriunda da resolução de uma relaxação de programação linear de (P). Por serem fracas, as desigualdades válidas assim construídas, levaram a uma convergência lenta desses algoritmos. Desta forma, os algoritmos de planos de cortes foram negligenciados durante muito tempo, até o desenvolvimento da teoria poliedral. O desenvolvimento desta teoria acarretou na introdução de desigualdades válidas fortes para problemas inteiros

específicos trazendo à tona tais algoritmos na década de 80.

Para a descrição dos algoritmos de planos de cortes, seja a relaxação linear de (P):

$$\text{(RL)} \qquad\qquad min\{cx : Ax \leq b, x \geq 0\}.$$

Então, dada a solução ótima $\overline{x}$ de (RL), se $\overline{x}$ satisfaz às condições de integralidade então $\overline{x}$ resolve (P). Se isto não ocorre, procuramos identificar um conjunto de desigualdades $\{\pi x \leq \pi_0\}$, pertencentes a uma dada família F_d de desigualdades válidas para X, violadas pela solução $\overline{x}$ de (RL). Se as referidas desigualdades são encontradas, estas são incorporadas à formulação $F = \{x : Ax \leq b, x \geq 0\}$ do problema. O problema obtido neste processo é, então, reotimizado. Este procedimento, constituído da identificação de desigualdades válidas e resolução da nova relaxação linear obtida, é repetido até que o algoritmo não consiga mais encontrar desigualdades de F_d violadas pela solução da relaxação atual ou até que uma solução inteira seja encontrada (solução ótima para (P)). O problema de identificação de desigualdades, pertencentes a uma família de desigualdades válidas para X, violadas por $\overline{x}$ é denominado Problema de Separação. A complexidade computacional deste problema varia de acordo com a família de desigualdades considerada e pode ser, inclusive, equivalente à complexidade de P.

Repare que ao inserir, na formulação atual, desigualdades válidas para X que estejam sendo violadas por $\overline{x}$, todos os vetores de X serão preservados na região poliedral correspondente a nova formulação do problema, porém, $\overline{x}$ será eliminado.

Em geral, não é possível resolver um problema inteiro apenas utilizando-se planos de cortes. Então, o que se faz, normalmente, é inserí-lo em um esquema *Branch-And-Bound*, que será visto na próxima seção, obtendo-se um algoritmo *Branch-And-Cut*, descrito na última seção deste capítulo.

3.2 Algoritmos *Branch-And-Bound*

Os algoritmos *Branch-And-Bound* foram propostos por Land e Doig [26] no início dos anos 60 para a resolução de problemas de programação inteira. Tais algoritmos consistem em uma forma inteligente de enumeração de todas as soluções de um determinado problema inteiro, em busca da melhor solução, uma vez que, para a maioria dos problemas, é totalmente impossível a enumeração completa, devido ao número exponencial de soluções. Para isso, o método combina as estratégias de *Branching* e *Bounding*. A primeira consiste em particionar o espaço de soluções $X = \{x : Ax \leq b, x \in \mathbb{Z}_+^n\}$, do problema P definido no início deste capítulo, em k subespaços menores e disjuntos X_i, $i = 1, \ldots, k$, tais que $(\bigcup_{i=1}^{k} X_i) = X$, nos quais serão definidos os problemas P_i da forma $min\{cx : x \in X_i\}$, $i = 1, \ldots, k$. A segunda estratégia busca encontrar a solução ótima da relaxação de Programação linear $\overline{P}_i$ de cada P_i. O valor $\underline{z}_i$, associado a esta solução, representa um limite inferior para P_i uma vez que o espaço das soluções viáveis $\overline{X}_i$ de $\overline{P}_i$ é tal que $X_i \subseteq \overline{X}_i$, $\forall i = 1, \ldots, k$.

Os algoritmos *Branch-And-Bound* começam por resolver a relaxação de programação linear do problema original P, obtendo uma solução $\overline{x}$ com valor $\underline{z} = c\overline{x}$. Se $\overline{x}$ é inteiro, obviamente, $x^* = \overline{x}$ é solução ótima e $z^* = \underline{z}$ é o valor ótimo correspondente (durante todo o procedimento, x^* e z^* irão guardar a melhor solução viável conhecida para P e o seu valor associado, respectivamente). Caso contrário, admite-se que $z^* = +\infty$ é o melhor limite superior conhecido para P e, portanto, $\underline{z} < z^*$. Neste caso, é realizado o primeiro *Branching* gerando $p \geq 2$ novos problemas P_i da forma $min\{cx : x \in X_i\}$, $i = 1, \ldots, p$. Então, resolve-se os p subproblemas gerados pelo *Branching* ao invés de resolver o problema P.

Na iteração seguinte, segundo algum critério, escolhe-se um problema P_i, $i = 1, \ldots, p$, e resolve-se a sua relaxação linear $\overline{P}_i$. Se a solução $\overline{x}_i$ obtida for tal que $\underline{z}_i = c\overline{x}_i \geq z^*$, não

será mais necessário prosseguir com o particionamento deste problema já que uma solução melhor do que a já conhecida garantidamente não estará em seu conjunto de soluções viáveis. Caso contrário, se a solução $\overline{x}_i$ for inteira então esta é solução ótima de P_i com valor $\overline{z}_i = \underline{z}_i = z_i^*$ associado. Se, além disso, $\overline{z}_i < z^*$, faz-se as atualizações $x^* = \overline{x}_i$ e $z^* = \overline{z}_i$. Caso contrário, se $\overline{x}_i$ é tal que nenhuma das situações anteriores se verificam, e P_i não é um problema inviável, então é realizado o *Branching* no conjunto X_i, gerando novos subproblemas. Desta forma, estes subproblemas recém criados serão resolvidos no lugar de P_i. O algoritmo prossegue com este mesmo procedimento (*Branching* e *Bounding*) até que, para todo subproblema gerado, tenhamos a garantia de que um limite inferior para ele é maior ou igual ao melhor limite superior conhecido para P ou, então, que ele é inviável. Desta forma, se $z^* = +\infty$, o problema P é inviável, senão, o valor da solução ótima é z^*.

O problema P e todos os subproblemas gerados constituem a chamada árvore de enumeração, associado ao algoritmo de *Branch-And-Bound*. O nó raiz de tal árvore representa o problema original P, e os demais, todos os subproblemas gerados pelo procedimento de *Branching*. Os nós associados aos subproblemas gerados pelo *Branching*, na região viável X_i, são chamados filhos do nó associado ao subproblema P_i.

A estratégia de enumeração, que seleciona o próximo problema a ser resolvido, deve ser escolhida de forma específica para cada problema. Desta forma, a experiência e o conhecimento a respeito do problema trabalhado devem ser levados em conta na tomada de decisão. A busca em largura, a busca em profundidade e a busca "melhor primeiro"são estratégias genéricas de enumeração freqüentemente utilizadas. Note que a escolha de tal estratégia reflete diretamente no tamanho da árvore do *Branch-And-Bound* e portanto no tempo de execução do algoritmo.

A escolha da estratégia de *branching* está totalmente atrelada à estrutura de cada

problema. Porém, um *Branching* comumente utilizado é aquele em que se particiona a região viável em duas, através da adição de desigualdades $x_j \leq \lfloor \overline{x}_j \rfloor$, para a primeira, e $x_j \geq \lceil \overline{x}_j \rceil$, para a segunda. A variável x_j é escolhida como sendo aquela de valor fracionário da solução do PL atual.

3.3 Algoritmos *Branch-And-Cut*

Os algoritmos *Branch-And-Cut*, assim nomeados por Padberg e Rinaldi [36], surgiram da combinação dos algoritmos de Planos de Cortes e *Branch-And-Bound*, descritos nas seções passadas. Neste algoritmo, além de particionarmos a região viável X do problema P, definido no início do capítulo, estaremos também incorporando desigualdades válidas para $conv(X)$.

O procedimento é análogo ao *Branch-And-Bound*, com a diferença que agora estaremos dispostos a empregar um maior esforço computacional, em cada nó da árvore de enumeração, com o procedimento de *Bounding*, que agora irá adicionar desigualdades globalmente válidas para o problema original P. O objetivo é obter limites inferiores locais melhores possibilitando, desta forma, a redução da quantidade de subproblemas gerados.

Nestes algoritmos, em cada nó da árvore de enumeração, inicialmente, é resolvida a relaxação linear $\overline{P}_i$ do problema associado ao nó, P_i, com sua formulação inicial, que, caso não seja o nó raiz, deverá conter desigualdades válidas incorporadas em iterações passadas. Caso a solução obtida $\overline{x}$ seja fracionária, tentaremos encontrar desigualdades, pertencentes a uma família F_d de desigualdades válidas para P, violadas por $\overline{x}$. Estas desigualdades, caso sejam encontradas, são incorporadas à formulação do problema P_i, que será reotimizado. Este procedimento será repetido até que nenhuma desigualdade seja

encontrada para eliminar a solução da relaxação de programação linear do P_i, com sua formulação atual, ou até que a solução da mesma seja inteira. No primeiro caso, o valor associado com a última solução obtida é um limite inferior para P_i. Então, prosseguimos com o *Branching* em X_i. No segundo caso, teremos resolvido P_i e atualizamos a melhor solução conhecida para o problema original, se for o caso. Veja que se P_i é o nó raiz, o problema terá sido resolvido e, portanto, nada mais será feito. É importante notar que, a solução ótima de cada P_i é viável para o problema original.

Os algoritmos *Branch-And-Cut* possuem grande importância se estamos resolvendo problemas de programação inteira, cuja formulação possui quantidade muito grande (possivelmente exponencial) de restrições. Neste caso, um subconjunto de tais restrições é inicialmente desconsiderada desta formulação e incorporada, à mesma, através do procedimento de *Bounding*, em cada nó da árvore de enumeração, de forma semelhante à descrita acima. A diferença é que no teste de viabilidade de $\overline{x}$ (para decidir se a identificação de desigualdades violadas será feita ou se tentamos atualizar a melhor solução viável conhecida para o problema original) deveremos verificar, além da integralidade de $\overline{x}$, se este satisfaz a todas as restrições descartadas no início do algoritmo. Isto deve ser feito porque se uma solução ótima para P_i, associado a um nó da árvore de enumeração, não satisfaz às tais restrições então não será viável para o problema original. Desta forma, mesmo que a solução $\overline{x}$ do problema $\overline{P}_i$ seja inteira, somente atualizaremos a melhor solução viável conhecida para o problema original caso esta satisfaça também ao referido conjunto de desigualdades. Caso contrário, pelo menos um plano de corte deverá ser separado.

Conforme já mencionado na seção 3.1, a complexidade computacional do problema de separação pode ser muito alta. Então, como a identificação de desigualdades é realizada diversas vezes ao longo do algoritmo *Branch-And-Cut*, utiliza-se, com freqüencia,

heurísticas para resolvê-lo.

Capítulo 4

Algoritmos de Soluções para o CVRP

No presente capítulo, faremos um resumo dos principais métodos de soluções do Problema de Roteamento de Veículos Capacitado, para ambos os casos: Simétrico e Assimétrico.

Para o caso Simétrico, falaremos brevemente sobre os principais algoritmos utilizados nos últimos anos. Tais métodos são o *Branch-And-Price*, o *Branch-And-Cut* e o *Branch-And-Cut-And-Price*. Para o Assimétrico, descreveremos dois algoritmos *Branch-And-Bound*. Este método ainda é bastante atual para o problema em questão, uma vez que não se tem notícia da aplicação de qualquer outro método exato ao mesmo.

4.1 Algoritmos de Soluções para o CVRP Simétrico

Nesta seção, daremos um histórico dos principais algoritmos exatos para a resolução do SCVRP.

Algoritmos baseados na relaxação Lagrangeana/Geração de colunas são comumente usados na resolução do SCVRP. Em 1981, Christofides, Mingozzi e Toth [11] propuseram uma abordagem em que os limites Lagrangeanos eram obtidos através da resolução do problema das q-rotas mínimas. Toda q-rota inicia no depósito, visita um conjunto de clientes, cujo a demanda total não excede C, e retorna ao depósito. Cada cliente deste conjunto poderá ser visitado mais de uma vez e, a cada visita, sua demanda será contabilizada. Podemos concluir, então, que todas as rotas viáveis do SCVRP estão inseridas no conjunto das q-rotas. Martinhon, Lucena, and Maculan [32] e Fisher [20], entre outros, utilizaram limites Lagrangeanos baseados em *K-trees* tendo grau $2K$ no depósito.

Em uma outra abordagem, os algoritmos exatos se baseiam na formulação de particionamento de conjunto do SCVRP, proposta originalmente por Balinski e Quandt [8]. Cada coluna da referida formulação cobre um conjunto de vértices clientes S, cujo a demanda total é no máximo C. Alguns destes algoritmos são apresentados por Balinski e Quandt [8], Agarwal [3], entre outros.

No início da década de 90, o foco principal da pesquisa do SCVRP tornou-se a descrição poliedral da envoltória convexa dos vetores que correspondem a k rotas viáveis, bem como o desenvolvimento de algoritmos de separação eficientes para as famílias de desigualdades válidas conhecidas. Em [1, 6, 5, 7, 14, 29, 35], são descritos algoritmos de separação das principais famílias de desigualdades válidas para o SCVRP, dentre elas, as desigualdades de Capacidade, *Combs* e *Multistars*. Algoritmos *Branch-And-Cut* completos são decritos em [2, 4, 9, 38]. Apesar dos bons resultados obtidos com estes algoritmos, muitas instâncias do problema com menos de 80 clientes não puderam ser resolvidas.

Em [37], foi apresentado um algoritmo exato que combina geração de colunas e geração de cortes para resolução de problemas inteiros. Este algoritmo foi denominado *Branch-And-Cut-and-Price* Robusto (o termo "Robusto"indica que a adição de restrições ao

problema não altera a estrutura do subproblema *pricing*). A aplicação de tal algoritmo ao SCVRP [21] conseguiu resolver todas as principais instâncias da literatura de até 135 clientes. Além disso, muitas das instâncias que se encontravam em aberto puderam ser resolvidas pela primeira vez.

4.2 Algoritmos de Soluções para o CVRP Assimétrico

Nesta seção, descreveremos os principais pontos de dois algoritmos *Branch-And-Bound* para o ACVRP. Estes referem-se aos métodos de *bounding*, *branching* e a estratégia de busca na árvore de enumeração do *Branch-And-Bound*.

Os limites inferiores, no primeiro algoritmo, foram obtidos através da resolução do problema de atribuição, resultante da relaxação das restrições de Capacidade de um problema equivalente ao ACVRP, enquanto que, no segundo, utilizou-se o procedimento aditivo, proposto por Fischetti e Toth [18], sobre duas relaxações baseadas na disjunção de subconjuntos inviáveis de arcos e no fluxo de custo mínimo.

Outros limites inferiores, resultantes de relaxações combinatórias do ACVRP e previamente utilizados em algoritmos *Branch-And-Bound* para resolução do mesmo, são baseados em arborescências e devem ser obtidos pelo enfraquecimento das restrições de capacidade, que agora irão levar em conta apenas a conectividade das soluções, e pela exclusão de uma parte das restrições de grau. Estes *bounds* possuem baixa qualidade, como pode ser verificado em [42], e quando inseridos em algoritmos *Branch-And-Bound*, consegue-se resolver apenas problemas pequenos até a otimalidade.

4.2.1 *Branch-And-Bound* baseado na relaxação das restrições de Capacidade

A primeira abordagem utilizando algoritmos *Branch-And-Bound* para resolver o ACVRP foi proposta por Laporte, Mercure e Nobert em [27]. Neste trabalho, é resolvido um problema (P) equivalente ao ACVRP obtido da adição de $k-1$ depósitos artificiais ao conjunto de nós $V = \{0, 1, \ldots, n\}$. Assim, obtemos o digrafo completo $G^{'} = (V^{'}, A^{'})$, onde $V^{'} = V \cup \{n+1, ..., n+k-1\}$ e $A^{'} = A \cup \{(i,j) : i, j \in V^{'}, i \neq j, i \text{ ou } j \in V^{'} \setminus V\}$,

extensão do original G. Os custos associados aos arcos deste novo grafo são:

$$c_a^{'} = \begin{cases} c_{ij} \quad \text{para} \quad a = (i,j), i,j \in V \setminus \{0\}, \\ c_{i0} \quad \text{para} \quad a = (i,j), i \in V \setminus \{0\}, j \in W \\ c_{0j} \quad \text{para} \quad a = (i,j), i \in W, j \in V \setminus \{0\} \\ \infty \quad \text{para} \quad a = (i,j), i,j \in W, \end{cases}$$

onde $W = \{0\} \cup \{n+1, ..., n+k-1\}$ é o conjunto dos k vértices associados aos depósitos. Desta forma, o problema (P) a ser resolvido visa determinar um único ciclo Hamiltoniano, contendo todos os nós de $V^{'}$, de forma que a demanda total $d(S)$ de todo conjunto S, de nós de $V \setminus \{0\}$, correspondente a uma seqüência de arcos (caminho) que começa e termina em um nó de W, não excede a capacidade do veículo.

A figura 4.1 abaixo ilustra um exemplo de solução deste novo problema, onde $k = 3$, $|V| = 9$, $V = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$, $C = 3$ e a demanda d_i, para cada i associado a um cliente, é unitária. Desta forma, a quantidade total de depósitos, resultantes da transformação, será três ($W = \{0, 9, 10\}$).

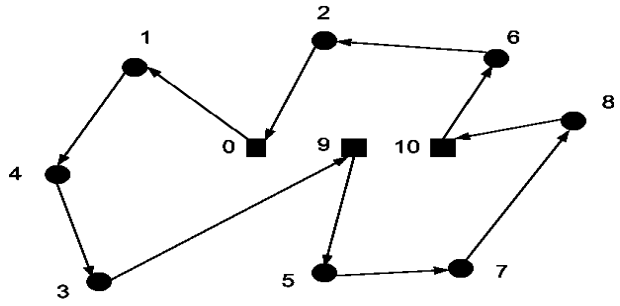

Figura 4.1: Solução do Problema (P)

O problema (P) mencionado acima possui a seguinte formulação:

(3.1) (P) $min \sum_{a \in A'} c'_a x_a$

sujeito a:

(3.2) $\sum_{a \in \delta^-(i)} x_a = 1, \forall i \in V',$

(3.3) $\sum_{a \in \delta^+(i)} x_a = 1, \forall i \in V',$

(3.4) $\sum_{a \in \delta^-(S)} x_a \geq \lceil d(S)/C \rceil, \forall S \subseteq V \setminus \{0\}, S \neq \emptyset,$

(3.5) $x_a \in \{0, 1\}, \forall a \in A',$

Se as restrições (3.4) forem relaxadas, o problema resultante é o de atribuição (*Assignment Problem* ou AP) cuja solução, obviamente, pode não ser viável para (P). Tais restrições garantem que nenhum nó será desconexo de W na solução ótima e que a demanda de um conjunto de nós clientes S, associado a um caminho que começa e termina em W, desta solução, não excede a capacidade do veículo.

A transformação acima foi proposta em [30] para converter o k-TSP no TSP. Para o ACVRP esta foi motivada pelo fato de que, na prática, resolver o AP é mais efetivo do que resolver o problema de transporte (*Transportation Problem* ou TP), que é obtido do ACVRP através da relaxção das restrições de Capacidade.

Uma vez obtida uma solução viável para (P), esta pode ser convertida para o ACVRP levando-se em conta as seguintes alterações nos arcos de A', pertencentes a esta solução:

- substituir (i, j) por $(i, 0)$ se $i \in V \setminus \{0\}$ e $j \in V' \setminus V$;
- substituir (i, j) por $(0, j)$ se $i \in V' \setminus V$ e $j \in V \setminus \{0\}$

A figura a seguir ilustra a conversão da solução de (P) para a solução do ACVRP. Os arcos pontilhados, incidentes aos nós de $W \setminus \{0\}$, em (b), foram substituídos por aqueles contínuos e mais escuros, incidentes ao nó depósito original representado por 0.

