HEURÍSTICAS E ALGORITMO EXATO PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEAS

DILSON LUCAS PEREIRA

HEURÍSTICAS E ALGORITMO EXATO PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEAS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: GERALDO ROBSON MATEUS

Belo Horizonte

Fevereiro de 2010

Pereira, Dilson Lucas
P436h Heurísticas e Algoritmo Exato para o Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas / Dilson Lucas Pereira. — Belo Horizonte, 2010
xvi, 67 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Geraldo Robson Mateus

1. Otimização Matemática - Teses. 2. Programação (Matemática) - Teses. 3. Algoritmos de Computador - Teses. 4. Transporte Rodoviário - Processamento de Dados - Teses. I. Orientador. II. Título.

CDU 519.6*61(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Heurísticas e algoritmo exato para o problema de roteamento de veículos com coleta e entrega simultâneas

DILSON LUCAS PEREIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. GERALDO ROBSON MATEUS - Orientador
Departamento de Ciência da Computação - UFMG

PROF. SÉRGIO RICARDO DE SOUZA
Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG

PROF. ALEXANDRE SALLES DA CUNHA
Departamento de Ciência da Computação - UFMG

PROF. ALYSSON MACHADO COSTA
Instituto de Ciências Matemáticas e de Computação - ICMC - USP

Belo Horizonte, 25 de fevereiro de 2010.

Agradecimentos

Agradeço aos meus pais, Dilson e Catarina, por todo o suporte que sempre me proporcionaram. Agradeço a minha namorada, Juliana, por sua importante presença em todos os momentos. Aos professores do DCC/UFMG, que tanto contribuíram para o meu desenvolvimento intelectual, e em especial ao professor Robson, por toda dedicação em minha orientação.

Resumo

Este trabalho trata do Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas, para o qual devem ser desenvolvidas rotas para atender as demandas de coleta e entrega de um conjunto de consumidores. Cada consumidor deve ser atendido por uma única rota, a carga recebida é trazida de um depósito central, para onde também é levada toda a carga coletada. A capacidade dos veículos utilizados não deve ser violada em nenhum ponto das rotas. O problema é abordado de maneira heurística e exata. Inicialmente, são desenvolvidas heurísticas híbridas baseadas em idéias de diversas metaheurísticas, em especial ILS, GRASP e VND. Essas heurísticas são testadas em diversas instâncias e comparadas aos melhores resultados da literatura, obtendo resultados competitivos com os melhores algoritmos existentes. É desenvolvido um método exato *Branch-and-price* e, nesse contexto, são propostas novas estratégias para a resolução do subproblema, um Problema de Caminho Elementar Mínimo com Restrição de Recursos. Em experimentos computacionais, essas estratégias demonstram redução significativa no tempo computacional da resolução da Geração de Colunas. É apresentada também uma estratégia em que limites inferiores obtidos a partir da geração de colunas são utilizados no processo de decisão do *branching*. O algoritmo *Branch-and-price* é testado e comparado a um algoritmo *Branch-and-cut-and-price* da literatura.

Palavras-chave: Roteamento de Veículos, Coleta e Entrega, Heurísticas, Otimização Combinatória, Programação Inteira.

Abstract

This work adresses the Vehicle Routing Problem with Simultaneous Pickup and Delivery, where routes must be devised to fulfil the pickup and delivery requests of a set of customers. Each customer must be served by only one route, the load it receives is brought from a central depot, to where the picked-up load is also taken. The capacity of the used vehicles must not be violated at any point of the routes. Heuristics and an exact algorithm are proposed to solve the problem. Initially, we devise some heuristics based on ideas of many metaheuristics, specially ILS, GRASP, and VND. These heuristics are tested in many instances and are compared to the best results of the literature, obtaining results comparable to the best existing algorithms. Afterwards, a Branch-and-price method is developed, and in this context, we propose new strategies to the resolution of the subproblem, an Elementary Resource Constrained Shortest Path Problem. Based on computational experiments, these strategies show considerable reduction on the computational time of the Column Generation resolution. We present a strategy in which lower bounds obtained from the column generation are used in the branching process. The Branch-and-price algorithm is tested and compared to a Branch-and-cut-and-price algorithm from the literature.

Keywords: Vehicle Routing, Pickup and Delivery, Heuristics, Combinatorial Optimization, Integer Programming.

Lista de Tabelas

Sumário

Capítulo 1

Introdução

1.1 Contextualização

Este trabalho trata do Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas (*Vehicle Routing Problem with Simultaneous Pickup and Delivery*, VRPSPD), cujo objetivo é desenvolver um conjunto de rotas para atender um conjunto de consumidores percorrendo a distância mínima possível. Cada rota determina um trajeto a ser percorrido por um veículo, que deve partir e retornar a um depósito central. Cada consumidor recebe e/ou entrega uma quantidade de carga ao veículo e só pode ser atendido por um único veículo e uma única vez.

1.2 Motivação

Nas últimas décadas um grande esforço de pesquisa tem sido empreendido à modelagem e à métodos de otimização no campo do roteamento de veículos.

A principal motivação para o estudo destes problemas são os altos gastos com transporte de carga, que representam uma grande parcela do preço da matéria prima e do produto final [1]. É estimado que os custos de distribuição somam quase metade dos custos totais de logística e em alguns casos, como na indústria de alimentos e bebidas, podem chegar a 70% do valor [7]. No Brasil, segundo dados da Confederação Nacional de Transportes (CNT), a maioria do transporte de cargas e passageiros é rodoviário. Diversos itens afetam o custo do transporte rodoviário, como, por exemplo: combustível, manutenção, pneus, pedágios, impostos, gastos com pessoal, etc.

Durante as últimas décadas, a crescente preocupação ambiental tem levado a um aumento dos esforços para proteger o meio ambiente, tanto por parte da indústria como

por parte das entidades legislativas. Segundo dados da CNT, no Brasil, levando em consideração os tipos de transporte, o transporte rodoviário é aquele que mais emite CO_2 na atmosfera. Além dos gases estufa, a maior preocupação com a reciclagem e reuso de bens e embalagens gera também um fluxo contrário na cadeia de suprimentos, bens devem ser entregues e recolhidos. Dessa maneira, busca-se uma integração ótima entre o fluxo de carga dos produtores para os consumidores e fluxo de material usado dos consumidores para depósitos ou locais de reciclagem [17].

O estudo dos problemas de roteamento proporciona um uso eficiente da malha rodoviária e da frota de veículos, ajudando a reduzir os custos e amenizando outros problemas como a poluição. Aplicações no mundo real, na América e na Europa, mostram que o uso de procedimentos computadorizados para o planejamento do processo de distribuição produz economia significativa nos custos globais de transporte [44]. No Brasil, dados da CNT mostram que apesar das vantagens da otimização de rotas, poucas empresas utilizam sistemas de informação com esse fim.

Uma outra motivação é que todos os tipos de problemas de roteamento de veículos com coleta e entrega são NP-Difíceis, pois são generalizações do problema do caixeiro viajante (*Traveling Salesman Problem*, TSP) [37].