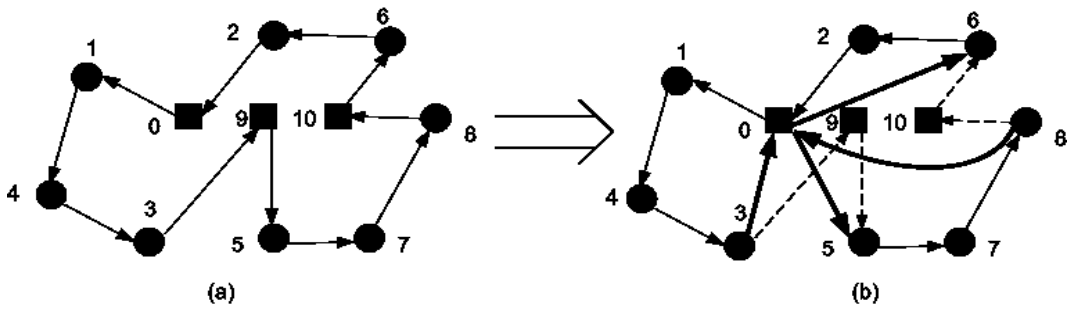

Figura 4.2: Obtenção da solução do ACVRP a partir da solução de (P)

Laporte, Mercure e Nobert [27] propuseram um *Branch-And-Bound* que procede da seguinte forma: em cada nó da árvore de enumeração, resolve-se, inicialmente, a relaxação de programação linear de AP, que deve conter variáveis fixas (em 0 ou 1), pelo procedimento de *Branching*, em iterações passadas. Devido a estrutura unimodular do problema, a solução será sempre inteira (coincide com a solução do AP) e, portanto, caso esta não seja viável para P (e o limite inferior seja menor que a melhor solução viável conhecida), deve-se "eliminá-la"particionando o subproblema atual h em subproblemas descendentes j, que serão caracterizados pelos conjuntos I_j e F_j de arcos impostos e proibidos da solução de j, respectivamente. O *Branching* será feito, como descrito abaixo, através da quebra de uma subrota violada (que poderá ser desconexa de W ou um caminho aberto começando e terminando em W), identificada na solução de h. Desta forma, as restrições relaxadas serão tratadas através da imposição da conectividade e do requerimento de capacidade das soluções viáveis do ACVRP. Em [27], são descritas 3 regras para o *branching*. Na primeira, obtida a partir de adaptações daquela usada por Carpaneto e Toth [10] na aplicação ao TSP, é escolhida uma subrota que possua a menor quantidade de arcos não pertencentes a I_h. Seja T o conjunto desses arcos, onde

$$T = \{(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_M, \beta_M)\}$$

É assumido que esses arcos são listados na ordem em que aparecem na subrota violada, onde (α_i, β_i) representa o i-ésimo arco, não pertencente a I_h, desta subrota.

Então, no procedimento de *branching* serão gerados M nós descendentes imediatos do nó h, onde para cada descendente j estarão associados dois conjuntos de arcos I_j e F_j, como abaixo:

$$I_j = \begin{cases} I_h, \text{ (j=1)} \\ I_h \cup \{(\alpha_u, \beta_u) : u = 1, \ldots, j-1\}, \text{ (j=2,...,M)} \end{cases}$$
$$F_j = F_h \cup \{(\alpha_j, \beta_j)\}, \text{(j=1,...,M)}$$

Com este procedimento, obtemos que a região viável de cada subproblema h está contida na união das regiões viáveis de seus subproblemas sucessores imediatos j. Tais regiões viáveis serão disjuntas duas a duas. Repare que, quando o subproblema h é o problema original, os conjuntos I_h e F_h são vazios.

Na figura 4.3 (a) abaixo está sendo representado o *Branching* em um nó h, onde $I_h = \{(7,6)\}$ e $F_h = \{(6,4)\}$. Na instância considerada, a quantidade de nós é 8, a demanda associada aos nós clientes é unitária, e o número de veículos é 3, todos com capacidade 3. Na solução de h, representada pelo grafo da figura 4.3 (b), temos uma subrota violada (excedendo a capacidade do veículo) composta dos arcos de $(8,4)$, $(4,7)$, $(\mathbf{7},\mathbf{6})$, $(6,5)$ e $(5,9)$. Teremos portanto, $T = \{(8,4),(4,7),(6,5),(5,9)\}$ e, então, 4 novos subproblemas descendentes diretos de h, onde os conjuntos I_j e F_j, $j = 1, \ldots, 4$ são definidos como:

$I_1 = I_h$; $F_1 = F_H \cup \{(8,4)\}$;

$I_2 = I_h \cup \{(8,4)\}$; $F_2 = F_H \cup \{(4,7)\}$

$I_3 = I_h \cup \{(8,4),(4,7),(7,6)\}$; $F_3 = F_H \cup \{(6,5)\}$

$I_4 = I_h \cup \{(8,4),(4,7),(7,6),(6,5)\}$; $F_4 = F_H \cup \{(5,9)\}$

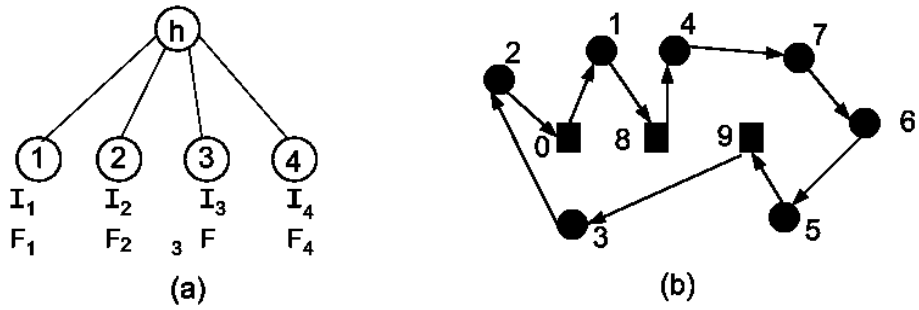

Figura 4.3: (a) Branching no nó h e (b) Solução inviável do subproblema associado a h

Na segunda regra, em cada nó descendente imediato de h serão proibidos $r = \lceil d(S)/C \rceil$ arcos, onde S é o conjunto de nós envolvidos na subrota violada, do conjunto T de arcos não impostos de alguma subrota violada identificada na solução de h. Desta forma, teremos $C(T,r)$ descendentes imediatos de h, obtidos por tomar todas as combinações de r arcos. Aqui, a subrota escolhida para o branching pode ser aquela que possui o menor valor de $C(T,r)$ ou, como na regra 1, a que possui a menor quantidade de arcos não impostos. A terceira regra é uma combinação das duas primeiras.

Repare que as restrições de capacidade violadas, associadas às subrotas consideradas no *branching*, são implicitamente incorporadas ao problema uma vez que tais subrotas são proibidas através daquele procedimento.

Foram feitos testes computacionais em [27] para todas as combinações das 3 regras de *branching* e das 2 de seleção dos nós, busca em profundidade e busca em largura. No entanto, apenas a melhor versão do algoritmo teve seus resultados apresentados. Esta versão utiliza a terceira regra de branching e a seleção dos nós por busca em largura.

4.2.2 *Branch-And-Bound* baseado no Procedimento Aditivo

O algoritmo *Branch-And-Bound* proposto por Fischetti, Toth e Vigo em [19], que será descrito a seguir, possui a mesma estrutura básica de [27]. Porém, enquanto que [27] usa a relaxação do AP para gerar limites inferiores, [19] usa o procedimento aditivo, proposto por Fischetti e Toth [18].

O procedimento aditivo combina q procedimentos de *bounding* $\varphi^{(1)}, \ldots, \varphi^{(q)}$, cada um explorando diferentes subestruturas do ACVRP. Cada $\varphi^{(i)}$, $i = 1, \ldots, q$, quando aplicado a uma instância do ACVRP, definido na seção 2.1.2, com custo c, irá retornar um limite inferior $\rho^{(i)}$ e um custo residual $\overline{c}^{(i)}$, tal que

$$\overline{c}^{(i)} \geq 0, \tag{4.1}$$

$$\rho^{(i)} + \overline{c}^{(i)} x \leq cx, \forall x \in P_{ACVRP}.$$

onde P_{ACVRP} é a envoltória convexa dos vetores x de soluções viáveis da formulação (2.1)-(2.6) do ACVRP.

Então, os procedimentos $\varphi^{(i)}, i = 1, \ldots, q$, serão aplicados, em seqüência, a uma instância do ACVRP que utiliza como custos a matriz de custos residuais retornada pelo procedimento anterior. Desta forma, para $h = 1, \ldots, q$, $\varphi^{(h)}$ é aplicado à instância com custo residual $\overline{c}^{(h-1)}$ retornado pelo procedimento $\varphi^{(h-1)}$ (note que o procedimento $\varphi^{(1)}$ é aplicado ao vetor de custos $\overline{c}^{(0)} = c$), e gera um custo residual $\overline{c}^{(h)}$ e um limite inferior $\rho^{(h)}$. O limite inferior total (ou limite inferior aditivo) é dado por $\rho^* = \rho^{(1)} + \ldots + \rho^{(q)}$. Sua validade pode ser verificada facilmente utilizando 4.1.

Em [19] são utilizados dois procedimentos de *bounding* para o ACVRP, um baseado na disjunção de subconjuntos inviáveis de arcos e o outro, na relaxação de fluxo de custo mínimo, deste problema. Cada procedimento computa uma seqüência de limites

inferiores de acordo com o procedimento aditivo e, enfim, são combinados segundo este mesmo procedimento para gerar os *bounds* utilizados pelo algoritmo *Branch-And-Bound*. Veremos abaixo cada um desses métodos:

- O primeiro método de Bounding estudado é baseado na disjunção de subconjuntos inviáveis de arcos.

  Diremos que um subconjunto de arcos $B \subset A$ é inviável se não existir solução x viável para o ACVRP que contenha todos os seus arcos. Se, ao retirar um arco de B, já encontro uma solução com todos os arcos restantes então B é um subconjunto minimal inviável de arcos. Seja, então, B um subconjunto minimal inviável de arcos $B \subset A$, teremos o seguinte:

  $$F \subset \bigcup_{(u,v)\in B} Q^{uv} = \{x \in \mathbb{R}^A : x_{uv} = 0\},$$

  onde F é o conjunto de todas as soluções viáveis do ACVRP. Esta inclusão ocorre porque para todo $x \in F$, existe pelo menos uma componente x_{uv}, com $(u, v) \in B$, tal que $x_{uv} = 0$ e, portanto, x pertence a um conjunto Q^{uv}, no mínimo. Isto é equivalente a seguinte disjunção lógica:

  $$\bigvee_{(u,v)\in B} (x \in Q^{uv}), \forall x \in F$$

  Então, $|B|$ problemas da forma $\nu(P^{uv}) = min\{cx : x \in F \cap Q^{uv}\}$, para cada $(u, v) \in B$, serão definidos e representados por P^{uv}. Para cada subproblema P^{uv}, um limite inferior válido v^{uv} e os custos residuais γ^{uv} serão computados através da resolução do problema de transporte TP^{uv}, obtido a partir da formulação do ACVRP dada em 2.1.2, com a relaxação das restrições de capacidade. Este problema possuirá a condição adicional $x_{uv} = 0$. Como a matriz de restrições do TP^{uv} é unimodular, sua solução ótima coincide com a solução de sua relaxação linear PL^{uv}. Então, o limite

inferior υ^{uv}, para o P^{uv}, será o valor da solução ótima de PL^{uv} e os custos residuais γ^{uv} coincidirão com os custos reduzidos correspondentes a υ^{uv}. A validade de tais custos residuais pode ser verificada utilizando a teoria de programação linear. Segue abaixo a formulação do TP^{uv}.

$$(4.1) \qquad (\mathrm{TP}^{uv}) \qquad min \sum_{a \in A} c_a x_a$$

sujeito a:

$$(4.2) \qquad \sum_{a \in \delta^-(i)} x_a = 1, \forall i \in V_0,$$

$$(4.3) \qquad \sum_{a \in \delta^+(i)} x_a = 1, \forall i \in V_0,$$

$$(4.4) \qquad \sum_{a \in \delta^-(0)} x_a = k,$$

$$(4.5) \qquad \sum_{a \in \delta^+(0)} x_a = k,$$

$$(4.6) \qquad x_a \in \{0, 1\}, \forall a \in A,$$

$$(4.7) \qquad x_a = 0, a = (u, v)$$

Portanto, um limite inferior para o ACVRP será dado por:

$$L_D = min\{\upsilon^{uv} : (u, v) \in B\},$$

e será denominado limite inferior disjuntivo. É evidente que L_D domina aquele produzido pela relaxação do ACVRP, que resulta no problema de transporte TP, sem qualquer restrição adicional, uma vez que $\upsilon^{uv} \geq L_{TP}, \forall (u, v) \in B$. Veja que $F \cap Q^{uv} \subseteq F, \forall (u, v) \in B$. Os custos residuais associados a L_D serão $\overline{c}_{ij} = min\{\gamma_{ij}^{uv} : (u, v) \in B\}$. Pode ser verificado facilmente que L_D e $\overline{c}$ computados desta forma satisfazem 4.1.

A seleção de diferentes subconjuntos inviáveis de arcos, deve levar a diferentes limites inferiores L_D. Portanto, [19] utilizou um procedimento aditivo, denominado ADD_DISJ, que considerou, em seqüência, diferentes subconjuntos inviáveis de arcos na tentativa de obter um limite inferior total melhor. O critério de escolha

de B será determinado abaixo, na descrição de ADD_DISJ.

Note que, exigir B minimal acarreta numa quantidade menor de problemas TP^{uv}, $(u, v) \in B$ a serem resolvidos. Veja que, caso B não seja minimal, para algum $(u, v) \in B$, ainda não haverá solução viável para o ACVRP que possua todos os arcos de $B \setminus \{(u, v)\}$. Então, toda solução x de F, com $x_{uv} = 0$, possuirá também, por exemplo, $x_{ab} = 0$ ou $x_{fd} = 0$, $(a, b), (f, d) \in B$. Isto quer dizer que, para todo $x \in F$, se $x \in Q^{uv}$ então $x \in Q^{ab}$ ou $x \in Q^{fd}$. Portanto, o problema definido em Q^{uv} seria resolvido desnecessariamente já que $F \cap Q^{uv} \subset Q^{ab} \cup Q^{fd}$. Para B minimal, existirá solução viável x tal que $x_{uv} = 0$ e $x_{ab} = 1, \forall (a, b) \in B$, ou seja, $x \in Q^{uv}, \forall (u, v) \in B$, e $x \notin Q^{ab}, \forall (a, b) \in B \setminus \{(u, v)\}$.

O procedimento ADD_DISJ começa resolvendo a relaxação TP, sem qualquer restrição adicional. Se a solução ótima correspondente x^* satisfaz a todas as restrições de Capacidade, seu valor associado L_{TP} não poderá ser incrementado (x^* é solução ótima para o ACVRP, uma vez que é inteira, e L_{TP} é valor ótimo correspondente). Caso contrário, tenta-se melhorá-lo utilizando a disjunção em um subconjunto inviável de arcos $B \subset A$ escolhido convenientemente (como veremos a seguir). Desta forma, definimos L_{TP} como limite inferior inicial do procedimento aditivo ADD_DISJ, o custo residual atual $\overline{c}$, correspondente a L_{TP} e o conjunto $A^* = \{(i, j) \in A; x^*_{ij} = 1\}$ associado a solução x^*. Tomaremos B como um subconjunto de A^* que corresponderá a um caminho da forma $\{(P_1, P_2), \ldots, (P_r, P_{r+1})\}$ e estará satisfazendo a um dos itens abaixo (considere $V(B) = \{P_1, P_2, \ldots, P_{r+1}\}$ o conjunto de nós cobertos por B):

(I) $P_1 = P_{r+1}$ e $0 \notin V(B)$, ou seja, B corresponde a um caminho fechado (circuito) desconexo do nó depósito;

(II) $\sum_{j \in v(B)} d_j > C$, ou seja, B corresponde a um caminho cujo a demanda total dos clientes visitados por ele excede a capacidade do veículo;

(III) $P_1 = P_{r+1}$, $0 \in V(B)$ e $r(V \setminus V(B)) > k - 1$, ou seja, B corresponde a um circuito (passando pelo nó depósito) cujo conjunto de nós, não cobertos por ele, não pode ser servido pelos $k - 1$ veículos restantes.

Então, enquanto existir conjuntos $B \in A^*$ como definidos acima, computamos o limite inferior disjuntivo L_D, tomando o B de menor cardinalidade, para acrescentar ao limite inferior aditivo L_{TP}. Para isso, para cada $(a, b) \in B$, resolvemos uma relaxação de TP^{ab} utilizando como entrada o custo residual atual $\overline{c}$. Serão gerados $\overline{x}^{ab}$, υ^{ab} e γ^{ab} referentes ao vetor de solução ótima, valor da solução ótima e o vetor de custos residuais, respectivamente. Se $\upsilon^{st} = L_D = \min\{\upsilon^{ab} : (a, b) \in B\}, (s, t) \in B$, então define-se $x^* = x^{st}$ e computa o custo residual $\overline{c}$ (que será o custo residual atual), associado a L_D, conforme já visto anteriormente. O limite inferior aditivo L_{TP} é então, atualizado fazendo-se $L_{TP} = L_{TP} + L_D$. Atualiza-se também $A^* = A^* \cap \{(i, j) \in A : x_{ij} = 1\}$. O procedimento termina quando A^* não mais contiver tais subconjunto $B \subset A^*$. Repare que A^* é atualizado através da remoção dos arcos (i, j) com $x^*_{ij} = 0$, além do arco (s, t) tal que $x^* = x^{st}$. Além disso, a escolha de arcos para A^* cujos valores na solução são todos 1 é justificada pelo fato de que impor $x_{uv} = 0$, para algum $(u, v) \in A$ tal que $x^*_{uv} = 0$, levará a um limite inferior $\upsilon^{uv} = L_{TP}$.

Repare que devido a minimalidade de B, o item (II) implica que $0 \notin V(B)$ e $P_1 \neq P_{r+1}$. Veja que se $0 \in V(B)$ então, se retiramos o arco $(0, P_2)$ (ou $(P_r, 0)$) incidente a ele continuamos não tendo solução viável, contendo todos os arcos restantes, uma vez que $V(B \setminus \{(0, P_2)\})$=$V(B)$=$V(B \setminus \{(P_r, 0)\})$. Além disso, se $P_1 = P_{r+1}$ (circuito), quando retiramos qualquer dos arcos de B ainda não

teremos solução viável contendo os arcos restantes, uma vez que $V(B \setminus \{(\alpha_i, \beta_i)\}) = V(B), \forall (\alpha_i, \beta_i) \in B$. Os dois casos entram em contradição com o fato de B ser minimal já que continuaremos tendo $\sum_{j \in v(B)} d_j > C$. Portanto, o conjunto B corresponderá a apenas um dos itens acima.

Daremos, a seguir, um exemplo com os primeiros passos do procedimento ADD_DISJ. Na instância do ACVRP considerada, a quantidade de nós é 10, a demanda dos nós clientes é unitária, a frota é composta de 3 veículos de capacidade 4 e a matriz de custos é c. Suponha que a solução x^* do TP, sem restrições adicionais, com valor L_{TP} (que também será considerado limite inferior aditivo), é como na figura abaixo. O conjunto de arcos correspondente a solução x^* é $A^* = \{(1,2), (2,3), (3,1), (0,8), (8,5), (5,10), (10,0) (0,4), (4,9), (9,0), (7,6), (6,7), (0,11), (11,12), (12,0)\}$. Seja $B \subset A^*$ o conjunto minimal inviável de arcos.

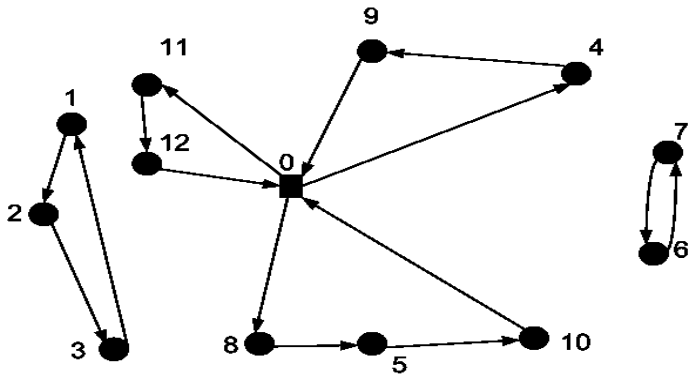

Figura 4.4: Solução viável para o ACVRP

Repare que B pode correponder ao caminho 1-2-3-1 e 7-6-7, ambos satisfazendo o item (I). Escolhemos o caminho de menor cardinalidade que é o segundo mencionado. Então, $B = \{(7,6)(6,7)\} \subset A^*$. Podemos, então, tentar melhorar o limite inferior L_{TP} criando dois subproblemas, por acrescentar a restrição $x_{76} = 0$ em um e $x_{67} = 0$ no outro, e resolvendo-os afim de determinar o limite inferior disjuntivo

L_D para acrescentar ao limite inferior aditivo L_{TP}, obtendo assim um limite inferior melhor para o problema ACVRP.

- Estudaremos agora o limite inferior baseado na relaxação de fluxo de custo mínimo descrito em [19].

  Neste procedimento de *bounding*, vamos considerar uma partição $\{S_0, \ldots, S_m\}$ do conjunto de nós clientes V, com $0 \in S_0$, e definir

$$A_1 = \bigcup_{h=0}^{m} \{(i,j) \in A : i,j \in S_h\}$$

$$A_2 = A \setminus A_1$$

  Em outras palavras, o conjunto de arcos A é particionado em dois subconjuntos $\{A_1, A_2\}$, onde A_1 possui todos os arcos cujos nós extremos pertencem ao mesmo conjunto $S_h, h = 0, \ldots, m$ e A_2, todos aqueles cujos nós extremos pertencem a conjuntos S_h diferentes.