1.3 Principais Contribuições deste Trabalho

As principais contribuições deste trabalho são:

- O desenvolvimento de heurísticas híbridas baseadas em metaheurísticas para resolver o problema. As heurísticas são comparadas com os melhores trabalhos da literatura, obtendo resultados semelhantes.

- A demonstração da equivalência entre os dois tipos de instâncias propostos por Salhi e Nagy [42].

- O desenvolvimento de um algoritmo *Branch-and-Price* para o problema. Nesse contexto:

  - O desenvolvimento de novas estratégias (que podem ser adotadas em algoritmos para outros VRPs) para acelerar a resolução do Subproblema e para gerar rotas de custo reduzido negativo para o Problema Mestre mais rapidamente. Em experimentos computacionais, essas estratégias proporcionaram uma grande diminuição do tempo computacional para a resolução da Relaxação Linear do Problema Mestre.

– O desenvolvimento de uma nova estratégia para a escolha do arco sobre o qual é realizado o *branching*. Esta estratégia também pode ser adotada em algoritmos para outros problemas de roteamento de veículos.

1.4 Organização do Texto

O texto da dissertação é organizado da seguinte maneira: No capítulo 2 é dada uma visão geral sobre Problemas de Roteamento de Veículos e Problemas de Coleta e Entrega. O VRPSPD é discutido em detalhes no capítulo 3, no qual são listados alguns trabalhos anteriores que abordam o problema e são reunidas algumas propriedades. As heurísticas desenvolvidas são apresentadas no capítulo 4, são dadas breves introduções às metaheurísticas utilizadas, conforme o necessário, e, ao fim do capítulo são apresentados resultados computacionais. No capítulo 5, é apresentado o algoritmo *Branch-and-price* desenvolvido, nesse capítulo é dada uma breve introdução ao método de Geração de Colunas e é apresentado um modelo matemático e uma reformulação para o problema (desenvolvidos em [15]). Na seção 5.3, a resolução do Subproblema é discutida em detalhes, são introduzidas estratégias para a aceleração da obtenção de colunas com custo reduzido negativo e experimentos computacionais para estas estratégias são apresentados. Após isso, são apresentados outros detalhes da implementação e resultados computacionais para o algoritmo *Branch-and-price*. Finalmente, o capítulo 6 conclui o trabalho.

Capítulo 2

Problemas de Roteamento de Veículos

2.1 O Problema de Roteamento de Veículos

Problemas de Roteamento de Veículos (*Vehicle Routing Problems*, VRP), envolvem o atendimento, em um dado período de tempo, de um conjunto de consumidores por um conjunto de veículos, que estão localizados em um ou mais depósitos, são operados por um conjunto de tripulantes e se movem através de uma malha rodoviária. A solução do VRP exige a determinação de um conjunto de rotas, cada uma utilizada por um único veículo, que começa e termina em seu depósito, de modo que todos os requisitos dos consumidores sejam atendidos, todas as restrições operacionais sejam satisfeitas e o custo de transporte global seja minimizado [44]. Geralmente, na literatura, dois objetivos são explorados: redução da distância total e/ou redução do número de veículos utilizados.

A malha rodoviária usada para o transporte dos bens é geralmente representada por um grafo, no qual os arcos representam estradas e os vértices correspondem a junções de estradas, depósitos ou consumidores. O grafo pode ser direcionado ou não direcionado, dependendo se os arcos podem ser atravessados em uma (mão única, típico de vias urbanas) ou nas duas direções. Cada arco possui um custo e um tempo de tráfego, que possivelmente é dependente do veículo ou período em que o arco é atravessado.

Considerando problemas envolvendo apenas a entrega de bens, tem-se os seguintes VRP's básicos [44]:

- VRP Capacitado (*Capacitated VRP*, CVRP). Todos os consumidores correspon-

dem a entregas e as demandas são determinísticas, conhecidas a priori e não podem ser divididas. Os veículos são idênticos e existe apenas um depósito, são impostas apenas as restrições de capacidade dos veículos.

- VRP com Restrição de Distância (*Distance-Constrained VRP*, DVRP). As restrições de capacidade são substituídas por restrições de comprimento máximo, ou seja, o comprimento de cada rota não pode exceder um valor T.

- VRP com Janelas de Tempo (*VRP with Time Windows*, VRPTW). Têm-se as restrições de capacidade e, além disso, cada consumidor possui associado um intervalo de tempo (Janela de Tempo). O veículo deve iniciar o serviço no consumidor dentro da janela de tempo e deve permanecer por um dado tempo de atendimento naquele consumidor. Geralmente, se o veículo chega antes do início da janela de tempo ele deve esperar até a abertura da janela. Se não for possível que o veículo chegue ou conclua seu serviço dentro da janela de tempo, o consumidor não será atendido ou o atraso será penalizado.

Para cada um destes problemas existem diversas variantes e extensões abordadas na literatura.

A seguir, é apresentada uma classe de problemas de roteamento em que bens são entregues e também coletados.

2.2 O Problema de Coleta e Entrega

Problemas de Coleta e Entrega (PCE) constituem uma classe importante de problemas de roteamento de veículos na qual, além de distribuídos, bens também devem ser coletados [4].

É importante ressaltar que a rápida expansão da pesquisa na área levou a uma terminologia confusa para descrever os vários tipos de problemas que surgem no contexto do Problema de Coleta e Entrega. Em alguns casos problemas iguais são denotados por nomes diferentes e problemas diferentes recebem a mesma denotação. Em [37] e [38] busca-se prover um esquema de classificação mais claro.

Podem-se distinguir duas classes de problemas diferentes no campo dos problemas de coleta e entrega:

- Problemas de Roteamento de Veículos com *Backhauls* (*Vehicle Routing Problems with Backhauls*, VRPB): Tratam de situações em que os bens a serem entregues precisam ser trazidos de um ou mais depósitos e os bens a serem recolhidos precisam ser transportados para um ou mais depósitos.

- Problemas de Roteamento de Veículos com Coleta e Entrega (*Vehicle Routing Problems with Pickup and Delivery*, VRPPD): Compreendem problemas em que os bens são transportados entre consumidores de coleta e consumidores de entrega sem a necessidade da intermediação do armazém.

Dentro da classe VRPB, tem-se quatro subclasses. Nas duas primeiras, consumidores recebem ou entregam carga, mas não as duas operações ao mesmo tempo. Nas duas últimas, consumidores podem receber e entregar mercadorias:

- Problema de Roteamento de Veículos com *Backhauls* Agrupados (*VRP with Clustered Backhauls*, VRPCB). O grupo de consumidores de entrega (consumidores que apenas recebem) deve ser visitado antes que qualquer consumidor de coleta (consumidores que apenas entregam) seja visitado.

- Problema de Roteamento de Veículos com *Linehauls* e *Backhauls* Mistos (*VRP with Mixed Linehauls and Backhauls*, VRPMB). Os consumidores de coleta e de entrega podem ser visitados de maneira mista e cada consumidor só pode ser visitado uma vez.

- Problema de Roteamento de Veículos com Entrega e Coleta Divisíveis (*VRP with Divisible Delivery and Pickup*, VRPDDP). Consumidores estão associados com uma demanda de coleta e uma demanda de entrega ao mesmo tempo, porém não precisam ser visitados apenas uma vez. Pode haver duas visitas, uma para a coleta e outra para a entrega.

- Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas (*VRP with Simultaneous Pickup and Delivery*, VRPSPD). Consumidores estão associados com uma demanda de coleta e uma demanda de entrega ao mesmo tempo e não podem ser visitados mais de uma vez. Trata-se do problema abordado neste trabalho e será discutido mais detalhadamente no capítulo 3.

Dentro da Classe VRPPD, tem-se duas subclasses:

- Não emparelhada. Pontos de coleta e entrega não são emparelhados, isto é, é considerado um bem homogêneo e cada unidade coletada pode ser utilizada para satisfazer a demanda de qualquer consumidor de entrega. Nessa classe está o Problema de Roteamento de Veículos de Coleta e Entrega (*Pickup and Delivery VRP*, PDVRP), em que cada unidade coletada pode ser usada para preencher a demanda de qualquer consumidor de entrega. Cada consumidor pode ser visitado apenas uma vez.

- Emparelhada. Problemas em que os pedidos de transporte tem uma origem e um destino associado, resultando em pontos de coleta e entrega emparelhados. Nessa classe tem-se:
  - Problema de Coleta e Entrega (*Pickup and Delivery Problem*, PDP).
  - Problema *Dial-a-Ride* (*Dial-a-Ride Problem*, DARP).

Tanto o PDP quanto o DARP consideram pedidos de transporte associados com uma origem e um destino. O PDP lida com o transporte de bens enquanto o DARP lida com transporte de passageiros. A diferença são restrições ou objetivos que levam a conveniência dos passageiros em conta (por exemplo o tempo máximo de viagem ou janelas de tempo para atendimento).

Capítulo 3

O Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas

Em muitas aplicações em sistemas com distribuição/redistribuição, os consumidores possuem ao mesmo tempo uma demanda de coleta e de entrega. Eles podem não aceitar ser atendidos separadamente para a coleta e para a entrega, devido aos esforços necessários para cada atividade. Esses esforços podem ser consideravelmente reduzidos com uma atividade simultânea de coleta e entrega. Consequentemente, os consumidores querem ser servidos em uma única parada [17, 32].

O Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas (VRPSPD) é definido sobre um grafo completo $G = (V, E)$. Há N_c consumidores, representados pelos vértices 1 a $N_c = |V| - 1$, um depósito, representado pelo vértice 0, e um conjunto de veículos com capacidade de carga Q. Cada consumidor i é associado à uma quantidade de entrega d_i e uma quantidade de coleta p_i. Um custo c_{ij} é associado à cada aresta $e = (i, j) \in E$. Uma rota é uma sequência $r = \{i_0, i_1, \ldots, i_{N_r-1}\}$ de consumidores que inicia e termina no depósito (i.e. $i_0 = i_{N_r-1} = 0$), sendo $N_r = |r|$. Uma rota viável é uma rota r tal que a carga máxima do veículo não é violada (i.e. $\sum_{0<k<N_r} d_{i_k} \leq Q$, $\sum_{0<k<N_r} p_{i_k} \leq Q$, e $\sum_{0<j\leq k} p_{i_j} + \sum_{k<j<N_r} d_{i_j} \leq Q$, $0 < k < N_r$), e nenhum consumidor é visitado mais de uma vez (i.e. $i_j \neq i_k$, $0 \leq j, k < N_r$). O custo c_r de uma rota r é dado pela soma dos custos das arestas conectando seus consumidores (i.e. $c_r = \sum_{j<N_r} c_{i_j i_{j+1}}$). O problema consiste em encontrar um conjunto R de rotas viáveis de custo mínimo, tal que cada consumidor seja visitado por exatamente uma rota.

Quando se tem apenas um veículo, o problema é denominado Problema do Cai-

xeiro Viajante com Coleta e Entrega Simultâneas (*Traveling Salesman Problem with Simultaneous Pickup and Delivery*, TSPSDP) [17, 23].

Se a quantidade de coleta em cada consumidor for menor ou igual à quantidade de entrega, o problema se reduz ao Problema de Roteamento de Veículos Capacitado, que é NP-difícil [15].

3.1 Algumas Propriedades Importantes

Nesta seção, são reunidas algumas propriedades que podem ajudar no desenvolvimento de algoritmos mais eficientes para o problema. Seja $\{i : i \in V, p_i > d_i\}$ os consumidores de coleta e $\{i : i \in V, d_i \geq p_i\}$ os consumidores de entrega. Dada uma rota $r = \{i_0, i_1, \ldots, i_{N_r}\}$, seja $q^r_{i_k}$ a carga do veículo logo após atender o consumidor i_k.

Propriedade 1. Dado um conjunto de consumidores $V' \subseteq V$, se $\max\{\sum_{i \in V'} d_i, \sum_{i \in V'} p_i\} \leq Q$, existe pelo menos uma rota viável sobre esses consumidores.

Demonstração. Basta atender primeiro todos os consumidores de entrega e depois os de coleta em V', em qualquer ordem. O veículo parte do depósito com carga inicial igual a $\sum_{i \in V'} d_i \leq Q$ e até que tenha atendido todos os consumidores de entrega, sua carga apenas diminuirá ou permanecerá a mesma, portanto, a carga nesse ponto é viável. Após isso, a carga do veículo apenas aumentará ou continuará a mesma e o veículo retornará ao depósito com carga $\sum_{i \in V'} p_i \leq Q$. □

Propriedade 2. Seja r uma rota fracamente viável, isto é, uma rota em que $\max\{\sum_{i \in r} d_i, \sum_{i \in r} p_i\} \leq Q$, mas que possivelmente tenha alguma violação de capacidade em seu percurso, isto é, pode existir $i \in r$ tal que $q^r_i > Q$. Seja $r = \{i_0, i_a, \ldots, i_b, i_c, \ldots, i_d, i_{N_r}\}$, uma rota fracamente viável em que a carga máxima $q^r > Q$ do veículo se dê logo após o atendimento do consumidor i_b ($q^r = q^r_{i_b}$). A inserção da sequência de consumidores $\{i_c, \ldots, i_d\}$ no início da rota gera uma rota viável $r' = \{i_0, i_c, \ldots, i_d, i_a, \ldots, i_b, i_{N_r}\}$ (veja [33]).

Demonstração. O veículo retorna ao depósito com carga $P = \sum_{i \in r} p_i \leq Q$, portanto, a sequência de consumidores $\{i_c, \ldots, i_d\}$ tem carga líquida $\delta = P - q^r < 0$. Dessa maneira, após passar pela sequência $\{i_c, \ldots, i_d\}$ em r', as quantidades de carga após o atendimento de cada consumidor na sequência $\{i_a, \ldots, i_b\}$ serão alteradas em δ ($q^{r'}_{i_k} = q^r_{i_k} + \delta$, $a \leq k \leq b$) e a carga após i_b será $q^{r'}_{i_b} = q_r + \delta = P \leq Q$. Uma nova violação não será produzida na sequência $\{i_c, \ldots, i_d\}$, pois cada sequência $\{i_c, \ldots, i_k\}, k \leq d$, tem carga líquida negativa, caso contrário q^r não seria máximo em r. □

Propriedade 3. Dada uma rota r sobre um conjunto de consumidores V' tal que r atende primeiro todos consumidores de entrega e depois os de coleta, em qualquer ordem, e seja i o último consumidor de entrega atendido na rota, nenhuma rota r' sobre V' terá carga menor que q_i^r, isto é, para qualquer $i' \in r'$, $q_{i'}^{r'} \geq q_i^r$.