  Serão definidos, então, dois problemas da forma $\nu(FR^t) = \min\{c_t x : x \in F^t\}$, onde F^t é a projeção do conjunto de soluções F do ACVRP em $\mathbb{R}^{A_t}, t = 1, 2$, ou seja, F^t é o conjunto de todos os vetores de dimensão $|A_t|$ obtidos dos vetores $x \in F$ através da remoção das componentes associadas aos arcos $(i,j) \in A \setminus A_t$. De forma análoga, c_t é a projeção de c em $\mathbb{R}^{A_t}, t = 1, 2$, e isto quer dizer que c_t é um vetor de dimensão $|A_t|$, obtido com a remoção dos elementos associados aos arcos de $A \setminus A_t$. A cada problema FR^t corresponderá um limite inferior v^t e um vetor custo residual $\gamma^t \in \mathbb{R}^{A_t}$, que serão obtidos da forma descrita no algoritmo a seguir.

  O limite inferior (aditivo) L_p para o ACVRP, será dado por

$$L_p = v^1 + v^2,$$

e será chamado de limite inferior projetivo. Os custos residuais correspondentes serão definidos como:

$$\overline{c}_{ij} = \begin{cases} \gamma_{ij}^1 & \text{para} \quad (i,j) \in A_1, \\ \gamma_{ij}^2 & \text{para} \quad (i,j) \in A_2. \end{cases}$$

Não é dificil mostrar que L_p e $\overline{c}$ computados desta forma satisfazem a 4.1.

Em seu algoritmo, [19] considerou desprezada a contribuição dos arcos internos de A_1 para o cálculo de L_p, ou seja, $v^1 = 0$ e, portanto, $L_p = v^2$. Este valor de v^1 é um limite inferior válido para FR_1 uma vez que é assumido que $c_{ij} \geq 0, \forall (i,j) \in A$. Os custos residuais correspondentes a v^1 serão $\gamma_{ij}^1 = c_{ij}, (i,j) \in A_1$, obviamente válidos para FR^1. Com respeito ao conjunto de arcos A_2, o limite inferior v^2 e os custos reziduais, γ^2, correspondentes serão computados através da resolução da seguinte relaxação do ACVRP:

$$(RF) \qquad min \sum_{a \in A_2} c_a x_a \tag{5.1}$$

Sujeito a:

$$\sum_{i \in V : a=(i,j) \in A_2} x_a \leq 1, \forall j \in V_0, \tag{5.2}$$

$$\sum_{j \in V : a=(i,j) \in A_2} x_a \leq 1, \forall i \in V_0, \tag{5.3}$$

$$\sum_{j \in V : a=(0,j) \in A_2} x_a \leq k, \tag{5.4}$$

$$\sum_{i \in V : a=(i,0) \in A_2} x_a \leq k, \tag{5.5}$$

$$\sum_{a \in \delta^-(S_h)} x_a \geq \begin{cases} r(V \setminus S_h), & h = 0 \\ r(S_h), & \forall h = 1, \ldots, m \end{cases}, \tag{5.6}$$

$$x_a \in \{0,1\}, \forall a \in A_2, \tag{5.7}$$

A relaxação acima foi obtida da formulação do ACVRP, dada em 2.1.2, através do enfraquecimento de parte das desigualdades de grau, para levar em conta a

remoção dos arcos de A_1, e também considerando as restrições de grau apenas para os subconjuntos de clientes $S_h, h = 1, \ldots, m$, que também leva em conta a remoção dos arcos de A_1 uma vez que todo arco que "entra"em um conjunto $S_h, h = 1, \ldots, m$ pertence a A_2. Este modelo pode ser resolvido de forma eficiente, uma vez que este pode ser visto como um problema de fluxo de custo mínimo na rede auxiliar descrita abaixo:

A referida rede possui 2(n+m+1) nós que são:

- Os nós i^+ e i^-, $\forall i \in V$;
- Os nós a_h e b_h, $\forall h = 1, \ldots, m$;
- Os nós origem s e destino t.

Os arcos da rede, seus custos e capacidades associados são:

(i) Para todo $(i, j) \in A_2$, os arcos (i^+, j^-) com custo c_{ij} e capacidade 1;

(ii) Para todo $h = 1, \ldots, m$, os arcos (a_h, i^+) e (i^-, b_h) para todo $i \in S_h$, com custo 0 e capacidade 1 (se $i \neq 0$) ou k (se $i = 0$);

(iii) Para todo $h = 1, \ldots, m$, os arcos (a_h, b_h) com custo 0 e capacidade $|S_h| - r(S_h)$ (se $h \neq 0$) ou $|S_0| + k - 1 - r(V \setminus S_0)$ (se $h = 0$);

(iv) Para todo $h = 1, \ldots, m$, os arcos (s, a_h) e (b_h, t), ambos com custo 0 e capacidade $|S_h|$ (se $h \neq 0$) ou $|S_0| + k - 1$ (se $h = 0$)

Determinar o fluxo $s - t$ de valor $n + k - 1$ com custo mínimo, na rede construída acima, resolve o problema RF. Para verificar este fato, repare que qualquer fluxo valendo $n + k - 1$, saindo de S com destino a t, satura todos os arcos (s, a_h) e (b_h, t), $h = 0, \ldots, m$. Então, sendo f_h e x_{ij} os fluxos nos arcos (a_h, b_h) e (i^+, j^-),

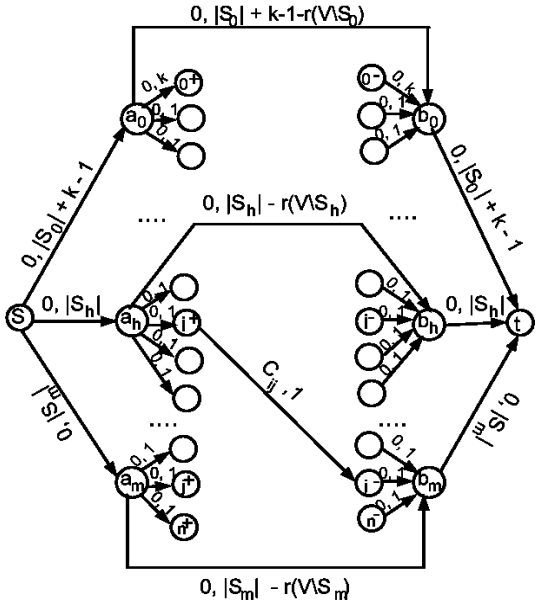

Figura 4.5: Rede associada ao problema RF

respectivamente, teremos que as restrições de grau (5.2)-(5.5) estão sendo satisfeitas por (x_{ij}) devido a conservação de fluxo nos nós i^+ e i^-. Além disso, da conservação do fluxo em a_h e b_h, temos que $\sum_{a \in \delta^-(S_h)} x_a + f_h = |S_h| = \sum_{a \in \delta^+(S_h)} x_a + f_h$. Mas como f_h não pode superar a capacidade do arco (a_h, b_h), pelo menos a quantidade $r(S_h)$ ($r(V \setminus S_0)$ para o conjunto S_0) de fluxo passa pelos nós i^+ e i^-, $i \in S_h$, ou seja, teremos que as restrições de capacidade (5.6) estão sendo satisfeitas. Como uma outra conseqüência da sconservação de fluxo de a_h e b_h é o fato de que $\sum_{a \in \delta^-(S_h)} x_a = \sum_{a \in \delta^+(S_h)} x_a$ e, portanto, a mesma quantidade de arcos que entra em S_h, sai dele (satisfaz a equação 2.7).

De acordo com a correpondência definida em (i)-(iv), os custos residuais γ^2_{ij}, $(i,j) \in A_2$ associados com o valor ótimo v^2 da relaxação RF, coincide com os custos reduzidos de PL associados com os arcos (i^+, j_-) da rede, retornados pelo procedimento de fluxo de custo mínimo.

Analogamente ao procedimento anterior, para cada partição do conjunto V ($\{S_0, \ldots, S_m\}$) teremos um limite inferior distinto. Portanto, em [19] um procedimento aditivo, denominado ADD_FLOW, é utilizado considerando diferentes partições de V, em seqüência, visando obter um limite inferior total melhor. Durante a descrição do ADD_FLOW, será determinada a forma com que as partições são escolhidas. Repare que escolhida a partição $S_h = h, \forall h \in V$ obtemos que a relaxação RF coincide com a relaxação do problema de transporte (TP).

O procedimento ADD_FLOW inicia com a partição $S_h = \{h\}, \forall h \in V \quad (m = n)$ (o custo inicial é c). Em cada iteração do ADD_FLOW, será resolvida a relaxação de RF, correspondente a partição atual, utilizando, como custo, o custo residual atual $\overline{c}$, obtendo solução x^*, o vetor de custos residuais γ^2 e um limite inferior υ^2, que coincide com o limite projetivo L_p e será adicionado ao limite inferior aditivo atual L_{TP}. O custo residual $\overline{c}$ atual será tal que $\overline{c}_{ij} = \gamma^2_{ij}, (i,j) \in A_2$ e $\overline{c}_{ij} = \gamma^1_{ij}, (i,j) \in A_1$. Procura-se então uma coleção $S_{h_1}, \ldots, S_{h_r}, (r \geq 2)$ de subconjuntos da partição atual, tal que a união S^* destes subconjuntos corresponde a uma restrição de capacidade violada. Atualizamos a partição atual através da substituição destes subconjuntos por S^*. O ADD_FLOW termina quando $m = 1$ ou nenhum S^* for encontrado. Veja que, para a partição inicial, FR é o problema dde transporte (TP) e então, sua solução x^* será inteira. Caso satisfaça as restrições de Capacidade, o valor associado a x^*, L_{TP} será ótima para o ACVRP. Neste caso, o limite inferior $L_{TP} = \upsilon^2$ produzido não poderá ser incrementado. Caso contrário, busca-se melhorá-la da forma desrita acima.

A próxima figura 4.6 ilustra um exemplo de uma solução da instância do ACVRP que possui 13 nós, 2 veículos de capacidade 10, e vetor demanda $(d_{ij}) = \{0, 1, 1, 1, 1, 3, 2, 2, 1, 1, 3, 1, 2\}$.

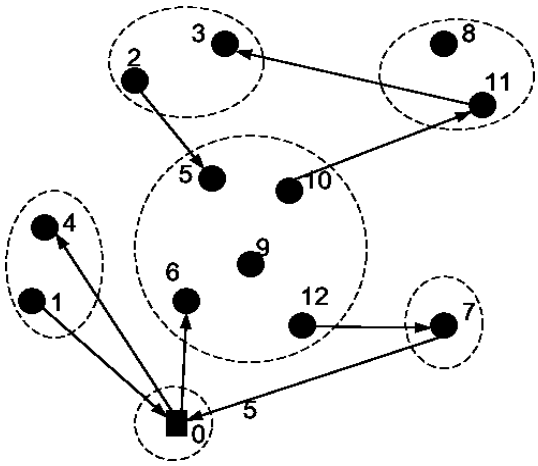

Figura 4.6: Solução de uma instância do ACVRP

Suponha que, em uma determinada iteração, a partição atual seja dada por m=6, $S_0 = \{0\}$, $S_1 = \{1, 4\}$, $S_2 = \{2, 3\}$, $S_3 = \{8, 11\}$, $S_4 = \{5, 6, 9, 10, 12\}$, $S_5 = \{7\}$. Os limites inferiores com respeito a $r(V \setminus S_0)$ e a $r(S_h), h = 1, \ldots, 5$, são $r(V \setminus S_0) = 2$, $r(S_1) = r(S_2) = r(S_3) = r(S_5) = 1$, $r(S_4) = 2$. A solução da relaxação RF com respeito a partição acima é x^* com $x^*_{ij} = 1$, para $(i, j) \in A^*_2 = \{(0, 4), (1, 0), (0, 6), (10, 1), (11, 3), (2, 5), (12, 7), (7, 0)\}$ e $x^*_{ij} = 0$, para todos os arcos de $A_2 \setminus A^*_2$. Veja que para os conjuntos $V \setminus S_h$ e $S_h, h = 1, \ldots, 5$, as desigualdades de capacidade não são violadas.

O *Branch-And-Bound* proposto por Fischetti, Toth e Vigo [19] funciona da seguinte forma: em cada nó da árvore de enumeração, o limite inferior é computado utilizando-se, em seqüência, os dois procedimentos de *bounding* descritos acima, segundo o procedimento aditivo. O ADD_DISJ é o primeiro a ser executado e retornará a última solução x^* obtida, o vetor de custos residuais final $\overline{c}$ e o limite inferior total final L_{TP}, que serão utilizados na inicialização do ADD_FLOW. Ou seja, este procedimento terá, de início, o limite inferior aditivo com valor L_{TP} e custo residual atual $\overline{c}$. Desta forma, a

resolução do primeiro RF não será necessária e o x^* retornado pelo ADD_DISJ será utilizado como solução inicial. A partição utilizada no começo do procedimento será $S_h = \{h\}, h = 1, \ldots, m$ e, então, procura-se uma coleção de subconjuntos da mesma tal que a união deles induz desigualdade violada pela solução inicial x^*. O ADD_FLOW, por sua vez, retornará a última solução x^* gerada e o limite inferior total final L_{TP} para o subproblema atual.

Caso a solução x^* gerada no procedimento de *bounding* seja inviável, para o subproblema atual, devemos eliminá-la da mesma forma que na regra 1 do procedimento da subseção 4.2.1, descrita no procedimento de *Branching* de [27]. A diferença é que aqui o conjunto T estará correspondendo a caminhos satisfazendo aos itens (I)-(III), definidos na página 36, contendo número mínimo de arcos não impostos. Além disso, repare que, no procedimento aditivo a função objetivo do problema é alterada, então uma solução ótima de alguma relaxação resolvida, durante este procedimento, que seja viável para o subproblema atual, não necessariamente é ótima para ele, a menos que o referido problema seja o primeiro TP resolvido (note que o vetor de custos deste problema é o original). Portanto, se x^* é viável para o subproblema atual mas não é ótima, escolhe-se T associado a uma rota viável com o menor número de arcos não impostos no subproblema atual h. Neste caso, mais um subproblema é gerado onde $I_{M+1} = I_h \cup T$ e $F_{M+1} = F_h$. A estratégia de enumeração utilizada neste algoritmo foi busca em largura. Este algoritmo, procura atualizar a melhor solução viável, após a fase de *Bounding*, caso o atual nó de *Branching* não for podado. Esta atualização é feita através do algoritmo heurístico de Vigo [43] e vai utilizar como entrada a solução x^* retornada pelo ADD_DISJ, bem como sua matriz de custos residuais correspondente $\overline{c}$.

Capítulo 5

Um Algoritmo Branch-And-Cut para o ACVRP

Apresentaremos, neste capítulo, as particularidades do nosso algoritmo *Branch-And-Cut*, para a resolução do ACVRP com a formulação dada em 2.1.2, utilizando como base o *Branch-And-Cut* descrito na seção 3.3 para algoritmos inteiros gerais.

No primeiro nó do algoritmo, começamos resolvendo a relaxação de Programação linear do ACVRP sem qualquer restrição de capacidade (Problema de Transporte). É obtida uma solução $\overline{x}$ que, se for inteira e satisfizer a todas as Desigualdades de Capacidade Assimétricas (DCA), é ótima para o ACVRP e, então, não há mais o que fazer. Se a solução é inviável, utilizamos a Heurística das Componentes Conexas, tratada no próximo capítulo, para procurar por DCA violadas por esta solução. Todas as restrições encontradas são, então, incorporadas à formulação e o problema resultante é reotimizado. Repare que, como o problema de transporte possui matriz de restrições unimodular, a primeira solução $\overline{x}$ será inteira e, portanto, nosso algoritmo de separação irá, garantidamente, encontrar todas as DCAs violadas por esta solução, que estejam associadas a rotas excedendo a capacidade do veículo e a subrotas (veremos com detalhes no próximo

capítulo). As etapas, que consistem em resolver a relaxação do problema com sua formulação atual, adicionar cortes que eliminem sua solução $\overline{x}$ e reotimizar o problema com a formulação resultante, são executadas até que nenhuma DCAs seja encontrada pelo nosso procedimento de separação ou até que a solução encontrada $\overline{x}$ seja inteira e satisfaça a todas as DCAs. No primeiro caso, o valor correspondente à última solução $\overline{x}$ é um limite inferior para o ACVRP. Criamos, então, dois subproblemas através do *Branching*, que, neste algoritmo, irá selecionar uma única variável com valor fracionário próximo de 0.5 e maior valor absoluto do coeficiente da função objetivo. No segundo caso, teremos resolvido o ACVRP.

Veja que, após a inserção de alguma DCA, não há mais garantia de que a solução obtida pela resolução do PL atual seja inteira (a matriz de restrições não é mais unimodular) e então não podemos mais garantir encontrar DCA violada no caso em que $\overline{x}$ é inviável.

Para os demais nós da árvore de enumeração, os passos são os mesmos que os realizados no nó raiz, porém, em nenhum momento poderemos garantir que alguma DCA será encontrada, uma vez que na primeira relaxação de PL resolvida teremos a presença de DCAs em sua formulação e, portanto, a matriz de restrições não será unimodular (solução não necessariamente será inteira). A estratégia de busca na árvore de enumeração utilizada é a que chamamos *Dive-And-Best*, que irá realizar uma busca em profundidade e após encontrar uma solução viável para o problema, seguirá com a estratégia do "melhor primeiro"(que irá minimizar a quantidade total de nós gerados na árvore). Executar primeiro uma busca em profundidade tem como objetivo encontrar solução viável rapidamente para que possamos eliminar nós abertos da árvore, que garantidamente não conterão a solução ótima (ver capítulo 3).

Para exemplificar o funcionamento deste algoritmo, utilizaremos uma instância pe-

quena, resolvida pela sua implementação. Esta instância possui 6 nós, com demanda unitária associada aos nós clientes, e uma frota composta de 3 veículos de capacidade 2. Aqui o depósito será representado pelo nó 5. A matriz de custos é:

$$c = \begin{bmatrix} 10000 & 10 & 20 & 30 & 20 & 20 \\ 12 & 10000 & 65 & 54 & 10 & 43 \\ 12 & 30 & 10000 & 10 & 10 & 15 \\ 18 & 9 & 15 & 10000 & 24 & 25 \\ 20 & 40 & 65 & 20 & 10000 & 23 \\ 34 & 20 & 17 & 23 & 25 & 10000 \end{bmatrix}$$

A formulação a seguir é relativa a primeira relaxação de PL a ser resolvida, onde $V_0 = \{0, 1, 2, 3, 4\}$.

$$\begin{aligned} \text{(PL)} \qquad & min \sum_{a \in A} c_a x_a \\ & \text{sujeito a:} \\ & \sum_{a \in \delta^-(i)} x_a = 1, \forall i \in V_0, \\ & \sum_{a \in \delta^+(i)} x_a = 1, \forall i \in V_0, \\ & \sum_{a \in \delta^-(n)} x_a = 3, \\ & \sum_{a \in \delta^+(n)} x_a = 3, \\ & 0 \leq x_a \leq 1, \forall a \in A, \end{aligned}$$

A figura 5.1 abaixo mostra a sua solução, que possui os seguintes valores das variáveis: $\overline{x}_{53} = \overline{x}_{31} = \overline{x}_{10} = \overline{x}_{05} = \overline{x}_{54} = \overline{x}_{45} = \overline{x}_{52} = \overline{x}_{25} = 1$ e $\overline{x}_a = 0$ para todos os outros arcos $a \in A$.

Repare que, esta solução é inviável (a rota 3-1-0 excede a capacidade do veículo) e então devemos utilizar a heurística das componentes conexas para identificar DCAs

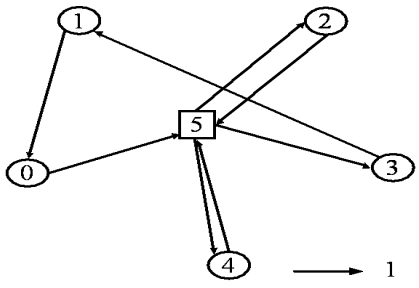

Figura 5.1: Solução inviável para o problema

violadas. Como já foi mencionado, na primeira iteração do *Branch-And-Cut*, todas as DCAs violadas associadas a rotas excedendo a capacidade do veículo e a subrotas serão encontradas. E portanto, neste caso, a DCA correspondente a rota 3-1-0 será encontrada e inserida na formulação do problema, resultando na formulação da relaxação de PL abaixo.

$$
\begin{aligned}
\text{(PL)} \qquad & min \sum_{a \in A} c_a x_a \\
& \text{sujeito a:} \\
& \sum_{a \in \delta^-(i)} x_a = 1, \forall i \in V_0, \\
& \sum_{a \in \delta^+(i)} x_a = 1, \forall i \in V_0, \\
& \sum_{a \in \delta^-(n)} x_a = 3, \\
& \sum_{a \in \delta^+(n)} x_a = 3, \\
& x_{20} + x_{40} + x_{50} + x_{21} + x_{41} + x_{51} + x_{23} + x_{43} + x_{53} \geq 2, \\
& 0 \leq x_a \leq 1, \forall a \in A,
\end{aligned}
$$

Na figura 5.2, temos a solução desta nova relaxação. Esta possui os seguintes valores para as variáveis: $\overline{x}_{53} = \overline{x}_{30} = \overline{x}_{05} = \overline{x}_{51} = \overline{x}_{14} = \overline{x}_{45} = \overline{x}_{52} = \overline{x}_{25} = 1$ e $\overline{x}_a = 0$ para todos os outros arcos $a \in A$.