Demonstração. Seja $r' = \{i_0, i_1, \ldots, i_{N_r}\}$ uma rota em que os consumidores são atendidos em uma ordem diferente de atender primeiro todos os consumidores de entrega. Seja i_k o primeiro consumidor de coleta atendido na rota. As cargas $q_{i_j}^{r'}$, $j < k$ podem ser reescritas como $q_{i_j}^{r'} = \hat{q}_{i_j}^{r'} + d_{i_k}$ e as cargas $q_{i_j}^{r'}$, $j > k$ podem ser reescritas como $q_{i_j}^{r'} = \hat{q}_{i_j}^{r'} + p_{i_k}$. Se i_k passar a ser atendido na última posição da rota, as cargas $q_{i_j}^{r'} = \hat{q}_{i_j}^{r'} + d_{i_k}$, $j < k$ permanecerão inalteradas, porém, as cargas $q_{i_j}^{r'} = \hat{q}_{i_j}^{r'} + p_{i_k}$, $j > k$ diminuirão para $\hat{q}_{i_j}^{r'} + d_{i_k}$, uma vez que $p_{i_k} > d_{i_k}$. Repetindo esse processo enquanto a rota não for uma rota que atende primeiro todos os consumidores de entrega, a carga mínima ao logo da rota ou diminui ou permanece inalterada. Quaisquer rotas que atendam primeiro todos os consumidores de entrega, têm carga mínima igual. Seja i_k o último consumidor de entrega, até esse consumidor a carga do veículo diminui ou não se altera, após ele a carga aumenta ou não se altera, $q_{i_k} = \sum_{j \leq k} p_j + \sum_{j > k} d_j$ em qualquer dessas rotas. □

Propriedade 4. Dada uma rota r sobre um conjunto de consumidores V' tal que r atende primeiro todos os consumidores de coleta e depois os de entrega, em qualquer ordem, e seja i o último consumidor de coleta atendido na rota, nenhuma rota r' sobre V' terá carga maior que q_i^r, isto é, para qualquer $i' \in r'$, $q_{i'}^{r'} \leq q_i^r$.

Demonstração. Semelhante à demonstração da propriedade 3, porém levando consumidores de coleta para a posição imediatamente à frente do depósito até que se torne uma rota que atenda primeiro todos os consumidores de coleta. □

As propriedades 1, 3 e 4 serão mencionadas na seção 4.1.1, quando a geração de soluções iniciais para o problema for discutida.

Propriedade 5. Sejam I e I' instâncias do VRPSPD definidas sobre o mesmo grafo, custos dos arcos e capacidade do veículo, mas com quantidades de coleta e entrega em cada veículo trocadas ($d_i' = p_i, p_i' = d_i$). Dada uma solução viável s para I, a solução s', com as rotas de s percorridas em sentido contrário, é viável em I'

Demonstração. Suponha uma rota viável r em s com N_r arestas e N_r consumidores (incluindo o depósito). Considere as N_r situações da carga do veículo nesta rota, isto é, q_i^r, $i \in r$. Como r é viável, $q_i^r \leq Q$, $i \in r$. Em r o veículo inicia a rota com carga $D_r = \sum_{i \in r} d_i$ e termina com carga $P_r = \sum_{i \in r} p_i$. Na rota r', correspondente a r

no sentido reverso, o veículo inicia a rota com carga $D_{r'} = P_r = q^r_{i_{N_r-1}}$, pois como as demandas são trocadas, terá que entregar toda a coleta de r. Quando o veículo r' passa pelo primeiro consumidor, penúltimo de r, entrega o que foi coletado e coleta o que foi entregue, ficando com carga $q^{r'}_{i_1} = q^r_{i_{N_r-2}}$. Ao prosseguir para o segundo consumidor, o mesmo acotecerá, e a carga voltará a situação $q^r_{i_{N_r-3}}$. O veículo continua seu percurso, desfazendo operações que foram feitas em r, repetindo as situações de N_r até 0 de r, todas viáveis. Portanto r' é viável. □

Esse resultado mostra que para qualquer solução para I existe uma solução equivalente, com o mesmo custo, mas em sentido contrário, para I'. Nas instâncias de Salhi e Nagy [42], largamente adotadas na literatura, as instâncias do tipo X e Y representam essa situação, a única diferença entre elas são demandas de coleta e entrega trocadas em cada consumidor. Porém, a maioria das heurísticas na literatura produzem um resultado para X (Y) e não são capazes de encontrar o mesmo resultado em Y (X), o que sugere que dado um problema a ser resolvido, uma heurística pode ser capaz de encontrar um resultado melhor se as demandas de coleta e entrega forem invertidas.

3.2 Trabalhos Anteriores

3.2.1 Heurísticas Construtivas

O VRPSPD foi introduzido por Min [30], trabalho no qual é desenvolvida uma heurística *Cluster-First Route-Second* para o problema. Inicialmente, os consumidores são agrupados e o TSP é resolvido em cada grupo. Caso ocorram inviabilidades, os arcos sobrecaregados são penalizados e o TSP é resolvido novamente, esse processo é repetido até que a solução seja viável.

Dethloff [17] propõe uma heurística de inserção utilizando um critério de inserção que leva em conta outros fatores além da tradicional distância adicional: capacidade residual de coleta e de entrega e sobrecarga de distância radial. Assis [3] propõe uma extensão à essa heurística em que utiliza a ferramenta de decisão multicritério PROMETHEE para decidir se deve e qual consumidor inserir na rota corrente, ou se deve criar uma nova rota para algum consumidor. Xie et al. [49] também estendem estendem o trabalho de Dethloff [17] levando outros fatores em consideração.

Salhi e Nagy [42] propõem heurísticas de inserção em que primeiro os consumidores de entrega são roteados e depois estudam a inserção de consumidores de coleta um a um, em pares ou em grupos.

No trabalho de Montané e Galvão [31] são apresentadas duas heurísticas construtivas. Em uma delas, inicialmente é criado um ciclo para o TSP, que é particionado de maneira sequencial para a criação de grupos. Na outra, é utilizada a heurística *Sweep* para a divisão dos consumidores em grupos. No algoritmo *Sweep*, inicialmente desenvolvido para o VRP, grupos são formados através da rotação de um raio centrado no depósito. Em ambas as heurísticas, após a divisão dos consumidores, é usado um algoritmo para o TSPSPD para a criação das rotas em cada grupo.

Assis [3] propõe heurísticas *Cluster-First Route-Second* em que os consumidores são agrupados com algoritmos baseados em algoritmos para Árvores Geradoras Mínimas.

Bianchessi e Righini [5] estendem algoritmos *Route-First Cluster-Second*, propostos por Mosheiov [34] para um problema semelhante, com novas possibilidades de particionamento.