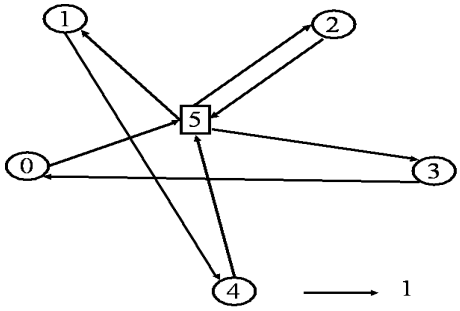

Figura 5.2: Solução ótima para o problema

Veja que esta solução é viável uma vez que não há rotas excedendo a capacidade do veículo ou subrotas. Como estamos resolvendo o nó raiz da árvore de enumeração, esta solução será a ótima para o problema original, o ACVRP, e não precisaremos realizar o *Branching*. O valor da solução ótima é 146. É importante ressaltar que este mesmo problema foi resolvido com um *Branch-And-Bound* puro, gerando 191 nós na árvore de enumeração enquanto que com o nosso *Branch-And-Cut* resolvemos no nó raiz.

Capítulo 6

Algoritmos de Separação

Neste Capítulo, serão relatados três procedimentos de identificação das Desigualdades de Capacidade violadas pela solução inviável do problema linear, que inclui todas as restrições de grau dos nós em sua formulação , podendo conter, inclusive, variáveis fixas (em 0 ou 1) pelo processo de *Branching*, e desigualdades de capacidade, incorporadas ao problema em iterações passadas. Em cada procedimento, faremos um estudo do caso simétrico para então estendê-lo ao caso em questão.

Seja P_{ACVRP} a envoltória convexa das soluções viáveis do ACVRP. Uma vez que tal envoltória constitui um poliedro, esta pode ser descrita por um conjunto finito de desigualdades lineares, e portanto, na teoria, é possível resolver o ACVRP como problema linear neste sistema de desigualdades lineares. Porém, na prática, isto não acontece porque nem todas as desigualdades descrevendo P_{ACVRP} são conhecidas e, além disso, a quantidade de tais desigualdades é muito grande e não podem ser todas incluídas no problema de programação linear. Busca-se, então, uma boa aproximação para P_{ACVRP} nas "redondezas"do ótimo. Daí, dadas formulação inicial para o ACVRP e uma função objetivo, aplica-se um algoritmo de geração de cortes para fazer tal aproximação.

Conforme já visto no capítulo 5, no nosso caso específico, utilizamos, em nosso al-

goritmo, uma fase de Geração de Cortes para gerar as Desigualdades de Capacidade, que já pertencem à formulação por nós utilizada, porém, pelo fato de aparecerem em número exponencial (2^n, onde n é o número de nós clientes), terão que ser incorporadas à medida em que forem sendo necessárias. Esta fase de geração de Cortes será executada em cada subproblema, conforme a descrição do *Branch-And-Cut* nos capítulos 3 e 5, e vai funcionar da seguinte forma:

Em cada iteração, resolve-se um PL, relativo a um determinado subproblema, que conterá as desigualdades mencionadas no primeiro parágrafo, obtendo solução $\overline{x}$ que:

- Se for inteira e satisfizer a todas as DCAs, pára (solução ótima do subproblema atual);
- Se for inviável, utilizamos algoritmo de Separação para encontrar Desigualdades de Capacidade violadas por $\overline{x}$ e incorporamos tais desigualdades à formulação atual;

  Neste caso temos duas possibilidades:

  - $\overline{x}$ é inteira. Daí analisamos outras duas situações;
    - $\overline{x}$ é solução do k-TSP (Existe alguma rota excedendo a capacidade C mas não há subrotas). Ver figura 6.1(a);
    - $\overline{x}$ não é solução do k-TSP (neste caso existe subrota).
      Ver figura 6.1(b);

    Como será visto na próxima seção, em qualquer das duas possibilidades, Desigualdades de Capacidade podem ser facilmente identificadas através de nossa primeira heurística, a heurística das Componentes Conexas.
  - $\overline{x}$ é fracionária. É neste caso que necessita-se concentrar todos os esforços na tentativa de encontrar Desigualdades de Capacidade de forma eficiente.

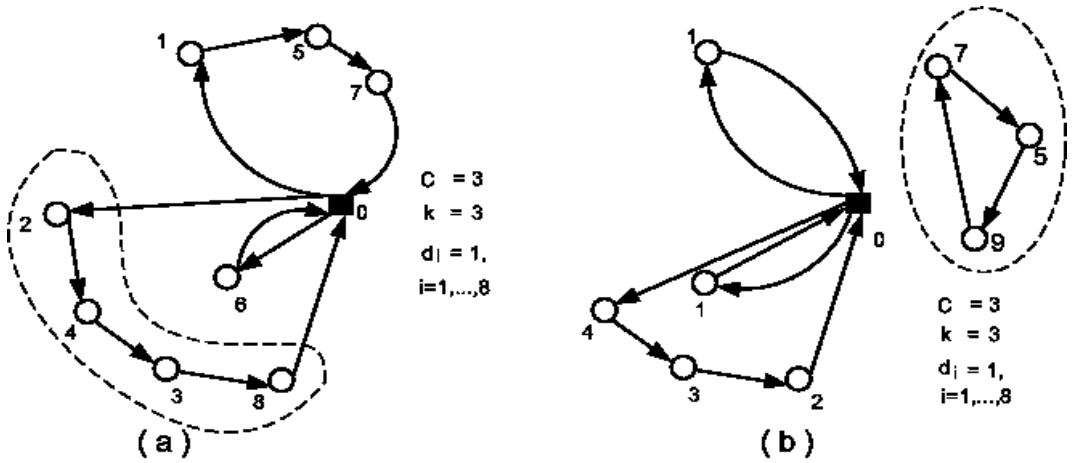

Figura 6.1: Soluções inteiras e inviáveis

Esta fase termina quando não mais encontrar desigualdades violadas ou quando for obtida solução $\overline{x}$, do PL atual, que seja inteira e satisfaça a todas as DCAs.

Então, obtida solução inviável $\overline{x}$ do PL atual, o algoritmo de Separação que apresentaremos vai tentar encontrar Desigualdades de Capacidade violadas. Na verdade, tal algoritmo irá procurar por um conjunto promissor S, contido no conjunto de nós $V \setminus \{0\}$, para o qual $\overline{x}(\delta^-(S))$ e $d(S)$ são computados com o objetivo de verificar se a desigualdade correspondente é violada.

Daí, encontrado $S \subseteq V \setminus \{0\}$, $S \neq \emptyset$, definimos a função $f(S) = x(\delta^-(S)) - \lceil (d(S)/C) \rceil$. E então:

- Se $f(S) < 0$ temos a restrição em questão violada;
- Se $f(S) \geq 0$, $\forall S \subseteq V \setminus \{0\}$, $S \neq \emptyset$ temos que nenhuma de tais restrições é violada.

Portanto, o problema de separação é equivalente a computar

$$min\{f(S); S \subseteq V \setminus \{0\}, S \neq \emptyset\}.$$

O Problema de separação da Desigualdade de Capacidade é NP-difícil para o caso simétrico (ver Harche e Rinaldi [24]) e, como este representa um caso particular do assimétrico, pode-se concluir o mesmo com relação ao Problema de separação da Desigualdade de Capacidade Assimétrica. Pelo que se sabe, ainda não foi encontrada, na literatura, qualquer tentativa de resolução do ACVRP por Geração de cortes.

Nas próximas seções, apresentaremos três procedimentos de separação da Desigualdade de Capacidade Assimétrica. Na primeira, tratamos do algoritmo heurístico das componentes conexas, que será o procedimento implementado. Na segunda seção, estudaremos a heurística de contração (*Shrinking*) e, na seção seguinte, o método exato de separação da desigualdade de Capacidade fracionária utilizando fluxo máximo. Em [31], este método exato serviu de base para um procedimento heurístico de identificação das DCS. Melhores resultados computacionais devem ser obtidos incluindo-se os outros dois procedimentos na ordem em que foram citados, como vem sendo feito no SCVRP.

6.1 Heurística das Componentes Conexas

Nesta seção, apresentaremos a Heurística das Componentes Conexas [31, 38] que possui um procedimento bem simples porém, como veremos, garante encontrar alguma Desigualdade de Capacidade violada, no caso em que a solução do programa linear é inteira e inviável. Neste caso, toda desigualdade encontrada, estará associada a uma rota, excedendo a capacidade do veículo, ou a uma subrota. Chamaremos de grafo suporte $\vec{G}(\overline{x})$ o grafo induzido pelos arcos de G tais que as componentes correspondentes, da solução $\overline{x}$ do PL atual, são estritamente positivas. Então, a heurística das componentes conexas considera cada uma das p ($p \geq 1$) componentes conexas maximais de $\vec{G}(\overline{x})$, obtidas com a retirada do nó depósito e de seus arcos incidentes, com o objetivo de encontrar

subconjuntos de nós clientes definindo DCAs violadas por $\overline{x}$. Como o procedimento desta heurística é o mesmo para os casos simétrico e assimétrico, faremos uma abordagem direta para o caso em que estamos realmente interessados.

Então, ao considerar cada componente conexa maximal c_i, $i = 1, ..., p$, devemos verificar se a DCA está violada para o conjunto de nós C_i, que compõe a componente em questão, ou seja, verificaremos se:

$$f(C_i) = x(\delta^-(C_i)) - \lceil (d(C_i)/C) \rceil < 0 \tag{6.1}$$

Neste caso, teremos que o conjunto C_i correspondente a componente conexa maximal c_i induz Desigualdade de Capacidade Assimétrica violada. Esta desigualdade também será verificada para o conjunto resultante da união de todas as componentes conexas que não possuem conexão com o depósito. As desigualdades resultantes deste procedimento serão inseridas na formulação atual do problema para a resolução do próximo PL.

Para um melhor entendimento deste procedimento, vamos utilizar o seguinte exemplo: um grafo suporte, associado a uma solução inteira inviável, com 10 nós, demanda de 1 para todos os nós clientes e 3 veículos de capacidade 3. A figura 6.2 ilustra o referido grafo, onde todos os arcos possuem valor 1.

Chamando $\lceil (d(C_i)/C) \rceil$ de $r(C_i)$, encontramos os valores $r(C_1) = 1$, $r(C_2) = 2$, $r(C_3) = 1$ e $r(C_4) = 1$. Analisando então as restrições 6.1 , para cada $c_i, i = 1, 2, 3, 4$, obtemos que:

$$f(C_1) = x(\delta^-(C_1)) - r(C_1) = 1 - 1 = 0 \tag{6.2}$$

$$f(C_2) = x(\delta^-(C_2)) - r(C_2) = 1 - 2 < 0 \tag{6.3}$$

$$f(C_3) = x(\delta^-(C_3)) - r(C_3) = 0 - 1 < 0 \tag{6.4}$$

$$f(C_4) = x(\delta^-(C_4)) - r(C_4) = 1 - 1 = 0 \tag{6.5}$$

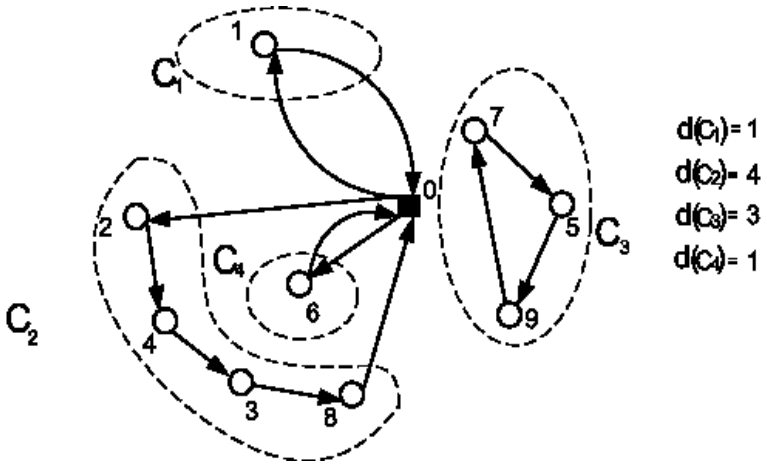

Figura 6.2: Componentes Conexas Maximais do grafo Suporte

Verificamos então que as Desigualdades correspondentes aos conjuntos C_2 e C_3 estão sendo violadas e, portanto, devem ser inseridas no PL para a próxima iteração da fase de geração de cortes do *Branch-And-Cut*.

Cabe salientar que, no caso em que $\overline{x}$ é fracionária, mesmo que existam restrições de Capacidade Assimétricas violadas, esta heurística não garante encontrá-las. Portanto, se nenhuma de tais restrições é identificada através desse procedimento, não poderemos afirmar a não existência das mesmas. Exemplificaremos esta situação com o grafo suporte, associado a uma solução da relaxação de programação linear, de 9 nós, todos com demanda 1, e uma frota de 3 veículos, com capacidade 3 cada. A figura 6.3 ilustra este exemplo, onde os arcos pontilhados indicam que seu peso é 1/2 e os contínuos, 1.

Verificando 6.1 para cada componente conexa $c_i, i = 1, 2, 3$, e S, obtemos:

$$f(C_i) = 0, i = 1, 2, 3 \text{ e } f(S) = 1,5 - 2 < 0$$

Vemos então que os conjuntos de nós associados às componentes conexas maximais do grafo não induzem restrição violada e, no entanto, existe um conjunto de nós S que não está associado a componente conexa alguma mas corresponde à uma desigualdade

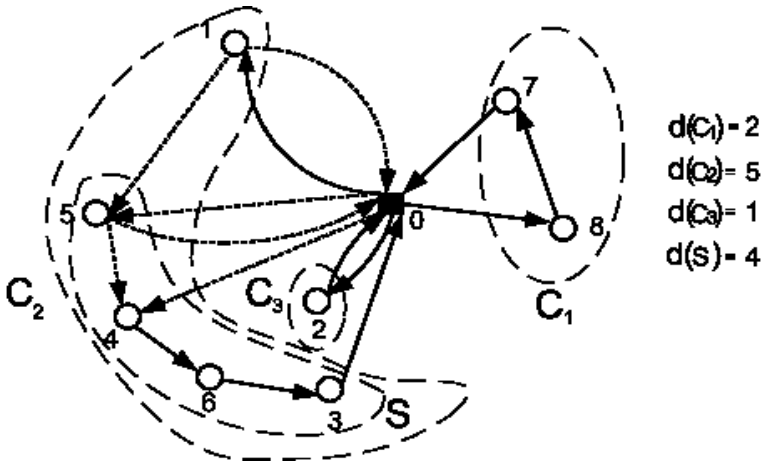

Figura 6.3: Componentes Conexas Maximais do grafo Suporte

de Capacidade violada.

Em nossa implementação da Heurística das Componentes Conexas, o grafo suporte será construído implicitamente através da verificação do valor assumido pelas variáveis, na solução do PL, cada vez que elas são consideradas. Elas serão levadas em conta apenas quando positivas. Para a descrição deste procedimento, o depósito será representado pelo nó n e os clientes, pelos nós $0, \ldots, n-1$.

PROCEDIMENTO: (Heurística das Componentes Conexas)

Entrada: Grafo suporte $\vec{G}(\overline{x})$, da solução $\overline{x}$ do PL, sem o nó depósito (n) e seus arcos incidentes.

Saída: Conjunto R de Desigualdades de Capacidade válidas violadas por $\overline{x}$

Fase 1) Construir as componentes conexas Maximais, $c_1 \ldots, c_p$, de $\vec{G}(\overline{x})/\{n\}$

(1) Defina o vetor v onde cada elemento i representa o nó i do grafo $\vec{G}(\overline{x}) \setminus \{0\}$ e guarda o número associado à componente conexa à qual esse nó pertence. Se o número for 0, este nó não está em componente

conexa alguma. Inicializamos todos os elementos de v e a variável k com zero, onde k é o número associado à componente conexa atual, e vá para o item (2) do fase 1.

(2) Faça $k = k + 1$ e construa a componente conexa c_k buscando o menor i com $v[i]$=0. Atribua o valor k à $v[i]$ e i à componente conexa c_k.
Tome i como nó atual e vá para o item (3) do passo 1 (atual é o nó para o qual todos os seus sucessores serão considerados em (3)). Se nenhum de tais nós existe, todos os nós de $\vec{G}(\overline{x})$ já pertencem a alguma componente conexa e portanto passamos para a Fase 2.

(3) Considere, em ordem crescente de rótulo, todo nó $j \in V \setminus \{n\}$ e verifique se este é sucessor de atual. Caso afirmativo, se $v[j] = 0$ então atribuímos valor k a $v[j]$ e incorporamos j na componente conexa c_k. Senão, se $v[j] = l \neq k$ vamos para o item (4). Quando não existirem mais sucessores de atual, atribui-se a ele o nó da componente conexa c_k que não tenha tido seus sucessores "varridos"(estes nós são tomados na ordem em que foram inseridos em c_k) e repete-se (3).
Se todos os nós de c_k já tiverem sido tomados como nó atual, atribuimos a k o maior número associado a alguma das componentes conexas construídas até então. Voltamos para o item (2).

(4) Seja $q = \max\{l, k\}$. Se $q = k$, fazemos $v[i] = l$ para todos os nós i da componente conexa c_k e movemos todos estes nós para a componente c_l. Neste caso, fazemos $k = l$. Se $q = l$, então fazemos $v[i] = k$ para todos os nós i da componente c_l e movemos todos estes nós para a componente c_k. Nos dois casos, eliminamos a componente conexa c_p e atualizamos o número de cada componente conexa c_t, $t > p$, e o valor $v[i]$, para

todos os seus nós i. Voltamos para o item (3).

Fase 2) Para toda componente conexa $c_i, i = 1, ..., p$, verificar se desigualdade de capacidade definida pelo conjunto de nós correspondentes C_i é violada. Adicionar todas as desigualdades violadas a R.

(1) Percorremos todos os nós de cada componente conexa computando a demanda total das mesmas. Verifico se para c_i atual $x(\delta^-(C_i)) < r(C_i)$. Caso afirmativo, adiciono cada nó de C_i ao conjunto V_j (os conjuntos V_j, $j = 1, ..., m$, $m \leq p$, serão aqueles associados a alguma desigualdade violada). Se além disso, $x(\delta^-(C_i)) = 0$, também, deve-se inserir os nós no conjunto V_0 associado a união das componentes conexas c_i que não possui conexão com o depósito.

(2) Para $j = 1, ..., m$ adiciono $x(\delta^-(V_j)) \geq r(V_j)$ a R. Se houver mais de uma componente conexa sem conexão com o depósito, também insiro o conjunto V_0 em R. Caso contrário, ignoro este conjunto.

Como já foi dito anteriormente, é garantido que este algoritmo encontrará alguma desigualdade de capacidade violada, no caso em que a solução do PL atual é inteira e inviável. Isto ocorre porque, no momento em que retiro o depósito (e todos os arcos incidentes a ele) do grafo $\vec{G}(\overline{x})$, que terá as características daquele da figura 6.1 (a) ou 6.1 (b), todas as componentes conexas obtidas corresponderão a uma rota ou subrota. E então, como em uma solução inteira e inviável haverá desigualdades violadas correspondendo a uma rota com a capacidade do veículo excedida ou a uma subrota, na segunda Fase do algoritmo, encontraremos todas estas desigualdades. Repare que a solução da primeira relaxação de programação linear resolvida no *Branch-And-Cut* (onde todas as restrições de capacidade são relaxadas) será sempre inteira, uma vez que a matriz de restrições do problema de transporte é unimodular. Portanto, se tal solução for inviável já teremos Desigualdade(s) de Capacidade violada(s) que serão incorporadas à formulação.

O problema de transporte é aquele que se obtém ao retirar da formulação do ACVRP todas as restrições de Capacidade.

6.2 Algoritmo Heurístico de Contração

Nesta Heurística produziremos um novo grafo $\vec{G}'(\overline{x})$, denominado Grafo suporte Contraído Assimétrico, a partir do Grafo Suporte $\vec{G}(\overline{x})$ induzido por $\overline{x}$, solução do Problema Linear atual. O Grafo Suporte Contraído produzido é menor que o Grafo Suporte, porém, equivalente a este no sentido de que resolver o Problema de separação no primeiro é equivalente a resolvê-lo no segundo, ou seja, o procedimento não nos fará perder nenhuma Desigualdade de Capacidade violada existente (isto será mostrado para os casos simétricos e assimétricos nas próximas duas seções). Portanto, ao término desta heurística, poderemos aplicar, por exemplo, o procedimento exato, descrito na próxima seção, no grafo $\vec{G}'(\overline{x})$ obtido. Durante procedimento de contração, estaremos gerando desigualdades de capacidade violadas pela solução do PL atual. Estaremos assumindo que este procedimento seja utilizado após a Heurística das Componentes Conexas e portanto, podemos supor que o grafo de entrada ($\vec{G}(\overline{x})$) para a Heurística de Contração é conexo.

Na próxima seção, estudaremos tal algoritmo aplicado ao Problema de Separação da Desigualdade de Capacidade Simétrica [31, 7, 38] e então daremos nossa contribuição, na seção seguinte, através da extensão do mesmo para o Problema de Separação da Desigualdade de Capacidade Assimétrica.

6.2.1 Algoritmo Heurístico de Contração Aplicado ao Problema de Separação da Desigualdade de Capacidade Simétrica

O procedimento do algoritmo de contração é, basicamente, escolher iterativamente um conjunto S de clientes, no Grafo Suporte $G(\overline{x})$, induzido pela solução $\overline{x}$ do problema atual, para ser contraído em um único supernó s. Este supernó possuirá demanda igual à demanda total do conjunto S, $d_s = d(S)$. Para a separação da Desigualdade de Capacidade Simétrica, toda vez que contrairmos um conjunto S em um único supernó s, atribuiremos às arestas $\{s, j\}$ o peso:

$$\overline{x}(E(S : \{j\})) = \sum_{e \in E(S:\{j\})} \overline{x}_e$$

Onde $E(S : \{j\})$ é o conjunto dos arcos que possuem um extremo em S e o outro em $\{j\}$. A figura 6.4 mostra a atribuição de pesos a um supernó s.

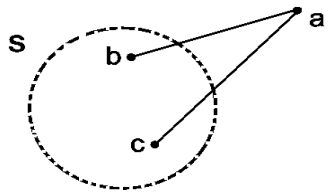

Figura 6.4: A aresta {s, j} possui peso $\sum_{i \in S} \overline{x}_{ij} = \overline{x}_{bj} + \overline{x}_{cj}$

Então, no novo grafo $G^{'}(\overline{x})$, obtido ao término do procedimento de contração, o peso associado a cada aresta incidente a um supernó será aquele definido acima. Para os demais, o peso será o mesmo que no grafo original $G(\overline{x})$.

A partir de agora, no grafo contraído, estaremos também chamando de supernó qualquer nó que não tenha "entrado"numa contração, ou seja, todos os nós do Grafo Contraído serão chamados de supernós. Se em nosso procedimento de contração tomarmos apenas conjuntos S que atendam a certas condições, podemos mostrar que a contração

será segura [31]. Isto quer dizer que sempre que existir uma Desigualdade de Capacidade violada em $G(\overline{x})$, existirá um conjunto de supernós no grafo contraído cuja união de seus nós originais define uma Desigualdade de Capacidade com, pelo menos, a mesma violação. Ou seja,

$$\text{Se} \quad \exists \quad T \quad \text{tal que} \quad \overline{x}(\delta(T)) - 2r(T) < 0 \quad \Longrightarrow$$

$$\exists\{S_1 \dots S_p\}; \quad S_i \cap T \neq \emptyset \quad (S_i \quad \text{conjunto de nós originais correspondentes a} \quad s_i),$$

$$\forall i = 1, \dots, p \quad \text{tal que}$$

$$\overline{x}(\delta(\bigcup_{i=1}^{p} S_i)) - 2r(\bigcup_{i=1}^{p} S_i) \leq \overline{x}(\delta(T)) - 2r(T)$$

Este fato pode ser concluido a partir do próximo resultado.

Proposição 1 *Para separação das desigualdades de capacidade simétricas, é seguro contrair um conjunto de clientes S se $\overline{x}(\delta(S)) \leq 2$ e $\overline{x}(\delta(R)) \geq 2$, $\forall R \subset S$.*

Na demonstração desta proposição faremos uso do próximo resultado:

Definição: *Uma função f é submodular se*

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$$

para $S, T \subseteq N$, N conjunto finito, $f : S \subseteq N \rightarrow \mathbb{R}$.

Lema 2 *A função corte $\overline{x}(\delta(S))$ é submodular,onde*

$$\overline{x}(\delta(S)) = \sum_{e \in \delta(S)} \overline{x}_e$$

Dem(Lema 2):

Vamos demonstrar este resultado para conjuntos T e S tais que $T \cap S$, $T \setminus S$, $S \setminus T \neq \emptyset$. Quando $T \subset S$ ou $S \subset T$ podemos demonstrá-lo de forma análoga.

Sejam:

- S e T subconjuntos de V_0;
- A conjunto de arestas que possuem um nó extremo em $(T \cap S)$ e o outro em $(V_0 \setminus (T \cup S))$;
- B conjunto de arestas que possuem um nó extremo em $(T \setminus S)$ e o outro em $(V_0 \setminus (T \cup S))$;
- C conjunto de arestas que possuem um nó extremo em $(T \setminus S)$ e o outro em $(S \cap T)$;
- D conjunto de arestas que possuem um nó extremo em $(S \setminus T)$ e o outro em $(V_0 \setminus (T \cup S))$;
- E conjunto de arestas que possuem um nó extremo em $(S \setminus T)$ e o outro em $(S \cap T)$;
- F conjunto de arestas que possuem um nó extremo em $(S \setminus T)$ e o outro em $(T \setminus S)$;

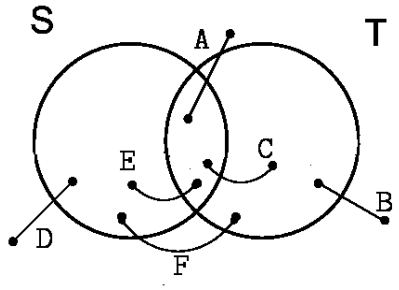

Figura 6.5: Conjuntos definidos acima

Devemos provar que $\overline{x}(\delta(S)) + \overline{x}(\delta(T)) - \overline{x}(\delta(T \cup S)) - \overline{x}(\delta(T \cap S)) \geq 0$

Daí, como

$\overline{x}(\delta(S)) = \overline{x}(D) + \overline{x}(A) + \overline{x}(F) + \overline{x}(C)$

$\overline{x}(\delta(T)) = \overline{x}(B) + \overline{x}(A) + \overline{x}(F) + \overline{x}(E)$

$\overline{x}(\delta(T \cup S)) = \overline{x}(D) + \overline{x}(B) + \overline{x}(A)$

$\overline{x}(\delta(T \cap S)) = \overline{x}(A) + \overline{x}(C) + \overline{x}(E)$

Teremos:

$(\overline{x}(D) + \overline{x}(A) + \overline{x}(F) + \overline{x}(C)) + (\overline{x}(B) + \overline{x}(A) + \overline{x}(F) + \overline{x}(E))$ - $(\overline{x}(D) + \overline{x}(B) + \overline{x}(A))$ -

$(\overline{x}(A) + \overline{x}(C) + \overline{x}(E)) =$

$= \quad 2\overline{x}(F) \quad \geq 0$

Logo, $\overline{x}(\delta(S)) + \overline{x}(\delta(T)) \geq \overline{x}(\delta(T \cup S)) + \overline{x}(\delta(T \cap S))$

E portanto, a função corte é submodular. ■

Dem(Prop. 1):

Seja T um conjunto de nós clientes, violando Desigualdade de Capacidade. Da mesma forma que na demonstração do lema, os conjuntos T e S serão tais que $T \cap S$, $T \setminus S$, $S \setminus T \neq \emptyset$. Os casos em que $T \subset S$ e $S \subset T$ podem ser demonstrados de forma análoga. Devemos mostrar que a Desigualdade de Capacidade em $T \cup S$ é violada, no mínimo, tanto quanto a Desigualdade de capacidade em T, isto é,

$$\overline{x}(\delta(T \cup S)) - 2r(T \cup S) \leq \overline{x}(\delta(T)) - 2r(T) \quad (*)$$

Mostraremos então que

$$\overline{x}(\delta(T)) - \overline{x}(\delta(T \cup S)) + 2r(T \cup S) - 2r(T) \geq 0 \quad (**)$$

É trivial que $\quad 2r(T \cup S) - 2r(T) \geq 0.$

Devemos então mostrar que $\quad \overline{x}(\delta(T)) - \overline{x}(\delta(T \cup S)) \geq 0$

Segue da submodularidade da função corte que:

$$\overline{x}(\delta(T)) - \overline{x}(\delta(T \cup S)) \geq \overline{x}(\delta(T \cap S)) - \overline{x}(\delta(S))$$

Uma vez que pela hipótese, $\overline{x}(\delta(T \cap S)) \geq 2$ e $\overline{x}(\delta(S)) \leq 2$

Temos que $\overline{x}(\delta(T \cap S)) - \overline{x}(\delta(S)) \geq 0$

Portanto, temos (**).

E então , (*) é verdade (a desigualdade de capacidade em $T \cup S$ é violada, no mínimo, tanto quanto a Desigualdade de Capacidade em T). ■

Observe que estamos chamando cada nó do Grafo Contraído de supernó e portanto podemos dizer que $T \cup S$ é união de conjuntos de nós originais associados a supernós ($T \setminus S$ é união de supernós correspondentes a um único nó).

Repare também que se tomarmos outro conjunto W satisfazendo às condições da proposição e tal que $W \cap T$, $W \setminus (T \cup S)$, $(T \cup S) \setminus W \neq \emptyset$ então, já que $T \cup S$ corresponde a DCS violada, obtemos, com raciocínio análogo, que:

$$\overline{x}(\delta(T)) - 2r(T) \geq \overline{x}(\delta(T \cup S)) - 2r(T \cup S) \geq \overline{x}(\delta(W \cup (T \cup S))) - 2r(W \cup (T \cup S))$$

Tudo isso mostra que o conjunto obtido da união de todos os supernós (quando as contrações são feitas conforme proposição 1) que interceptam T define uma Desigualdade de Capacidade violada, no mínimo, pela mesma quantidade que a Desigualdade em T. Então, a contração é **segura** e, desta forma, podemos aplicar qualquer outro procedimento de separação a este grafo contraído, aproveitando o fato de que houve uma redução no tamanho do problema, ao término do processo.

Serão geradas DCS violadas, correspondentes a conjuntos S, contraídos pelo procedimento descrito acima. Veja que se $\overline{x}(\delta(S)) < 2$, então a Desigualdade de Capacidade Simétrica correspondente estará violada.

A teoria vista nesta subseção, generaliza o caso em que todo supernó s, cujo conjunto de nós originais correspondente S contém mais de dois nós, é obtido através da contração de arestas e, não incidentes ao depósito, tais que $\overline{x}_e \geq 1$ [38]. Este procedimento foi

anteriormente utilizado na separação das desigualdades de eliminação de subrotas, do problema do caixeiro viajante [23]. Na heurística de Contração de [7] são contraídos apenas arestas com peso 1, na identificação das Desigualdade de Capacidade Simétrica.

Então, dada uma aresta $e = (i, j)$, $i, j \neq 0$, de $G(\overline{x})$ com $\overline{x}_e \geq 1$, iremos contraí-la em um supernó q de modo que o valor da demanda d_q e de um aresta (q, v) são dados, respectivamente, por:

$$d_q = d_i + d_j \qquad \text{e} \qquad \overline{x}_{qv} = \overline{x}_{iv} + \overline{x}_{jv}$$

A figura 6.6 ilustra o processo de contração das arestas e tais que $\overline{x}_e \geq 1$.

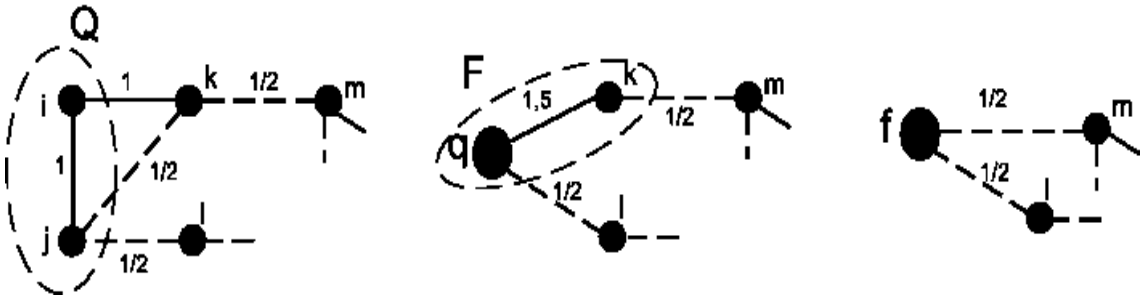

Figura 6.6: contração de arestas e com peso $\overline{x}_e \geq 1$

Cada supernó construído desta forma será constituído de todos os nós extremos das arestas e contraídas. Observe que arestas e com $\overline{x}_e > 1$ são obtidas na contração devido ao peso atribuído às arestas incidentes a um supernó.

O procedimento de contração é aplicado recursivamente até que nenhuma aresta com valor maior ou igual a 1 permaneça.

6.2.2 Algoritmo Heurístico de Contração Aplicado ao Problema de Separação da Desigualdade de Capacidade Assimétrica

Nesta seção, vamos mostrar que o algoritmo de Contração, apresentado na seção anterior, pode ser perfeitamente adaptado para aplicação no Problema de Separação da Desigualdade de Capacidade Assimétrica.

Para tanto, devemos verificar se para aquelas condições da Proposição 1 é **seguro** contrair um conjunto de clientes S do Grafo Suporte orientado $\vec{G}(\overline{x})$. Caso contrário, estabeleceremos condições para as quais isto ocorre.

O procedimento de contração para o nosso problema é semelhante ao já visto na seção anterior: contrairemos, iterativamente, um conjunto S em um supernó s , para o qual a demanda será igual a demanda total de S, ou $d_s = d(S)$. Como o grafo $\vec{G}(\overline{x})$ é orientado, ao realizarmos tal contração, atribuiremos aos arcos (s, j) e (j, s) os pesos:

$$\overline{x}(E(S : \{j\})) = \sum_{(i,j)\in E(S:\{j\})} \overline{x}_{ij} \qquad \text{e} \qquad \overline{x}(E(\{j\} : S)) = \sum_{(j,i)\in E(\{j\}:S)} \overline{x}_{ji}$$

respectivamente, como ilustrado na figura 6.7 .

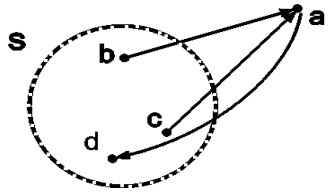

Figura 6.7: Os arcos (s, j) e (j, s) possuem, respectivamente, pesos $\sum_{i\in S} \overline{x}_{ia} = \overline{x}_{bj} + \overline{x}_{cj}$ e $\sum_{i\in S} \overline{x}_{ji} = \overline{x}_{ad}$

Ao fim deste procedimento, obteremos o Grafo Suporte Contraído $\vec{G}'(\overline{x})$ que possuirá pesos, associados aos arcos incidentes a supernós, conforme definidos acima. Quanto aos demais, o peso será o mesmo de $\vec{G}(\overline{x})$.

Veremos agora que, na heurística de contração aplicada ao nosso problema de separação, as condições que garantem contração segura não podem ser as mesmas da seção

anterior. Isto porque a exigência de que a soma dos arcos, que chegam em todo conjunto $R \subset S$ $(\delta^-(R))$, é maior ou igual a 2, entraria em conflito com o fato de que $\overline{x}((\delta^-(R))) = 1$, $\forall R = \{i\}, i \in S$(pelas restrições de Grau). Portanto substituiremos esta parte da hipótese (da proposição 1) por $\overline{x}((\delta^-(R))) \geq 1$. Acontece que, para que possamos enfim garantir que com a contração não perderemos nenhuma Desigualdade de Capacidade Assimétrica violada, caso exista alguma, devemos impor também que $\overline{x}((\delta^-(S))) \leq 1$. E então a versão adaptada da proposição 1 é a seguinte (assim como no caso simétrico, no Grafo Contraído, estaremos também chamando nós, que não "entraram"em uma contração, de supernós):

Proposição 3 *Para separação das desigualdades de capacidade assimétricas, é seguro contrair um conjunto de clientes S, se $\overline{x}(\delta^-(S)) \leq 1$ e $\overline{x}(\delta^-(R)) \geq 1$, $\forall R \subset S$.*

Para sua demonstração, da mesma forma que na seção passada, precisaremos do resultado enunciado a seguir:

Lema 4 *A função corte $\overline{x}(\delta^-(S))$ é submodular,onde*

$$\overline{x}(\delta^-(S)) = \sum_{e \in \delta^-(S)} \overline{x}_e$$

Dem(Lema 4):

Assim como no caso simétrico, provaremos apenas para os conjuntos S e T tais que $T \cup S, T \setminus S, S \setminus T \neq \emptyset$. Para os casos $T \subset S$ e $S \subset T$, a demontração é análoga.

Sejam:

- S e T subconjuntos de V_0;
- A conjunto de arcos que deixam $(V_0 \setminus (T \cup S))$ e chegam em $(T \cap S)$;
- B conjunto de arcos que deixam $(V_0 \setminus (T \cup S))$ e chegam em $(T \setminus S)$;

- C conjunto de arcos que deixam $(V_0 \setminus (T \cup S))$ e chegam em $(S \setminus T)$;
- D conjunto de arcos que deixam $(S \setminus T)$ e chegam em $(T \setminus S)$;
- E conjunto de arcos que deixam $(T \setminus S)$ e chegam em $(S \setminus T)$;
- F conjunto de arcos que deixam $(S \setminus T)$ e chegam em $(S \cap T)$;
- G conjunto de arcos que deixam $(T \setminus S)$ e chegam em $(S \cap T)$;

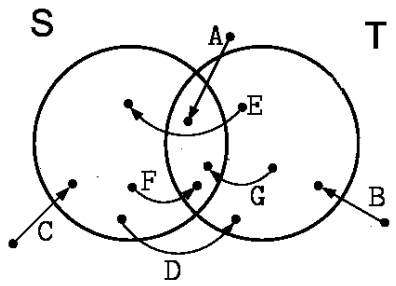

Figura 6.8: Conjuntos de arcos definidos acima

Mostraremos, equivalentemente ao caso Simétrico, que:

$$\overline{x}(\delta^-(S)) + \overline{x}(\delta^-(T)) - \overline{x}(\delta^-(T \cup S)) - \overline{x}(\delta^-(T \cap S)) \geq 0$$

Daí, como

$\overline{x}(\delta^-(S)) = \overline{x}(C) + \overline{x}(A) + \overline{x}(E) + \overline{x}(G)$

$\overline{x}(\delta^-(T)) = \overline{x}(B) + \overline{x}(A) + \overline{x}(D) + \overline{x}(F)$

$\overline{x}(\delta^-(T \cup S)) = \overline{x}(C) + \overline{x}(B) + \overline{x}(A)$

$\overline{x}(\delta^-(T \cap S)) = \overline{x}(A) + \overline{x}(G) + \overline{x}(F)$

Teremos:

$(\overline{x}(C) + \overline{x}(A) + \overline{x}(E) + \overline{x}(G)) + (\overline{x}(B) + \overline{x}(A) + \overline{x}(D) + \overline{x}(F))$ - $(\overline{x}(C) + \overline{x}(B) + \overline{x}(A))$ -

$(\overline{x}(A) + \overline{x}(G) + \overline{x}(F)) =$

$= \quad \overline{x}(E) + \overline{x}(D) \quad \geq 0$

Logo, $\overline{x}(\delta^-(S)) + \overline{x}(\delta^-(T)) \geq \overline{x}(\delta^-(T \cup S)) + \overline{x}(\delta^-(T \cap S))$

E portanto, a função corte é submodular. ■

Dem(Prop. 3):

A demonstração segue exatamente os mesmos passos da demonstração da Proposição 1 e, da mesma forma, somente o caso em que $T \cup S, T \setminus S, S \setminus T \neq \emptyset$, onde T é um conjunto de clientes violando Desigualdade de Capacidade, serão considerados. Os casos em que $T \subset S$ e $S \subset T$ são tratados da mesma forma.