Nagy e Salhi [35] apresentaram uma heurística rotear e dividir em que uma solução inicial é construída utilizando o conceito de *Multiple Giant Tour* e depois melhorada utilizando diversos operadores. Em [3] também são apresentados dois métodos *Route-First Cluster-Second*, baseados no TSP.

3.2.2 Metaheurísticas

Vural [47] resolve o problema sob o ponto de vista de um problema de sequenciamento de máquinas, utilizando Algoritmo Genético baseado na estratégia de *Random Keys* e em uma estratégia de melhoria. Um algoritmo genético também é desenvolvido no trabalho de Li e Zhang [28]. a população inicial é gerada com heurísticas construtivas, um novo mecanismo de *crossover* é proposto, a substituição das gerações anteriores é feita por um mecanismo de *ranking replacement* e 2-opt é usado para melhoria das soluções.

Gökçe [24] desenvolve um método baseado em Colônia de Formigas. As trilhas iniciais de feromônio são baseadas em uma solução construída com um algoritmo de inserção. Então, soluções são construídas por cada formiga. Após isso, cada solução é melhorada com um procedimento 2-opt, as informações de feromônio são atualizadas e novas soluções são construídas. Chen et al. [12] e Zhang et al. [51] também propõem algoritmos baseados em Colônias de Formigas, um procedimento de inserção e 2-opt são usados para a contrução de rotas pelo primeiro, enquanto fatores propostos por Dethloff [17] são usados na construção das rotas pelo segundo.

Bianchessi e Righini [5] propõem heurísticas construtivas. Busca Tabu é usada com vizinhanças simples, vizinhanças complexas e vizinhanças variáveis. É implemen-

tado um mecanismo reativo para controlar o tamanho das Listas Tabu. Busca Tabu também é utilizada por diversos outros autores. Chen [10] usam *Simulated Annealing*, Listas Tabu e algumas vizinhanças para melhorar uma solução inicial criada com uma heurística de inserção. Chen e Wu [11] propõem uma heurística híbrida entre Busca Tabu e *Record-to-Record Travel*. Soluções iniciais são geradas utilizando uma heurística de inserção e a geração de soluções adjacentes é feita através de algumas vizinhanças. Montané e Galvão [32] desenvolvem uma Busca Tabu, que utiliza as heurísticas propostas em [31] para a geração de soluções iniciais e os movimentos são efetuados em algumas vizinhanças. Wassan et al. [48] desenvolvem uma Busca Tabu para o VRPSPD e para um problema com coleta e entrega mistas em que um mecanismo reativo é implementado para controlar o tamanho das Listas Tabu. Um híbrido entre Busca Tabu e *Guided Local Search* (GLS) é proposto por Zachariads et al. [50]. É proposta uma nova maneira para escolha do arco penalizado e para a penalização. A Busca Tabu se movimenta no espaço com função objetivo penalizada do GLS, a cada movimento da Busca Tabu é feita a penalização GLS.

Ropke e Pisinger [40] tratam o VRPSPD e outros problemas sob a perspectiva de um problema de coleta e entrega mais geral e desenvolvem uma heurística baseada em *Large Neighborhood Search* para esse problema mais geral.

Assis [3] desenvolve, além de diversas heurísticas de contrução, uma heurística baseada em *Greedy Randomized Adaptive Search Procedure* (GRASP), utilizando diversas vizinhanças. GRASP também é utilizado por Subramanian et al. [43], que implementam um híbrido com *Iterated Local Search* (ILS) e *Variable Neighborhood Descent* (VND). Em cada iteração do GRASP a solução é melhorada com ILS/VND.

3.2.3 Métodos Exatos

Quando problemas de roteamento de veículos são resolvidos por Geração de Colunas, o Subproblema de Precificação geralmente pode ser formulado como um Problema de Caminho Elementar Mínimo com Restrições de Recursos (*Elementary Resource Constrained Shortest Path Problem*, ERCSPP) [6, 16, 39], que consiste em encontrar o menor caminho sem ciclos entre dois vértices específicos em um grafo com arcos possivelmente negativos, de maneira que o consumo de determinados recursos não ultrapasse as quantidades disponíveis. Por se tratar de um problema NP-difícil, na maioria das vezes é resolvida uma relaxação, o Problema de Caminho Mínimo com Restrições de Recursos (*Resource Constrained Shortest Path Problem*, RCSPP)[6, 16, 39], onde são permitidos ciclos no caminho. No trabalho de Salani [41] são propostas novas estratégias para a resolução deste problema, tais como Busca Bi-direcional, *Decremental State Space*

Relaxation (Estratégia desenvolvida paralelamente por Boland et al. [6] com o nome *State Space Augmenting Algorithm*), e estratégias de poda. Além disso, é implementado um algoritmo *Branch-and-price* para alguns tipos de problemas de roteamento, inclusive para o VRPSPD. A resolução do Subproblema de maneira exata e relaxada são comparadas.

O algoritmo *Branch-and-price* implementado por Salani é inicializado com um conjunto de colunas gerado por uma heurística Busca Tabu. Para a geração de colunas, antes da execução do algoritmo exato de Programação Dinâmica, são utilizadas heurísticas. Algumas dessas heurísticas são simples heurísticas gulosas. Uma outra heurística utiliza a regra simples de eliminação de caminhos parciais do RCSPP no ERCSPP. Também é utilizada uma heurística que remove alguns arcos do grafo caso seus custos reduzidos ultrapassem um determinado limiar. O algoritmo implementado faz uso do método de Estabilização de Pontos Interiores. A estratégia de *branching* utilizada consiste em determinar um vértice atendido por rotas fracionárias, então, parte dos arcos que passam por esse vértice é proibída em um nó filho e a outra parte é proibída em outro. Quatro implementações são comparadas. A primeira resolve o Subproblema relaxado. A segunda resolve o Subproblema relaxado e são incorporados cortes *2-path* no *Branch-and-price*. A terceira resolve o Subproblema de maneira exata. A quarta resolve o Subproblema e maneira exata e também incorpora cortes *2-path*, essa implementação é a que obtém os melhores resultados na resolução do VRPSPD é utilizada para a comparação com o algoritmo *Branch-and-price* implementado no presente trabalho.

Capítulo 4

Heurísticas Para o VRPSPD

Neste capítulo, são discutidas e testadas as heurísticas desenvolvidas. Estas heurísticas são baseadas nas metaheurísticas *Greedy Ranzomized Adaptive Procedures*, *Iterated Local Search*, *Variable Neighborhood Descent*, *Simulated Annealing* e *Guided Local Search*. Nas seções seguintes, estas metaheurísticas são introduzidas conforme necessário para definição das heurísticas desenvolvidas.

4.1 GRASP

O *Greedy Ranzomized Adaptive Procedures* (GRASP) [20] é uma metaheurística que pode ser vista como uma técnica de amostragem do espaço de soluções. Cada iteração do GRASP consiste em uma fase de construção, na qual uma solução é construída de maneira aleatorizada, e uma fase de busca local, na qual a solução inicial é melhorada. Esse procedimento é repetido até que determinado critério de parada seja satisfeito e, então, a melhor solução encontrada é retornada. O algoritmo 1 descreve o arcabouço GRASP.

algoritmo 1 GRASP

1: $f(s^*) = \infty$
2: **enquanto** o critério de parada não for satisfeito **faça**
3: construa uma solução s de maneira aleatorizada.
4: encontre um mínimo local s' partindo de s.
5: **se** $f(s') < f(s^*)$ **então**
6: $s^* = s'$
7: **fim se**
8: **fim enquanto**
9: **retorne** s^*

4.1.1 Solução Inicial

Uma solução inicial de boa qualidade pode acelerar o processo de busca local. Para gerar soluções iniciais, usamos uma heurística *Cluster-First Route-Second* baseada no algoritmo de Kruskal, proposta por Assis em [3]. Esse algoritmo foi escolhido devido à sua simplicidade, velocidade computacional e qualidade das soluções.

Inicialmente os consumidores são agrupados e depois são geradas rotas para cada grupo. Duas condições podem ser usadas durante a construção dos grupos:

1. Uma rota que visita todos os consumidores de entrega antes dos consumidores de coleta no grupo g é viável (ver Propriedades 1 e 3). Seja M_g^1 a carga máxima nessa rota.

2. Uma rota que visita todos os consumidores de coleta antes dos consumidores de entrega no grupo g é viável (ver Propriedade 4). Seja M_g^2 a carga máxima nessa rota.

Se grupos forem criados usando a condição 1, os consumidores em cada grupo resultante devem ser roteados usando algum algoritmo para o TSPSDP. Se a condição 2 for usada, como M_g^2 é a carga máxima possível em qualquer rota usando os consumidores do grupo g, qualquer algoritmo para o TSP pode ser usado.

A heurística usada baseia-se no algoritmo de Kruskal para a construção dos grupos. Inicialmente cada grupo consiste em apenas um consumidor. Em cada passo, o algoritmo analisa as arestas do grafo em ordem não decrescente e caso os consumidores da aresta pertençam a grupos diferentes e a união desses grupos obedeça a condição sendo usada, os grupos são unidos. Para que gere soluções iniciais aleatórias, antes da fase de agrupamento, os pesos das arestas são modificados com um parâmetro $\alpha \in [0, 1]$. O Algoritmo 2 descreve o procedimento, $c_{\max}$ e $c_{\min}$ denotam o maior e menor custo de aresta, respectivamente e M_g^{con} denota a carga máxima na rota sobre o grupo g de acordo com a condição $con \in \{1, 2\}$ sendo usada.

A implementação desenvolvida faz uso da condição 2 e da heurística Inserção Mais Próxima para o TSP (algoritmo 3) para a criação de rotas para cada grupo.

4.1.2 Vizinhanças

Dada uma solução $s \in S$, em que S é o espaço de soluções viáveis, uma vizinhança $\mathcal{N}(s)$ é composta de soluções obtidas pela aplicação de uma mudança elementar bem definida (movimento) em s. Uma solução $s \in S$ é um mínimo global (ótimo global) se

algoritmo 2 DKRUSKAL

Entrada: Instância do Problema
Saída: Conjunto R de rotas viáveis

1: **para todo** $(i, j) \in E$ **faça**
2: $a =$ número aleatório no intervalo $(-\alpha(c_{\max} - c_{\min}), \alpha(c_{\max} - c_{\min}))$
3: $\hat{c}_{ij} = \max(0, c_{ij} + a)$
4: **fim para**
5: Criar um conjunto E' com as arestas em E ordenadas em ordem não decrescente de acordo com os pesos $\hat{c}_{ij}$, $(i, j) \in E$.
6: Criar um conjunto de grupos $G = \emptyset$
7: Para cada consumidor $i \in V$, criar um grupo g_i o contendo e adicionar g_i em G
8: **para** cada $(i, j) \in E'$ em ordem **faça**
9: **se** $g_i \neq g_j$ e $M^{con}_{g_i \cup g_j} \leq Q$ **então**
10: $G = G \setminus \{g_i, g_j\}$
11: Unir g_i e g_j e adicionar o conjunto resultante a G
12: **fim se**
13: **fim para**
14: Criar um conjunto de rotas $R = \emptyset$
15: **para** cada $g \in G$ **faça**
16: Criar uma rota viável r com os consumidores de g (Algoritmo 3).
17: $R = R \cup \{r\}$
18: **fim para**
19: **retorne** R

algoritmo 3 Inserção Mais Próxima

Entrada: Grupo g
Saída: Rota r

1: $r = \{i_0 = 0, i_1 = 0\}$
2: $u = i_0$
3: **enquanto** $g \neq \emptyset$ **faça**
4: Encontrar a aresta (u, i), $i \in g$ de mínimo custo
5: Inserir i após u em r
6: $g = g \setminus \{i\}$
7: $u = i$
8: **fim enquanto**
9: **retorne** r

não houver solução $s' \in S$ tal que $f(s') < f(s)$. uma solução $s \in S$ é um mínimo local (ótimo local) se não houver solução $s' \in \mathcal{N}(s)$ tal que $f(s') < f(s)$.

Usamos as seguintes vizinhanças para busca local:

1. *2-Opt.* Essa vizinhança consiste em soluções obtidas quando duas arestas não adjacentes de alguma rota são removidas e a rota é reconectada pela adição de outras duas arestas.

2. *3-Opt.* Consiste em soluções obtidas pela remoção de três arestas não adjacentes de alguma rota e a substituição dessas três arestas por outras três arestas de modo que a rota seja reconectada.

3. *Or-Opt*(n). Uma sequência de n consumidores de uma mesma rota é movida para outra posição da mesma rota. Foram usadas sequências de tamanhos 1, 2 e 3 consumidores.

4. *Route Inversion.* Consiste em soluções nas quais consumidores são servidos em ordem reversa. Essa vizinhança é usada apenas na tentativa de reduzir a carga máxima da rota.

5. *Shift*(n). Essa vizinhança consiste em soluções obtidas quando uma sequência de n consumidores de uma rota é transferida para alguma outra rota.

6. *Swap*(n, m). É composta pelas soluções obtidas pela remoção de uma sequência s_1 de n consumidores e de uma sequência s_2 de tamanho m, de duas rotas diferentes r_1 e r_2, respectivamente, seguida pela inserção de s_1 em alguma posição de r_2 e inserção de s_2 em alguma posição de r_1.

7. *Crossover.* Consiste em soluções obtidas pela remoção de uma aresta (i, j) de uma rota r_1 e de uma aresta (k, l) de uma outra rota r_2, seguida pela inserção das arestas (i, l) e (k, j) na solução. Isto é, r_1 e r_2 são substituídas por r'_1 e r'_2, em que r'_1 é formada pela primeira parte de r_1 (de 0 até o vértice i) e pela segunda parte de r_2 (de l até o depósito), e r'_2 é formada pela primeira parte de r_2 (de 0 até o vértice k) e pela segunda parte de r_1 (de j até o depósito).