Mostraremos que:

$$\overline{x}(\delta^-(T)) - \overline{x}(\delta^-(T \cup S)) + r(T \cup S) - r(T) \geq 0$$

É trivial que $\quad r(T \cup S) - r(T) \geq 0.$

Devemos, então, mostrar que $\quad \overline{x}(\delta^-(T)) - \overline{x}(\delta^-(T \cup S)) \geq 0$

Segue da submodularidade da função corte que:

$$\overline{x}(\delta^-(T)) - \overline{x}(\delta^-(T \cup S)) \geq \overline{x}(\delta^-(T \cap S)) - \overline{x}(\delta^-(S))$$

Uma vez que pela hipótese, $\overline{x}(\delta^-(T \cap S)) \geq 1$ e $\overline{x}(\delta^-(S)) \leq 1$

Temos que $\overline{x}(\delta^-(T \cap S)) - \overline{x}(\delta^-(S)) \geq 0$

E então , $\overline{x}(\delta^-(T)) - r(T) \geq \overline{x}(\delta^-(T \cup S)) - r(T \cup S)$. ■

E como cada nó que não tenha "entrado"em uma contração também é chamado supernó, podemos dizer que $T \cup S$ é união de conjuntos de nós originais associados a supernós ($T \setminus S$ é união de supernós correspondentes a um único nó).

Se tomarmos conjunto W satisfazendo as condições da proposição e tal que $W \cap T$, $W \setminus (T \cup S)$, $(T \cup S) \setminus W \neq \emptyset$ então, como $(T \cup S)$ corresponde a DCA violada, com

raciocínio análogo, concluimos que:

$$\overline{x}(\delta^-(T)) - r(T) \geq \overline{x}(\delta^-(T \cup S)) - r(T \cup S) \geq \overline{x}(\delta(W \cup (T \cup S))) - r(W \cup (T \cup S))$$

Concluimos então que o conjunto obtido da união de T e todos os supernós (construídos de acordo com a proposição 3) que interceptam T define uma Desigualdade de Capacidade Assimétrica violada, no mínimo, pela mesma quantidade que a Desigualdade em T. Ou seja, é **seguro** contrair sob aquelas condições da proposição 3. E portanto, é possível aplicar qualquer outro procedimento de separação neste novo grafo, que possui a vantagem de ser menor que o grafo original $\vec{G}(\overline{x})$.

Este procedimento irá gerar DCAs que estejam sendo violadas nos conjuntos S, associados a supernós s. Veja que se $\overline{x}(\delta^-(S)) < 1$ então S corresponde a um DCA violada.

De forma análoga ao caso simétrico, consideraremos a situação em que cada supernó q é obtido através da contração de arcos $a = (i, j)$ do grafo $\vec{G}(\overline{x})$, $i, j \neq 0$, tais que $\overline{x}_a = 1$ (veremos que não serão obtidos arcos com valor superior a 1 com a contração). Esta situação é generalizada pela teoria já vista nesta seção, conforme será demonstrado nas proposições 5 e 6.

O valor da demanda d_q e de todos os arcos (q, v) e (u, q) são definidos como:

$$d_q = d_i + d_j, \qquad \overline{x}_{qv} = \overline{x}_{iv} + \overline{x}_{jv} \qquad \text{e} \qquad \overline{x}_{uq} = \overline{x}_{ui} + \overline{x}_{uj}$$

Cada supernó obtido desta maneira terá, em sua composição, todos os nós extremos dos arcos a contraídos. No entanto, diferentemente do caso simétrico, com a contração nunca obteremos novos arcos a tais que $\overline{x}_a = 1$, ou com valor superior a 1, e então, os arcos contraídos serão somente aqueles que já possuiam valor 1 no início do procedimento. Conforme será demonstrado na proposição 5, isto se deve à orientação dos arcos e às equações de grau dos nós clientes. Antes, porém, daremos a seguinte definição.

Definição: Um caminho é uma seqüência de nós $(\eta_1, \ldots, \eta_q)$ ligados dois a dois por arcos. Se $\eta_1 \neq \eta_q$ então este caminho é chamado caminho aberto. Ver exemplo na figura 6.9.

Proposição 5 *Nenhum arco a tal que* $\overline{x}_a = 1$ *ou* $\overline{x}_a > 1$ *será obtido com o procedimento de contração de arcos com valor unitário. Além disso, nesta contração, todo supernó s será tal que* $\overline{x}(\delta^-(S)) = 1$, *S conjunto de nós originais.*

Dem:

Suponha que o arco $a \in A$, com origem i e destino j, é tal que $\overline{x}_a = 1$. Então, este será contraído em um supernó q. Pela definição, os pesos dos arcos (q, v) e (u, q) serão dados por:

$$\overline{x}_{qv} = \overline{x}_{iv} + \overline{x}_{jv} \qquad \text{e} \qquad \overline{x}_{uq} = \overline{x}_{ui} + \overline{x}_{uj}$$

Porém, pelas restrições de grau dos nós, não pode haver outro arco saindo de i uma vez que já tenho arco (i, j) valendo 1. Logo, $\overline{x}_{iv} = 0$ e, portanto, $\overline{x}_{qv} = \overline{x}_{jv}$. Por outro lado, chegando em j já temos o arco (i, j) valendo 1 e, então $\overline{x}_{uq} = \overline{x}_{ui}$ $(\overline{x}_{uj} = 0)$.

Suponhamos que haja arco incidente a um supernó q com valor 1, digamos (q, v). Então, da mesma forma, iremos contraí-lo em um supernó f, obtendo arcos (f, l) e (m, f) com pesos:

$$\overline{x}_{fl} = \overline{x}_{ql} + \overline{x}_{vl} \qquad \text{e} \qquad \overline{x}_{mf} = \overline{x}_{mq} + \overline{x}_{mv}$$

Saindo de q (de j) já temos o arco (j, v) valendo 1 e então $\overline{x}_{fl} = \overline{x}_{vl}$ $(\overline{x}_{ql} = 0)$ e, da mesma forma, $\overline{x}_{mf} = \overline{x}_{mq}$.

A partir daí, podemos ver que todo arco incidente a um supernó s, obtido com a contração de arcos com peso originalmente 1, será incidente ao nó origem ou ao destino do caminho aberto (formado por todos os nós originais de s ligados dois a dois por arcos de peso 1 contraídos em s). Isto ocorre porque todo nó intermediário deste caminho já possui um

arco chegando e um saindo dele, ambos com valor 1. Como em um dos nós extremos só existem arcos saindo dele (para nós externos a s) e no outro somente chegando (de nós externos a s) e , além disso, tais arcos valem no máximo 1 (pelas restrições de grau), não será possível obter novos arcos com valor 1 ou maior que 1. E além disso, $\overline{x}(\delta^-(S)) = 1$, S conjunto de nós originais associados a s. ■

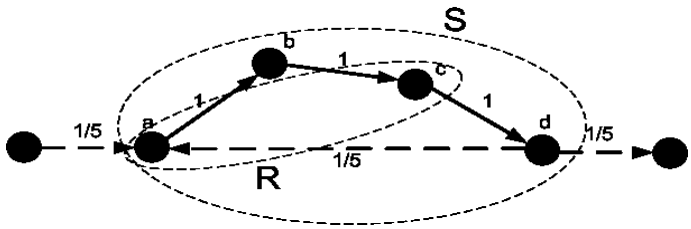

Figura 6.9: Caminho aberto(a,b,c,d) e conjuntos de nós S e R.

Proposição 6 *Se todo supernó é obtido através da contração de arcos com peso 1, então* $\forall R \subset S$, *S conjunto de nós originais correspondentes a s, temos que* $\overline{x}(\delta^-(R)) \geq 1$.

Dem:

O início desta demonstração é idêntica a da proposição 5. Daí se R é um conjunto constituído de nós consecutivos no caminho, formado por todos os nós originais de s ligados dois a dois por arcos de peso 1 contraídos em s, então, $\overline{x}(\delta^-(S)) = 1$. Se R não contém apenas nós consecutivos então, $\overline{x}(\delta^-(R)) \geq 1$. ■

Então, das proposições 5 e 6, obtemos que este procedimento de contração de arcos com peso 1 é um caso particular da teoria desta seção e então, a contração é segura. Este procedimento de contração é aplicado recursivamente até que nenhum arco com valor 1 permaneça. No exemplo da figura 6.9, $\overline{x}(\delta^-(S)) = 1/2$ e $\overline{x}(\delta^-(R)) = 2$.

6.3 Procedimento Exato Aplicado na Separação da Desigualdade de Capacidade Fracionária

O problema de separação da desigualdade de Capacidade Fracionária Assimétrica (DCFA) pode ser resolvido, em tempo polinomial, através da tranformação deste em um Problema de Fluxo Máximo, em um Grafo valorado definido de forma conveniente.

Se encontrarmos um conjunto S tal que a DCFA correspondente é violada, obviamente, teremos também Desigualdade de Capacidade Arredondada Assimétrica (DCAA) violada neste conjunto. Poderemos então, incorporar a segunda desigualdade no PL atual, uma vez que esta desigualdade domina a primeira.

Faremos, primeiramente, um estudo deste procedimento na separação da Desigualdade de capacidade Fracionária Simétrica (DCFS) [31, 35, 7] (na seção seguinte) e, posteriormente, veremos quais as alterações necessárias para que tal procedimento possa ser utilizado na separação da Desigualdade de Capacidade Fracionária Assimétrica. As referidas alterações constituem nossa contribuição para esta seção.

Estaremos assumindo que este procedimento seja utilizado após a Heurística das Componentes Conexas e portanto, podemos supor que o grafo de entrada ($\vec{G}(\overline{x})$) é conexo.

6.3.1 Procedimento Exato Aplicado na Separação da DCFS

Denotaremos por $G_{\overline{x}}$ o grafo valorado induzido pelos arcos de G tais que as componentes correspondentes , da solução $\overline{x}$, do PL atual, são estritamente positivas e por todos os arcos incidentes ao nó depósito. Cada arco deste Grafo $G_{\overline{x}}$ terá a componente de $\overline{x}$ como peso. Recorde que no SCVRP o grafo é não orientado e portanto, $\overline{x}_{ij} = \overline{x}_{ji}$. A figura 6.10, dá um exemplo de um grafo $G_{\overline{x}}$.

A partir de $G_{\overline{x}}$, construimos um novo grafo $G'_{\overline{x}}$ substituindo os pesos de todo arco

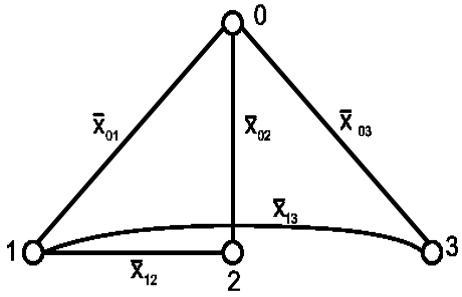

Figura 6.10: Grafo valorado $G_{\overline{x}}$ cujos pesos dos arcos correspondem às respectivas componentes de $\overline{x}$

$(0, j)$, por $b'_{0j} = \overline{x}_{0j} - 2\frac{d_j}{C}, \forall j \in V_0$ (os valores dos demais são conservados), e dando direção aos arcos deste grafo, determinando, assim, arcos (i, j) e (j, i) com pesos $\overline{x}_{ij}$ e $\overline{x}_{ji}$ ($\overline{x}_{ij} = \overline{x}_{ji}$), respectivamente, $\forall (i, j), \quad i, j \neq 0, \quad em \quad G_{\overline{x}}$. Para os arcos $(0, j)$ e $(j, 0)$ teremos pesos b'_{0j} e b'_{j0} ($b'_{j0} = b'_{0j}$), respectivamente. Podemos ver que se o corte de capacidade mínima F' possui peso negativo, então, o lado de F' que não contém o depósito define Desigualdade de Capacidade Fracionária violada. Este resultado será enunciado e provado na próxima proposição. É necessário lembrar que a capacidade do corte é dada pela soma dos pesos dos arcos que possuem origem no lado, determinado por F', que contém o nó zero, e destino, no outro lado. A alteração feita acima nos arcos está ilustrada na figura 6.11.

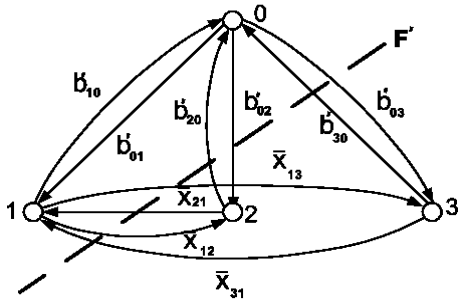

Figura 6.11: Corte F' no Grafo $G'_{\overline{x}}$ construido a partir de $G_{\overline{x}}$

Proposição 7 *Se em $G'_{\overline{x}}$ o corte de capacidade mínima possui valor negativo então, o lado deste corte que não contém nó zero corresponde a uma Desigualdade de capacidade Fracionária Simétrica violada.*

Dem:

A capacidade de F' é dada por $\sum_{(i,j)\in(V_0\setminus S',S')} \overline{x}_{ij} + \sum_{j\in S'} b'_{0j}$, onde S' é conjunto de nós, de $G'_{\overline{x}}$, que se encontram do lado de F' que não contém o depósito.

Uma vez que $b'_{0j} = \overline{x}_{0j} - 2\frac{d_j}{C}, \forall j \in V_0$, se a capacidade acima for negativa,

Teremos que $\sum_{(i,j)\in(V_0\setminus S',S')} \overline{x}_{ij} + \sum_{j\in S'} \overline{x}_{0j} - 2\sum_{j\in S'} \frac{d_j}{C} < 0$

E então, $\overline{x}(\delta^-(S')) - 2\frac{d(S')}{C} < 0$

E como os arcos da rede $G'_{\overline{x}}$ são simétricos, temos que $\overline{x}(\delta^-(S')) = \overline{x}(\delta(S'))$ e portanto:

$$\overline{x}(\delta(S')) - 2\frac{d(S')}{C} < 0$$

(Ou seja, DCFS violada) ■

Poderíamos então tentar identificar DCFS determinando o corte de capacidade mínima em $G'_{\overline{x}}$, porém, como deve existir arco com peso negativo neste grafo, não poderemos lançar mão de qualquer algoritmo conhecido, em tempo polinomial.

Com o objetivo de resolver o problema dos pesos negativos, vamos construir o grafo auxiliar $G''_{\overline{x}}$, a partir de $G_{\overline{x}}$, direcionando os arcos de $G_{\overline{x}}$, obtendo, desta forma, arcos (i,j) e (j,i), $\forall (i,j) \in G_{\overline{x}}$, e adicionando um nó $n+1$, destino de todo nó $j \in \{1,\ldots,n\}$. Os arcos do grafo $G''_{\overline{x}}$ possuirão pesos como abaixo:

- Os arcos (i,j) e (j,i) , $i,j \neq 0$, terão pesos $\overline{x}_{ij}$ e $\overline{x}_{ji}$, respectivamente.
- Os arcos $(0,j)$ e $(j,0)$,$\forall j \in V_0$, terão pesos $b_{0j} = b_{j0} = Max\{\overline{x}_{0j} - 2\frac{d_j}{C}, 0\}$.

- Os arcos $(n+1, j)$ e $(j, n+1), \forall j \in V_0$, terão pesos

  $b_{(n+1)j} = b_{j(n+1)} = Max\{2\frac{d_j}{C} - \overline{x}_{0j}, 0\}$

Podemos definir corte de capacidade, neste grafo modificado $G''_{\overline{x}}$, separando os nós 0 e $n+1$. E portanto, em vez de encontrarmos o corte de capacidade mínima F' em $G'_{\overline{x}}$, vamos determinar o corte de capacidade mínima F'' neste novo grafo, resolvendo o problema de fluxo máximo, de 0 até $n+1$ em $G''_{\overline{x}}$. Esta mudança, é motivada pelo fato de que o corte de capacidade mínima em $G''_{\overline{x}}$ corresponde a um corte de capacidade mínima em $G'_{\overline{x}}$ (e como já visto na Proposição 7, se este corte, em $G'_{\overline{x}}$, tem capacidade negativa então obtemos conjunto S tal que a Desigualdade de Capacidade Fracionária correspondente é violada). O fato mencionado acima será estabelecido e demostrado no próximo resultado. O novo grafo produzido é ilustrado na figura 6.12.

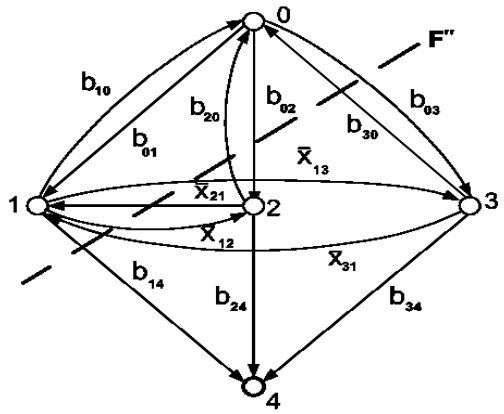

Figura 6.12: Corte F'' no Grafo $G''_{\overline{x}}$ construido a partir de $G_{\overline{x}}$

Repare que o grafo auxiliar $G''_{\overline{x}}$ possui pesos não negativos e, portanto, podemos determinar tal corte em tempo polinomial.

Proposição 8 *O corte de Capacidade Mínima F'' separando 0 de $n+1$ em $G''_{\overline{x}}$ gera corte de capacidade mínima F' em $G'_{\overline{x}}$ se removemos os arcos incidentes a $n+1$.*

Dem:

Daremos aqui uma demostração detalhada deste resultado.

Seja F um corte de capacidade qualquer separando 0 de $n+1$ em $G''_{\overline{x}}$.

Vamos primeiro mostrar que F gera corte em $G'_{\overline{x}}$ se removemos os arcos que chegam em $n+1$.

Seja S conjunto de nós clientes, do lado de F que não contém 0.

A capacidade do corte é $\sum\limits_{(i,j)\in(V_0\setminus S,S)} \overline{x}_{ij} + \sum\limits_{j\in S} b_{0j} + \sum\limits_{j\in V_0\setminus S} b_{j(n+1)} = Somat$

Se retiramos desta quantidade o valor $P = \sum\limits_{j\in V_0} b_{j(n+1)} = \sum\limits_{j\in V_0\setminus S} b_{j(n+1)} + \sum\limits_{j\in S} b_{j(n+1)}$

Obtemos:

$$Somat - P = \sum_{(i,j)\in(V_0\setminus S,S)} \overline{x}_{ij} + \sum_{j\in S} b_{0j} + \sum_{j\in V_0\setminus S} b_{j(n+1)} - \sum_{j\in V_0\setminus S} b_{j(n+1)} - \sum_{j\in S} b_{j(n+1)} =$$

$$= \sum_{(i,j)\in(V_0\setminus S,S)} \overline{x}_{ij} + \sum_{j\in S} (b_{0j} - b_{j(n+1)}) \tag{6.6}$$

Na diferença acima, quando fazemos $\sum\limits_{j\in V_0\setminus S} b_{j(n+1)} - \sum\limits_{j\in V_0\setminus S} b_{j(n+1)}$ estamos eliminando todos os arcos, do grafo $G''_{\overline{x}}$, incidentes a $n+1$, que estão no corte. Observe que se $(j, n+1)$ estiver no corte, $(0, j)$ não estará (e vice-versa).

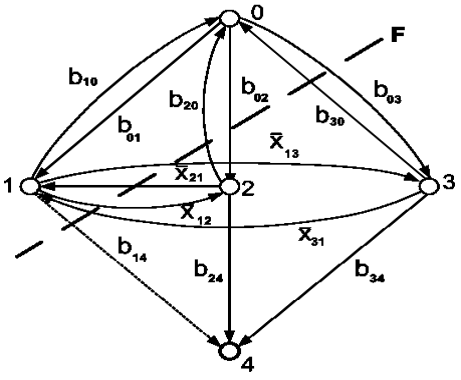

Figura 6.13: Remoção do arco $(j, n+1)$, do corte F de $G''_{\overline{x}}$

Enquanto que em $\sum\limits_{j\in S}(b_{0j} - b_{j(n+1)})$, estamos removendo os arcos incidentes a $n+1$, que

não estão no corte e, ao mesmo tempo, estamos atribuindo aos arcos $(0, j)$, que estão no corte F, os mesmos valores que eles possuem em $G'_{\overline{x}}$. Isso ocorre porque $b_{0j} - b_{j(n+1)} = Max\{0, \overline{x}_{0j} - 2\frac{d_j}{C}\} - Max\{0, 2\frac{d_j}{C} - \overline{x}_{0j}\}$ e então:

- Se peso de $(0, j)$ for zero $\Longrightarrow \overline{x}_{0j} - 2\frac{d(S)}{C} \leq 0 \Longrightarrow 2\frac{d(S)}{C} - \overline{x}_{0j} \geq 0 \Longrightarrow$ peso de $(j, n+1)$ é $2\frac{d(S)}{C} - \overline{x}_{0j} \geq 0$

  E daí, $b_{0j} - b_{j(n+1)} = \overline{x}_{0j} - 2\frac{d(S)}{C} \leq 0$

  Isto equivale a eliminar arco $(j, n+1)$ e atribuir a $(0, j) \in F$ o valor de seu peso em $G'_{\overline{x}}$.