4.1.3 Busca Local

Para a fase de busca local do GRASP, usamos o *Variable Neighborhood Descent* (VND) [25]. O VND é uma metaheurística na qual a idéia básica consiste na combinação de busca local (*Descents*) em diversas vizinhanças, baseada no fato de que um mínimo

local em uma vizinhança não é necessariamente um mínimo local em outra vizinhança. Assim, o VND realiza a troca sistemática de vizinhanças durante a busca local. Ao fim do procedimento, o VND alcança uma solução que é um mínimo local em todas as vizinhanças usadas e, consequentemente, com boas chances de ser uma boa solução.

O VND é uma variante da estratégia *Variable Neighborhood Search* (VNS), cuja idéia básica é a troca sistemática de vizinhanças, e da qual diversas outras metaheurísticas são derivadas.

O Algoritmo 4 exibe o procedimento básico do VND.

algoritmo 4 VND(s)

1: $k = 1$
2: **enquanto** $k \leq k_{\max}$ **faça**
3: $s' =$ melhor vizinho de s em $\mathcal{N}_k$
4: **se** s' não for melhor que s **então**
5: $k = k + 1$
6: **senão**
7: $k = 1$
8: **fim se**
9: **fim enquanto**

Neste trabalho, é adotada uma estratégia ligeiramente diferente do arcabouço básico. Ao invés de realizar apenas um movimento em direção ao melhor vizinho na vizinhança $\mathcal{N}_k$ (linha 3 do algoritmo), nas heurísticas desenvolvidas é feita uma busca local completa em $\mathcal{N}_k$, com movimentos sempre em direção ao melhor vizinho. As vizinhanças são usadas na seguinte sequência:

1. Route Inversion.
2. Or-Opt(3).
3. Or-Opt(2).
4. Or-Opt(1).
5. 2-Opt.
6. 3-Opt.
7. Crossover.
8. Shift(3).
9. Shift(2).
10. Shift(1).
11. Swap(2,2).
12. Swap(2,1).
13. Swap(1,1).

Essa sequência foi determinada basicamente por dois critérios. O primeiro é usar vizinhanças intra-rota primeiro e depois vizinhanças inter-rotas. Vizinhanças intra-rotas são mais baratas computacionalmente e reorganizam as rotas após mudanças

inter-rotas. O segundo é usar primeiro vizinhanças que trabalham sobre sequências maiores de consumidores, já que, após a busca local ter sido realizada em vizinhanças que trabalham sobre sequências menores, pode ser mais difícil encontrar bons vizinhos nas vizinhanças definidas sobre sequências maiores.

4.2 ILS

Dado o espaço de soluções viáveis S e uma função $BL : S \rightarrow S$, tal que dada $s \in S$ e $s^* = BL(s)$ tem-se $f(s) \geq f(s^*)$. A idéia básica do *Iterated Local Search* (ILS) [29] é fazer uma busca estocástica no espaço $S^* = BL(S)$. Isto é, se a função BL é um procedimento de busca local, o método ILS realiza uma busca pelo espaço de mínimos locais dado pela função. O ILS mantém uma solução corrente $s^* \in S^*$ e, a cada passo, encontra uma nova solução $\hat{s}^* \in S^*$ pela geração de uma solução perturbada $s^{*'} \in S$ a partir de s^*, seguido pela aplicação de BL em $s^{*\prime}$ para gerar $\hat{s}^* \in S*$. Nesse ponto, $\hat{s}^*$ pode ou não ser aceita como solução corrente. O ideal seria que esse processo gerasse uma busca tendenciosa pelo espaço de busca. Porém, perturbações muito fortes da solução atual farão com que o espaço de soluções S^* seja explorado de maneira aleatória, levando a um algoritmo *Random Restart*, enquanto perturbações fracas podem fazer com que a busca volte à solução corrente e poucas novas soluções sejam exploradas. O arcabouço ILS é exibido no Algoritmo 5.

algoritmo 5 ILS

```
1: s_0 = SolucaoInicial
2: s* = BL(s_0)
3: repita
4:    s*' = Perturbacao(s*, história)
5:    ŝ* = BL(s*')
6:    se CriterioDeAceitacao(s*, ŝ*, história) então
7:       s* = ŝ*
8:    fim se
9: até que o critério de parada seja satisfeito
```

O ponto de partida para a busca ILS no algoritmo desenvolvido é uma solução gerada com o GRASP discutido na seção 4.1. A função BL define o espaço de busca S^* a ser explorado e assim, é crucial para a qualidade das soluções. Usamos como função de busca local a implementação VND discutida na seção 4.1.3.

A aceitação de novas soluções como solução corrente e perturbações guiam a busca pelo espaço S^*. A abordagem mais comum seria não levar em consideração nenhuma informação sobre a história da busca nesse processo e aceitar apenas soluções

com qualidade melhor. Entretanto, mecanismos mais completos podem levar a procedimentos de busca mais robustos. São usados três tipos de perturbação, com uma estratégia para evitar que o método fique encurralado em soluções específicas, essas perturbações são discutidas na seção 4.2.1. Três estratégias diferentes para a troca da solução corrente são testadas, essas estratégias são discutidas na seção 4.2.2.

4.2.1 Perturbações

Para perturbações, selecionamos vizinhos aleatórios alternadamente nas seguintes vizinhanças:

- $H(p)$. Essa vizinhança consiste nas soluções nas quais até p consumidores da solução corrente estão em posições diferentes da solução e foi proposta em [36]. Na perturbação, p elementos são removidos aleatoriamente da solução e reinseridos em alguma posição viável. Caso não seja possível reinserir algum consumidor, uma nova rota é criada para servi-lo. O mesmo consumidor pode ser considerado mais de uma vez.

- *Relocate*(p). É caracterizada pelas soluções nas quais até p elementos, em cada rota, estão em posições diferentes da mesma rota. Na perturbabação, em cada rota, por p vezes algum consumidor aleatório é removido e é realizada a tentativa de inserí-lo em outra posição viável da mesma rota (se não houver posição viável, o consumidor é reinserido em sua posição original). O mesmo consumidor pode ser considerado mais de uma vez.

- *Swap*(p). Vizinhança Swap$(1, 1)$ discutida na seção 4.1.2. São feitos p movimentos aleatórios nessa vizinhança.

O uso de movimentos aleatórios nas perturbações evita a ciclagem. A definição de um parâmetro p pode depender da instância, p grande fará com que o algoritmo ande aleatoriamente pelo espaço de mínimos locais, p pequeno fará com que o algoritmo volte à mesma solução após a busca local. Para aliviar esse problema, usamos uma função $\omega(n)$, que retorna um inteiro i no intervalo $[1, n]$ com probabilidade $(n - i + 1)^2 / \sum_{j=1}^{n} j^2$. Isto é, a probabilidade de escolher o elemento i decresce quadraticamente com o aumento de seu valor. Intervalos $[p_{\min}, p_{\max}]$ são definidos para cada tipo de perturbação. Durante a fase de perturbação, um número p é retirado desse intervalo da seguinte maneira, $p = (p_{\min} - 1 + \omega(p_{\max} + 1))$. Esses intervalos são estabelecidos com o uso de porcentagens de consumidores na instância, por exemplo, dada uma instância de N_c consumidores, podem ser usados $p_{\min} = 0,1 \times N_c$ and $p_{\max} = 0,5 \times N_c$.