- Se peso de $(0, j)$ for $\overline{x}_{0j} - 2\frac{d(S)}{C} \Longrightarrow \overline{x}_{0j} - 2\frac{d(S)}{C} \geq 0 \Longrightarrow 2\frac{d(S)}{C} - \overline{x}_{0j} \leq 0 \Longrightarrow$ Peso de $(j, n+1)$ é zero.

  E daí, $b_{0j} - b_{j(n+1)} = \overline{x}_{0j} - 2\frac{d(S)}{C} \geq 0$

  O que equivale a eliminar arco $(j, n+1)$ e atribuir a $(0, j) \in F$ o mesmo valor de seu peso em $G'_{\overline{x}}$.

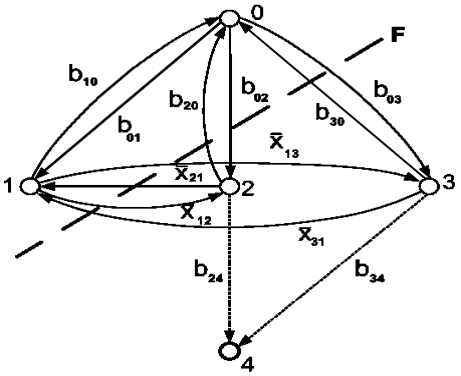

Figura 6.14: Remoção do arco $(j, n+1)$, não pertencente ao corte F de $G''_{\overline{x}}$

Então, retirando os arcos $(j, n+1)$ e atribuindo os pesos $\overline{x}_{0j} - 2\frac{d(S)}{C}, \forall (0, j)$, recaimos no grafo $G'_{\overline{x}}$ e o corte F de $G''_{\overline{x}}$ determina o corte L de $G'_{\overline{x}}$ com capacidade dada por (6.6).

Mostraremos agora que se tomarmos corte de capacidade mínima F'', em $G''_{\overline{x}}$, determinando $S'' \subseteq V_0$, então, F'' gera Corte de Capacidade Mínima em $G'_{\overline{x}}$.

O valor do corte F'' é:

$$\sum_{(i,j)\in(V_0\setminus S'',S'')} \overline{x}_{ij} + \sum_{j\in S''} b_{0j} + \sum_{j\in V_0\setminus S''} b_{j(n+1)} = Somat''$$

Se retiramos o valor P, desta quantidade, obtemos:

$$\sum_{(i,j)\in(V_0\setminus S'',S'')} \overline{x}_{ij} + \sum_{j\in S''} b_{0j} + \sum_{j\in V_0\setminus S''} b_{j(n+1)} - P = Somat'' - P$$

Como F'' é Corte de Capacidade Mínima $\Longrightarrow Somat'' \leq Somat \Longrightarrow$

$$\Longrightarrow Somat'' - P \leq Somat - P$$

E então, $Somat'' - P$ é o valor do corte de Capacidade Mínima F' de $G'_{\overline{x}}$.

Logo, o Corte de Capacidade Mínima F'' de $G''_{\overline{x}}$ gera Corte de Capacidade Mínima F' de $G'_{\overline{x}}$, removendo-se os arcos incidentes a $n+1$. ■

Em [31] este procedimento serve de base para uma heurística em que torna-se possível encontrar vários cortes diferentes, e não somente um.

6.3.2 Procedimento Exato Aplicado na Separação da DCFA

Nesta seção, vamos ver quais alterações serão necessárias no grafo $\vec{G}$ para que possamos determinar Desigualdade de capacidade Fracionária Assimétrica violada $\overline{x}(\delta^-(S')) - \frac{d(S')}{C} < 0$, em tempo polinomial, através da resolução de um problema de fluxo máximo.

Definimos o grafo $\vec{G}_{\overline{x}}$ como sendo o grafo valorado orientado induzido pelos arcos tais que as componentes associadas, da solução $\overline{x}$, do problema linear atual, são estrita-

mente positivas, e pelos arcos que possuem nó 0 como origem. Cada arco de $\vec{G}_{\overline{x}}$ tem a componente de $\overline{x}$ como peso. Repare que o ACVRP é orientado e então $\overline{x}_{ij} \neq \overline{x}_{ji}$.

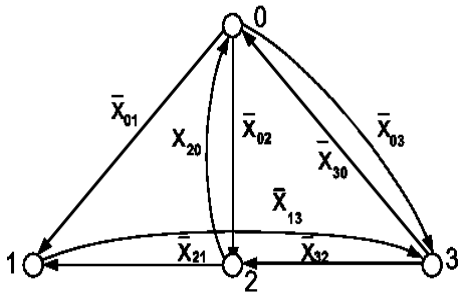

Figura 6.15: Grafo Suporte

Analogamente à seção anterior, produziremos o grafo orientado $\vec{G}'_{\overline{x}}$ atribuindo o peso $b'_{0j} = \overline{x}_{0j} - \frac{d_j}{C}$, a todos os arcos $(0, j)$, $j \in V_0$, e conservando todos os outros (como aqui o grafo já é orientado, obviamente, não será necessário dar direcão aos seus arcos). Sendo F' o Corte de Capacidade Mínima de $\vec{G}'_{\overline{x}}$, enunciaremos e demonstraremos um resultado, análogo àquele da seção passada (Proposição 7).

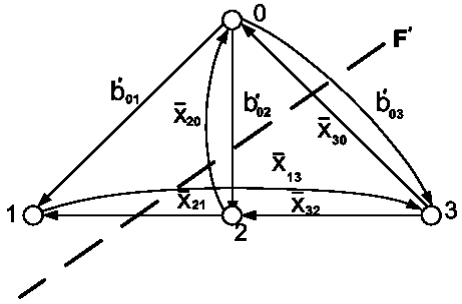

Figura 6.16: Grafo $\vec{G}'_{\overline{x}}$

Proposição 9 *Se em $\vec{G}'_{\overline{x}}$ o corte de capacidade mínima possui peso negativo, então, o lado deste corte que não contém nó zero define uma Desigualdade de capacidade Fracionária assimétrica violada por $\overline{x}$.*

Dem:

A capacidade de $F^{'}$ é a mesma do problema simétrico: $\sum_{(i,j)\in(V_0\setminus S^{'},S^{'})} \overline{x}_{ij} + \sum_{j\in S^{'}} b^{'}_{0j}$, onde $S^{'}$ é conjunto de nós, de $G^{'}_{\overline{x}}$, que se encontram do lado de $F^{'}$ que não contém o depósito.

Uma vez que $b^{'}_{0j} = \overline{x}_{0j} - \frac{d_j}{C}, \forall j \in V_0$, se a capacidade acima for negativa,

Teremos que $\sum_{(i,j)\in(V_0\setminus S^{'},S^{'})} \overline{x}_{ij} + \sum_{j\in S^{'}} \overline{x}_{0j} - \sum_{j\in S^{'}} \frac{d_j}{C} < 0$

E então, $\overline{x}(\delta^{-}(S^{'})) - \frac{d(S^{'})}{C} < 0$

(Ou seja, DCFA violada) ■

Como deve existir arco com peso negativo, não iremos determinar o Corte de Capacidade Mínima neste grafo, mas sim, no grafo $\vec{G}^{''}_{\overline{x}}$ que será construido a partir de $\vec{G}^{'}_{\overline{x}}$ por acrescentar o nó $n+1$, destino do arco$(j, n+1), \forall j \in \{1, \ldots, n\}$. Neste novo grafo, os pesos dos arcos $(0, j)$ e $(j, n+1)$ serão $b_{0j} = Max\{\overline{x}_{0j} - \frac{d_j}{C}, 0\}$ e $b_{j(n+1)} = Max\{\frac{d_j}{C} - \overline{x}_{0j}, 0\}$, respectivamente. O restante dos arcos possuirá o mesmo peso que em $\vec{G}^{'}_{\overline{x}}$. Este novo grafo não possuirá pesos negativos, o que permite determinar o Corte de Capacidade mínima em tempo polinomial. A figura 6.17 ilustra o novo grafo.

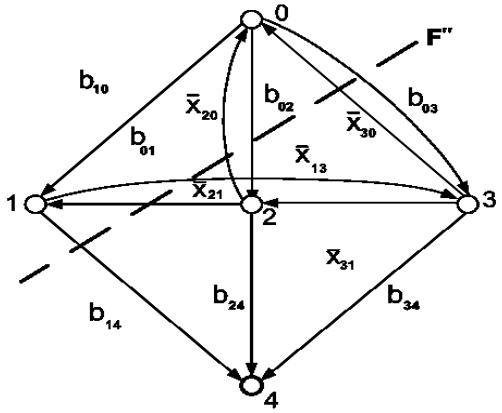

Figura 6.17: Grafo $\vec{G}^{''}_{\overline{x}}$

Determinaremos, então, o Corte de Capacidade Mínima em $\vec{G}^{''}_{\overline{x}}$, separando os nós 0 e

$n+1$, através da resolução do problema de fluxo Máximo, do nó 0 até $n+1$. A motivação para tal alteração será apresentada a seguir através de uma proposição.

Proposição 10 *O corte de Capacidade Mínima F'' separando 0 de $n+1$ em $\vec{G}''_{\overline{x}}$ gera corte de capacidade mínima F' em $\vec{G}'_{\overline{x}}$ se removemos os arcos incidentes a $n+1$.*

A demonstração desta proposição é idêntica a demonstração da proposição 8 e, da mesma forma, a partir dela, concluimos que se a Capacidade do Corte de Capacidade Mínima F' for negativa obtemos:

$$\overline{x}(\delta^-(S')) - \frac{d(S')}{C} < 0$$

que é Desigualdade de Capacidade Fracionária assimétrica violada.

A separação das DCAs pode ser feita utilizando os três procedimentos vistos neste capítulo, na ordem em que foram apresentadas. Conforme já mencionado, a heurística das componentes conexas foi o procedimento escolhido para ser implementado.

Capítulo 7

Resultados Computacionais

Neste capítulo, discutiremos o desempenho do nosso algoritmo *Branch-And-Cut* aplicado ao ACVRP, através da comparação deste com dois outros algoritmos existentes. Utlilizamos, para execução de nosso programa, computador pentium III 600 MHz com 384Mb de memória RAM. A implementação foi feita na linguagem de programação C\C++ e foi compilada pelo **GNU gcc**, versão 2.95 para Linux. Nossa implementação utilizou, como base, o **ABACUS**, que é uma biblioteca de classes para a implementação de algoritmos *Branch-And-Bound*, *Branch-And-Cut*, *Branch-And-Price* e *Branch-And-Cut-And-Price*. Este possui uma interface com o XPRESS, responsável pela resolução de todos as relaxações lineares de nosso *Branch-And-Cut*.

Como já mencionado anteriormente, o ACVRP não tem recebido muita atenção dos pesquisadores e, desta forma, poucos são os trabalhos que tratam este problema [27, 19]. Como uma conseqüência deste fato, há uma quantidade muito pequena de instâncias disponíveis na literatura e, portanto, para efetuar os testes computacionais, geramos outra classe de instâncias de acordo com o procedimento descrito em Laporte, Mercure e Nobert [27]. Esta nova classe de problemas será chamada P_2. Os resultados obtidos com as duas classes de problemas testados serão dados nas próximas seções e comparados

com aqueles de FTV [19] e LMN [27]. Na primeira seção, apresentaremos os resultados dos testes com as instâncias encontradas na literatura (chamaremos de P_1 esta classe de problemas) e na seguinte, apresentaremos aqueles produzidos para P_2.

Os computadores utilizados por LMN e FTV foram: VAX 11/780 (0.14 Mflops) e DECstation 5000/240 (5.3 Mflops), respectivamente. O critério de parada adotado em LMN foi a imposição de um limite para a memória total utilizada enquanto que, em FTV, foi imposto um limite de 1000 segundos. Interrompemos a execução do BC quando foi alcançado o tempo de 4000 segundos. Este valor foi escolhido com base no tempo máximo de resolução de algumas instâncias de sucesso, previamente testadas. Portanto, todos os resultados apresentados referem-se à instâncias de sucesso apenas. Veja que estes dados são meramente informativos uma vez que não estamos comparando o tempo de execução.

Estaremos considerando, nas próximas seções, o conjunto de nós clientes $V_0 = \{0, 1, \ldots, n-1\}$, e n o nó associado ao depósito. Definimos o percentual de carga como sendo o valor $100 \sum_{j \in V} d_j/(kC)$, que será denotado por $\%load$ e estará informando a quantidade de espaço ocupado em cada veículo, e a razão percentual ($\%LB/opt$) como $100(LB/opt)$, onde LB é o limite inferior obtido no nó raiz pelo nosso algoritmo e opt, o valor ótimo. O valor $\%LB/opt$ nos dará informação a respeito da qualidade dos limites inferiores obtidos no nó raiz da árvore de enumeração, ou seja, nos dirá o quão próximos eles se encontram do ótimo. O limite inferior no nó raiz, do algoritmo LMN [27], será representado por $L1$ e para o algoritmo FTV [19], por $L2$. Estaremos nos referindo ao nosso algoritmo como BC.

7.1 Classe de Problemas P_1

A primeira classe de problemas que utilizamos em nossos testes foi proposta na literatura e estão disponíveis eletronicamente no endereço http://www.or.deis.unibo.it/research_ pages/ORinstances/VRPLIB/VRPLIB.html. Tais problemas são referidos como problemas do "mundo real"e surgiram da distribuição de produtos farmacêuticos no centro da cidade de Bologna. A rede de transporte, neste caso, é constituída de 181 nós, 93 ligações bidirecionais e 226 ligações unidirecionais, sendo que estas últimas representam as ruas com sentido único obrigatório. Veja o mapa na figura 7.1, onde o quadrado representa o depósito e o circulo, os clientes. As ligações bidirecionais substituem duas ligações unidirecionais incidentes a um mesmo par de nós. Os custos de cada ligação foram computados através da distância Euclideana entre seus nós extremos, que representam clientes ou junção entre ruas e são definidos por suas coordenadas no plano, e arredondados para o inteiro mais próximo.

Existem 70 clientes, associados às farmácias, e cada um deles está representado por um nó da rede. A frota disponível possui três veículos, com capacidade 1000 cada. Todos eles partem de um único depósito para servir os clientes.

Cada uma das 8 instâncias do problema corresponde ao pedido de um dado subconjunto de clientes S em um determinado dia. Então, para o conjunto V_0 (de nós da rede) correspondente a S, os custos c_{ij}, $i, j \in V_0 \cup \{n\}, i \neq j$, n nó depósito, são determinados como o caminho mínimo do nó i até o nó j, ao longo de toda a rede. Ou seja, a matriz de custos satisfaz a desigualdade triangular. Aos $|V_0| = n$ nós clientes são associadas demandas d_i não-negativas.

As instâncias desta classe são identificadas por um nome que indica suas principais características. Por exemplo, $A039 - 03f$ representa a instância com 39 nós (38 nós clientes) e 3 veículos. Como já mencionado, em todas as instâncias, os veículos possuem

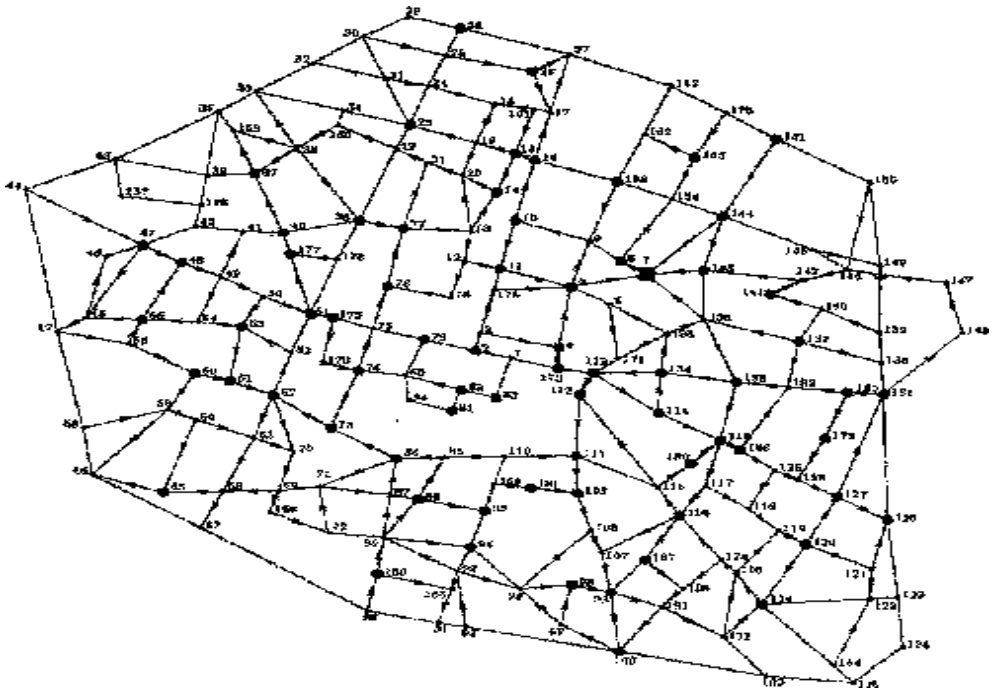

Figura 7.1: Mapa do centro da cidade de Bologna.

a mesma capacidade de 1000. Todas as instâncias utilizam os três veículos da frota, com excessão da instância $A034 - 02f$, que utiliza dois, como já é indicado em seu nome.

A tabela 7.1 apresenta os resultados computacionais, para as instâncias desta classe, dos algoritmos BC, FTV e LMN. Serão relatados, para cada uma dessas instâncias, o número de nós gerados (*sub*), e a razão percentual. Além disso, serão informados, nesta tabela, o número de nós clientes, o número de veículos utilizados, a carga percentual, o valor da solução ótima e o tempo de processamento (*tpo*) do BC.

O nosso algoritmo foi capaz de resolver todas essas instâncias e, para cada uma delas, conseguiu obter limites inferiores de melhor qualidade se comparados aos resultados de FTV e LMN (a razão média, nas 8 instâncias, foi 91.1, 94.5 e 97.2 para LMN, FTV e BC, respectivamente). Este resultado foi obtido do cálculo da percentagem da razão $\%LB/otimo$. Com relação ao FTV, conseguimos, também, reduzir a quantidade de

subproblemas gerados em todas as instâncias, sendo a média da redução do tamanho da árvore, nas 8 instâncias, igual a 7,90%. Devido a baixa qualidade dos limites inferiores fornecidos pela relaxação utilizada em [27], nenhuma dessas instâncias pôde ser resolvida pelo LMN.

Para a instância A065-03f, alteramos a estratégia de enumeração para busca em profundidade, uma vez que a memória que tínhamos disponível tornou-se insuficiente. Neste caso, com o intuito de evitar uma profundidade muito grande na árvore de enumeração, inicializamos a melhor solução viável, utilizando para isso, o valor 1975, bem próximo do ótimo.

As instâncias desta classe foram resolvidas em um tempo bem inferior ao tempo limite estabelecido. O tempo médio nas 8 instâncias foi de 93.3 segundos, menor que o da classe P2, como veremos na próxima seção. A quantidade média de cortes gerados, neste grupo, foi 1163.

Tabela 7.1: Instâncias teste da classe P1.

Nome	$\|V_0\|$	k	$\%load$	opt	FTV		BC		$\%(L1/opt)$	$tpo(s)$
					$\%(L2/opt)$	sub	$\%(LB/opt)$	sub		
$A034-02f$	33	2	97	1406	90.1	6331	95.6	347	85.8	18.5
$A036-03f$	35	3	81	1644	93.2	21197	95.9	411	90.9	23.6
$A039-03f$	38	3	93	1654	96.1	556	98.7	71	93.8	06.1
$A045-03f$	44	3	89	1740	95.7	2473	97.0	253	93.4	23.5
$A048-03f$	47	3	76	1891	97.2	2120	98.1	385	93.6	36.1
$A056-03f$	55	3	82	1739	94.3	89011	97.4	1441	88.5	303.7
$A065-03f$	64	3	85	1974	95.1	54198	97.2	6165	91.0	190.7
$A071-03f$	70	3	80	2054	94.6	19807	97.8	337	91.7	144.2
Razão Média:					94.5		97.2		91.1	

7.2 Classe de Problemas P_2

Nesta seção, os resultados computacionais apresentados foram obtidos através de testes nos problemas da classe P_2, proposta por Laporte, Mercure e Nobert [27]. Neles, as demandas d_j, $j \in V_0 = \{0, 1, \ldots, n-1\}$, e os custos c_{ij} são determinados de forma aleatória, a partir de uma distribuição uniforme no intervalo [0,100], e arredondado para o próximo inteiro. Para $j = n$ (depósito), fazemos $d_j = 0$. Então a capacidade do veículo é determinada por

$$C = (1-\alpha)\max_{j \in V_0}\{d_j\} + \alpha \sum_{j \in V} d_j,$$

onde α é um parâmetro real entre 0 e 1. A quantidade de veículos será computada como

$$k = \lceil \sum_{j \in V} d_j / C \rceil.$$

Repare que valores grandes de α resultam em valores grandes para C e, portanto, em valores pequenos para k. Quando $\alpha = 1$ recaimos no problema do caixeiro viajante uma vez que k=1.

Os resultados para as instâncias desta classe de problemas são dados, na tabela 7.2. Foram gerados 5 problemas para cada combinação dos parâmetros n e α, que, para efeito de comparação, assumiram os valores $n < 100$ (n múltiplo de 10) e $\alpha = 0.25, 0.5, 0.75, 1.0$. O valores de k produzidos foram $k = 4, 2, 2, 1$, respectivamente (com exceção daquela com n=10 que possui k=3 quando $\alpha = 0.25$). Para $n = 100, 150, 200, 250, 300$, consideramos também $\alpha = 0.30, 0.40$ levando a $k = 4, 3$, respectivamente. Todas as instâncias geradas, para cada par (n, α), possuirão o mesmo valor de k, como definido acima, uma vez que as demandas são geradas pela distribuição uniforme. Instâncias com o valor de α igual a 0.25 não foram testadas para $n \geq 100$. Perceba que o valor de $\%load$ será o mesmo para todas as instâncias com a mesma combinação de n e α. Isto ocorre devido a distribuição uniforme das demandas.

Assim como na classe P_1, os nomes das instâncias de P_2 as caracterizam. Por exemplo, $A100-04-030$ representa o problema com 100 nós e 4 veículos. Os últimos números indicam que $\alpha = 0.30$. Na tabela 7.2 faremos, também, comparações com os resultados dos algoritmos FTV [19] e LMN [27]. Os problemas desta classe não são idênticos aos de FTV [19] e de LMN [27], uma vez que estes foram gerados aleatoriamente. Portanto, a tabela 7.2 apresenta resultados médios computados nas 5 instâncias geradas para cada par (n, α). Serão dados, então, a média da razão percentual e do número de nós (sub) gerados, para todos os algoritmos. Nesta mesma tabela, além desses resultados são informados o número de instâncias resolvidas, entre parênteses, quando este for diferente de 5, e os seguintes dados de estrada: número de nós clientes da instância, o parâmetro α, o número de veículos utilizados e a carga percentual média, para cada par (n, α).

O algoritmo BC foi capaz de resolver todas as instâncias de até 100 nós se $k \leq 3$. Podemos perceber que, conforme a quantidade de veículos aumenta, as instâncias tornam-se mais difíceis. Repare que várias das instâncias com $\alpha = 0.25$ (possuem 4 veículos) deixaram de ser resolvidas e, para aquelas que conseguimos resolver, o BC gerou quantidade elevada de subproblemas. As instâncias com $\alpha = 0.30$ (que também possuem $k = 4$) são mais fáceis que aquelas com $\alpha = 0.25$. Isto ocorre porque a carga percentual associada a primeira é menor do que a da segunda e, portanto, torna-se mais fácil encontrar soluções viáveis, o que reduz a quantidade de subproblemas gerados. A carga percentual influencia diretamente na resolução dos problemas. Veja, por exemplo, que para as instâncias $A100-04-030$ ($\%load = 80$) foi gerada quantidade menor de subproblemas do que para $A100-02-050(\%load = 98)$, que utiliza menor quantidade de veículos. Para instâncias com mais de 100 nós, só conseguimos resolver todas as 5 instâncias quando $\alpha = 0.75$ ou α =1. Para os demais valores, esta quantidade foi reduzida a medida em que aumentamos o número de nós no grafo. Quando $\alpha = 0.50$,

nenhuma delas foi resolvida. O tempo médio de CPU, desta classe de problemas, foi de 388.5 segundos. Algumas delas chegaram bem próximas do limite estabelecido. Veja que tal limite foi tomado com base no máximo de tempo dentre as instâncias previamente resolvidas, desta classe. As instâncias que não puderam ser resolvidas ultrapassaram o limite de tempo imposto. A quantidade média de cortes gerados nesta classe de instâncias foi de 195.24.

Ao comparar nossos resultados com os de FTV e LMN, verificamos que para a maioria das instâncias resolvidas, a qualidade de nossos limites inferiores é superior (veja que a razão média obtida por BC, nas instâncias resolvidas, é maior que a de LMN e de FTV, calculada em todas desta classe). No entanto, para algumas delas isto não foi suficiente para que a quantidade de subproblemas gerados fosse menor. Por exemplo, nas instâncias $A100-02-050$, a razão média para o BC foi de 99.3 e a quantidade de subproblemas gerados foi 571, enquanto que para o FTV estes valores foram, respectivamente, 98.5 e 527. Acreditamos que isto decorra do fato de FTV ter utilizado limites superiores obtidos por heurísticas. Para os casos em que a qualidade do limite inferior de FTV foi melhor do que a de BC, já era de se esperar que a quantidade de subproblemas gerados tenha sido menor. Porém, note que a ocorrência de tais limites inferiores deve-se ao fato de que, em FTV, assim como em BC, o procedimento de *Bounding*, descrito em 4.2.2, representa uma melhoria daquele que consiste da resolução da relaxação do problema de transporte, o que o torna competitivo com o BC. Conseguimos gerar menores quantidades de subproblemas em pouco mais de 50% das instâncias resolvidas.

Tabela 7.2: Instâncias teste da classe P2.

$\|V_0\|$	α	k	$\%load$	LMN		FTV		BC	
				$\%(L1/opt)$	sub	$\%(L2/opt)$	sub	$\%(LB/opt)$	sub
10	0.25	3	83	86.9	1443	93.8	15	94.0	28
	0.50	2	83	97.9	98	100	1	98.0	4
	0.75	2	62	97.9	109	100	1	98.0	3
	1.00	1	100	97.4	12	99.3	3	100	1
20	0.25	4	81	85.9	9648	89.4	187	94.4	235
	0.50	2	91	88.5	635	92.9	36	96.5	61
	0.75	2	64	90.4	367	94.9	10	99.9	2
	1.00	1	100	92.4	20	95.7	16	99.4	3
30	0.25	4	83	93.5	–	95.3	233	95.3	306
	0.50	2	94	97.4	662	98.8	28	98.3	71
	0.75	2	65	98.4	137	99.8	8	98.7	10
	1.00	1	100	98.8	31	99.1	7	99.0	9
40	0.25	4	87	93.1	–	94.5	498	96.3	655
	0.50	2	95	95.0	5153	96.8	335	97.8	107
	0.75	2	66	96.5	3042	98.3	18	99.2	15
	1.00	1	100	97.4	25	99.7	8	99.6	13
50	0.25	4	90	93.4	–	95.3	522	97.1	1991
	0.50	2	96	97.2	5266	98.1	150	97.4	436
	0.75	2	66	98.5	1530	99.4	18	99.8	15
	1.00	1	100	98.0	14	99.0	41	99.3	23

Tabela 7.3: Instâncias teste da classe P_2 (continuação da tabela 7.2).

$\|V_0\|$	α	k	$\%load$	LMN		FTV		BC	
				$\%(L1/opt)$	sub	$\%(L2/opt)$	sub	$\%(LB/opt)$	sub
60	0.25	4	01	94.7	–	95.4	1272	97.3	$599_{(4)}$
	0.50	2	97	97.7	3630	98.2	206	98.6	239
	0.75	2	66	98.9	3342	99.4	32	99.7	7
	1.00	1	100	98.7	42	99.0	44	99.3	20
70	0.25	4	92	96.2	–	96.7	1398	98.1	$1161_{(4)}$
	0.50	2	97	97.1	138	97.5	632	98.8	324
	0.75	2	66	98.5	1364	98.8	14	99.5	33
	1.00	1	100	98.3	18	98.5	39	99.6	28
80	0.25	4	93	95.4	–	96.9	7032	—	—
	0.50	2	98	97.3	7601	98.3	409	98.6	574
	0.75	2	66	98.6	3570	99.5	22	99.5	85
	1.00	1	100	98.2	76	98.9	56	100	16
90	0.25	4	94	95.1	–	96.6	3262	98.2	$1941_{(3)}$
	0.50	2	98	97.2	7224	98.0	543	98.8	438
	0.75	2	66	98.2	3239	99.0	25	99.6	21
	1.00	1	100	98.1	22	99.2	35	99.9	22
100	0.30	4	80	97.0	–	98.5	583	99.4	$366_{(4)}$
	0.40	3	81	98.6	–	99.5	63	99.4	155
	0.50	2	98	97.5	–	98.5	527	99.3	571
	0.75	2	66	98.2	–	99.1	27	99.6	58
	1.00	1	100	98.3	–	99.1	57	99.7	23

Tabela 7.4: Instâncias teste da classe P_2 (continuação da tabela 7.2).

$\|V_0\|$	α	k	$\%load$	LMN		FTV		BC	
				$\%(L1/opt)$	sub	$\%(L2/opt)$	sub	$\%(LB/opt)$	sub
150	0.30	4	81	98.6	–	99.4	604	99.6	$158_{(3)}$
	0.40	3	82	99.0	–	99.2	265	99.8	$144_{(4)}$
	0.50	2	99	98.3	–	98.8	1823	99.5	$416_{(3)}$
	0.75	2	66	99.4	–	99.8	56	99.9	51
	1.00	1	100	99.4	–	99.8	80	99.9	31
200	0.30	4	81	98.8	–	99.0	2132	100	$328_{(2)}$
	0.40	3	82	99.2	–	99.7	220	99.8	239
	0.50	2	99	98.9	–	99.2	2979	—	—
	0.75	2	66	99.2	–	99.5	92	100	25
	1.00	1	100	98.9	–	99.7	27	99.9	61
250	0.30	4	82	98.9	–	100	1896	100	$146_{(2)}$
	0.40	3	82	98.9	–	100	835	100	159
	0.50	2	99	98.1	–	99.1	1987	—	—
	0.75	2	66	99.2	–	100	62	100	23
	1.00	1	100	99.2	–	99.6	79	99.9	27
300	0.30	4	82	100	–	100	1770	100	$13_{(1)}$
	0.40	3	83	100	–	100	413	100	$51_{(1)}$
	0.50	2	99	100	–	100	3447	—	—
	0.75	2	67	100	–	100	27	100	20
	1.00	1	100	100	–	100	15	100	14
Razão Média:				97.1		98.3		98.9	

Capítulo 8

Conclusões e Sugestões para Trabalhos Futuros

A contribuição deste trabalho está na introdução de três algoritmos de separação para a Desigualdade de Capacidade do ACVRP, obtidos da extensão daqueles já existentes para a versão simétrica do mesmo, e da resolução deste problema através do algoritmo *Branch-And-Cut* (BC) utilizando a primeira daquelas heurísticas para separar os cortes de Capacidade. Até então, os métodos utilizados para sua resolução eram algoritmos *Branch-And-Bound* [19] e [27], estado da arte com relação aos métodos exatos, e algumas heurísticas. Este é um problema de difícil solução e tem sido muito pouco trabalhado, apesar de sua importância em casos em que todo trajeto possui sentido obrigatório.

Os resultados computacionais obtidos com nosso algoritmo mostraram-se bem satisfatórios para problemas com até três veículos. Para quatro veículos, começamos a não resolver as instâncias a medida em que a quantidade de nós clientes aumenta. Nossos limites inferiores, obtidos com o métodos de planos de cortes, possuem melhor qualidade que os de FTV [19] e LMN [27] para a maioria das instâncias resolvidas (em 100% delas, com relação ao LMN). Em função disso, salvo poucas excessões, conseguimos também

reduzir o tamanho da árvore de enumeração. Para os casos em que o gap menor não foi suficiente para gerarmos menos nós, com relação a FTV, acreditamos que isto se deva ao fato de FTV ter utilizado limites superiores obtidos por heurísticas. Para o BC, ao contrário do FTV e LMN, as instâncias da classe P2 se mostraram mais difíceis de se resolver, do que aquelas de P1.

De acordo com as conclusões mencionadas acima, percebemos que é possível melhorar nossa implementação para obter melhores resultados computacionais. Tais melhorias poderiam ser alcançadas através da implementação de outros algoritmos de separação, como, por exemplo, o de contração e o exato, ambos apresentados nesta dissertação. Isto, provavelmente, levaria a melhores limites inferiores e portanto reduziria o tamanho da árvore de enumeração. Outra idéia seria a implementação de alguma heurística que gerasse uma solução viável inicial para o nosso *Branch-And-Cut*, o que poderia, também, reduzir a quantidade de nós da árvore. Repare que, nos problemas da classe $P2$, o nosso algoritmo *Branch-And-Cut*, na maioria das instâncias, encontrou um limite inferior no nó raiz melhor que aquele do *Branch-And-Bound* de FTV [19] e, no entanto, em algumas delas, geramos quantidade maior de subproblemas, certamente, porque FTV lançou mão deste artifício.

Acreditamos que uma forma de se obter bons resultados, para instâncias cujo número de veículos é grande, seria através da inclusão de uma fase de geração de colunas. O algoritmo obtido é denominado *Branch-And-Cut-And-Price* que, para o SCVRP [21], levou a excelentes resultados, o que traz uma grande motivação para experimentos com o ACVRP.

Referências Bibliográficas

[1] N. Achuthan, L. Cacceta and S. Hill; *Capacited Vehicle Routing Problem: Some New Cutting Planes*. Asia-Paci.c Journal of Operational Research, 15(1998) 109-123.

[2] N. Achuthan, L. Cacceta, S. Hill; *An Improved Branchand- Cut Algorithm for the Capacitated Vehicle Routing Problem*. Transportation Science, 37(2003) 153-169.

[3] Y. Agarwal, K. Mathur and H. Salkin; *A Set-Partitioning Based Exact Algorithm for the Vehicle Routing Problem*. Networks, 19(1989) 731-739.

[4] J. Araque, G. Kudva, T. Morin, J. Pekny; *A branch-and-cut algorithm for the vehicle routing problem*. Annals of Operations Research, 50(1994) 37-59.

[5] J. Araque, L. Hall, T. Magnanti; *Capacitated trees, capacitated routing and polyhedra*. Technical Report SOR-90-12, Princeton University, 1990.

[6] P. Augerat; *Approche polyèdrale du problème de tournées de véhicles*. PhD thesis, Institut National Polytechnique de Grenoble, 1995.

[7] P.Augerat, J.M.Belenguer, E.Benavent, A. Corberán, D.Naddef; *Separating Capacity Constraints in the CVRP using tabu search*, European Journal of Operational Research 106(1998) 546-557.

[8] M. Balinski and R. Quandt; *On an Integer Program For a Delivery Problem*. Operations Research, 12(1964) 300-304.

[9] U. Blasum and W. Hochstättler; *Application of the Branch and Cut Method to the Vehicle Routing Problem*. Technical Report ZPR2000-386, Zentrum fur Angewandte Informatik Köln, (2000).

[10] G. Carpaneto and P.Toth; *Some new branching and bounding criteria for the asymmetric travelling salesman problem*. Management Sci. 26(1980) 736-743.

[11] N. Christofides, A. Mingozzi and P.Toth; *Exact Algorithms for the Vehicle Routing Problems, based on Spanning Tree and Shortest Path Relaxations*. Mathematical Programming, 20(1981) 255-282.

[12] W.Cook and J.L. Rich; *A parallel cutting plane algorithm for the vehicle routing problem with time windows*. Technical report, Computational and Applied Mathematics, Rice University, Houston, TX, (1999).

[13] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver; *Combinatorial Optimization*. John Wiley and Sons, New York (1998).

[14] G. Cornuéjols and F. Hsrche; *Polyhedral study of the capacitated vehicle routing problem*. Mathematical Programming, 60(1993) 21- 52.

[15] G. B.Dantzig, J. H.Ramser; *The Truck Dispatching Problem*, Management Science, 6(1):80-91, 1959.

[16] J.Desrosiers, Y. Dumas, and F.Soumis; *A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows*. American Journal of Mathematical and Management Sciences, 6 (1986) 301-325.

[17] J.Desrosiers, Y. Dumas, M.M. Solomon and F.Soumis; *Time constrained routing and scheduling*. In M.O.Ball, T.L. Magnanti, C.L Monma, and G.L. Nemhauser, editors, Network Routing , Handbooks in Operations Research and Management Science 8 (1995), North-Holland, Amsterdam, 35-139.

[18] M.Fischetti and P. Toth; *An Additive Bounding Procedure for Combinatorial Optimization Problems*. Opns.Res. 37 (1989) 319-328.

[19] M.Fischetti, P. Toth, and D. Vigo; *A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs*. Operations research, 42:846-859, 1994.

[20] M. Fisher; *Optimal solution of vehicle routing problem using minimum K-trees*. Operations Research, 42(1994) 626-642.

[21] R. Fukasawa, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa and R. F. Werneck; *Robust branch-and-cut-and-price for the capacitated vehicle routing problem*. In: PROCEEDINGS OF THE X IPCO, volume 3064 de Lecture Notes in Computer Science, p. 1-15, New York, June 2004.

[22] R.E. Gomory; *Outline of an algorithm for integer solution to linear programs*. Bulletim Amer. Math. Soc. 64-5(1958) 275-278.

[23] M. Grötschel, M.W. Padberg (1985). *Polyhedral Theory*. In The Traveling Salesman Problem, Lawler, Lenstra, Rinooy Kan and Shmoys, eds., John Wiley, 251-306.

[24] F. Harche, G. Rinaldi (1991). *Vehicle Routing*. Private Communication.

[25] B. Kallehauge, J. Larsen and O.B.G. Madsen; *Lagrangean Duality and Nondifferentiable optimization applied on routing with time windows - experimental results*. Internal report IMM-REP-2000-, Department of mathematical modelling, Technical University of Denmark, Lyngby, Denmark., 2000.

[26] A.H. Land and A. G. Doig; *An Automatic Method for Solving Discrete Programming Problems*. Econometrica, 28(1960) 497-520.

[27] G. Laporte, H. Mercure, and Y. Nobert; *An exact algorithm for the asymmetrical capacitated vehicle routing problem*. Networks, (1986) 16:33-46.

[28] J. Larsen; *Parallellization of the vehicle routing problem with time windows*. Ph.D. Thesis IMM-PHD-1999-62, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1999.

[29] A. Letchford, R. Eglese, J. Lysgaard; *Multistars, partial multistars and the capacitated vehicle routing problem*. Mathematical Programming, 94(2002) 21-40.

[30] J.K. Lenstra and A.H. Rinnooy Kan; *Some simple applications of the travelling Salesman Problem*. Operational res. Q. 26 (1975) 717-734.

[31] J.Lysgaard, A Letchford, and R. Eglese; *A new branch-and-Cut algorithm for Capacitated vehicle routing problems* ,Matematical Programming, 100(2004).

[32] C. Martinhon, A. Lucena and N. Maculan; *Stronger k-tree relaxations for the Vehicle Routing Problem*. European Journal of Operational Research, 158(2004) 56-71.

[33] D. Miller. *A matching based exact algorithm for capacitated vehicle routing problems*; ORSA Journal on Computing, 7(1995) 1-9.

[34] A. Mingozzi, S. Giorgi and R.Baldacci; *An exact method for the vehicle roouting problem with Backhauls* Transportation Science, 33 (1999) 315-329.

[35] - D. Naddef, G. Rinaldi; *Branch and Cut Algorithm for the Capacitated VRP*. In P.Toth and D.Vigo. The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications, vol. 9, Philadelphia, PA, 53-84, (2002).

[36] M. Padberg, and G. Rinaldi; *A Branch-and-Cut Algorithm for the Resolution of Large-Scale Traveling Salesman Problems*. SIAM Review, 33(1991) 60-100.

[37] M.Poggi de Aragão and E. Uchoa; *Integer program reformulation for RBCP*, August 2003. Working paper presented at ISMP'03 Copenhagen.

[38] T.K. Ralphs, L.Kopman, W.R. Pulleyblank, and L.E.Trotter, Jr.; *On the Capacitated Vehicle Routing Problem*, Mathematical Programming 94 (2003) 343-359.

[39] K.S Ruland and E.Y.Rodin; *The Pickup and Delivery problem: Faces and branch-and-cut algorithm*. Computers and Mathematics with Applications, 33 (1997) 1-13.

[40] M.M. Solomon and J. Desrosiers; *Time Windows Constrained routing and scheduling problems*. Transportation Science, 22 (1988) 1-13.

[41] P.Toth and D. Vigo; *An exact Algorithm for the vehicle routing problem with Backhauls*, Trasnportation Science, 31 (1997) 372-385.

[42] P.Toth and D. Vigo; *Branch-And-Bound Algorithms for the Capacitated VRP*. In P.Toth and D.Vigo. The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications, vol. 9, Philadelphia, PA, 29-52, (2002).

[43] D. Vigo; *A Heuristic Algorithm for the Asymmetric Capacitated Vehicle Routing Problem*, Eur.J.Opnl. Res. 89 108-126, (1996)

[44] L. Wolsey; *Integer Programming*. Wiley Interscience, (1998).

[45] L. Wolsey and G. Nemhauser. Integer and Combinatorial Optimization. Wiley-Interscience Series in Discrete Mathematics and Optimization, (1999).