Com essa estratégia, diferentes tamanhos de perturbação, dentro de um intervalo pré estabelecido, serão usados, evitando que a busca fique presa em uma determinada solução por muito tempo e reduzindo a necessidade de otimizar o parâmetro p de acordo com a instância que está sendo resolvida.

4.2.2 Critério de Aceitação

Três critérios de aceitação foram testados:

1. *Aceitar se for melhor.* A nova solução $\hat{s}^*$ é aceita caso tenha custo total menor que a solução corrente s^*.

2. *Aceitar de acordo com SA.* Simulated Annealing (SA) [26] [21] é uma metaheuristica inspirada em processos de resfriamento de certos materiais. No SA, a solução $\hat{s}^*$ é aceita como solução corrente se possuir menor custo, ou com probabilidade $\exp^{-\delta/t}$ caso contrário, sendo $\delta = f(\hat{s}^*) - f(s^*)$. Assim, a aceitação da solução depende apenas de sua qualidade e da temperatura t. Com a probabilidade de soluções piorantes serem aceitas, o SA escapa de mínimos locais. Entretanto, ao longo do algoritmo, a temperatura decresce, tornando menos provável a aceitação dessas soluções de piora e intensificando a busca.

   Em nossa implementação para a obtenção da temperatura inicial t_0, uma solução é gerada com $p = p_{\max}$ em cada perturbação e t_0 é calculado tal que a pior dessas soluções tenha 95% de chance de ser aceita ($t_0 = \delta \log_{20/19} \exp$). A temperatura é atualizada a cada iteração ILS, ao fim da fase de aceitação faz-se $t = t \times \gamma$, em que γ é um parâmetro do algoritmo. Diferentes valores de γ foram testados e são discutidos na seção 4.3.

3. *Aceitar de acordo com estratégia baseada em GLS.* O Guided Local Search [46] é uma metaheurística que conduz algoritmos de busca local pelo espaço de busca. Toda vez que o algoritmo encontra um mínimo local, uma característica desta solução é selecionada e soluções que possuem esta característica têm seu custo penalizado. A função objetivo $f(s)$ é substituída por $h(s) = f(s) + \lambda \sum(p_i I_i(s))$, em que p_i é a penalidade associada a característica i, $I_i(s)$ é uma variável binária que assume valor 1 caso a solução possua a característica i e valor 0 caso contrário, e λ é um parâmetro do algoritmo. Inicialmente, todas as penalidades são ajustadas em 0. Ao se deparar com um mínimo local s^*, um valor $u_i(s^*) = I_i(s^*)c_i/(1+p_i)$, que define a utilidade de se penalizar cada característica i, é calculado. A característica com maior valor u tem sua penalidade aumentada por uma unidade.

Esse mecanismo penaliza características com alto custo na solução corrente e que não tenham sido muito penalizadas durante a execução do algoritmo.

Usamos o GLS para guiar o ILS. A estratégia consiste em aceitar a nova solução como solução corrente se $h(\hat{s}^*) \leq h(s^*)$. Isto é, caso $\hat{s}^*$ for uma solução com bom custo e características atrativas (não muito penalizadas e, portanto, pouco exploradas). Quando $\hat{s}^*$ é aceita como solução corrente, as penalidades são atualizadas. A busca local e perturbação são feitas usando função objetivo simples. Com essa estratégia, tenta-se introduzir alguma memória na caminhada do ILS pelo espaço de busca S^*.

Diferentes valores de λ são testados na seção 4.3.

4.3 Resultados Computacionais

Sumarizando o que foi discutido nas seções anteriores: três heurísticas foram implementadas, todas usam GRASP/VND para encontrar uma solução inicial, usada como ponto de partida para uma busca ILS/VND. Os três métodos diferem no critério de aceitação para troca da solução corrente durante a busca ILS/VND. O primeiro método aceita apenas solução com menor distância total (Esse método será denominado método 1 de agora em diante), o segundo aceita de acordo com SA (método 2), e o terceiro aceita caso o custo penalizado da nova solução seja menor que o custo penalizado da solução corrente, de acordo com a metaheurística GLS (método 3).

Os algoritmos foram implementados em c++ e compilados com o compilador g++. Os experimentos computacionais foram realizados em uma máquina Intel Xeon Core 2 Quad, 2,00GHz, 8GB de RAM e sistema operacional Ubuntu.

As instâncias propostas por Salhi e Nagy [42] e por Dethloff [17] são amplamente adotadas na literatura e foram utilizadas nos testes computacionais. As instâncias de Salhi e Nagy são geradas a partir das instâncias de Christofides [13] que compreendem 14 problemas contendo de 50 a 199 consumidores. Duas instâncias, chamadas X e Y, são geradas a partir de cada instância de Chrisofides. Para cada consumidor i, com demanda $\hat{d}_i$, em uma instância de Christofides calcula-se um valor $r_i = \min\{x_i/y_i, y_i/x_i\}$, em que x_i e y_i são as coordenadas do consumidor e gera-se a demanda de entrega $d_i = r_i t_i$ e de coleta $p_i = (1 - r_i)t_i$, tem-se então uma instância do tipo X, a instância do tipo Y é gerada invertendo-se as demandas de coleta e entrega (a propriedade 5 mostra que X e Y são equivalentes). Algumas dessas instâncias possuem restrições de distância para as rotas e não foram usadas. As instâncias de Dethloff compreendem 40 problemas de 50 consumidores cada e são geradas estocasticamente. Nas instâncias do

tipo SCA as coordenadas dos consumidores são distribuídas uniformemente no intervalo $[0, 100]$, já nas instâncias do tipo CON, metade dos consumidores são distribuídos da mesma forma que em SCA e a outra metade é distribuída no intervalo $[100/3, 200/3]$, dando uma configuração mais urbana à instância, segundo o autor. A capacidade do veículo é dada por $Q = \sum_{i \in V, i \neq 0} d_i/\mu$, em que μ tem valor 3 ou 8 (esse valor aparece após as letras no nome da instância).

Em cada execução do algoritmo baseado em Kruskal um parâmetro de aleatorização $\alpha = \omega(10)/30$ é usado, em que $\omega(n)$ retorna um número entre 1 e n, como definido na seção 4.2.1. Portanto, valores $\{0,0333; 0,0666; 0,1; \ldots; 0,3\}$ são usados, com menor probabilidade para números grandes (a probabilidade decresce quadraticamente conforme o número cresce). Os valores menores propiciam a geração de soluções variadas e de boa qualidade, porém, a possibilidade de valores maiores faz com que soluções mais diversificadas também sejam exploradas.

Foram usados tamanhos de perturbação nos intervalos $[0,1 \times N_c; 0,3 \times N_c]$ para H, $[0,1 \times N_c/N_R; 0,5 \times N_c/N_R]$ para Relocate e $[0,05 \times N_c; 0,15 \times N_c]$ para Swap. Sendo N_c o número de consumidores e N_R é o número de rotas na solução. Valores são retirados desses intervalos como descrito na seção 4.2.1.

Para a escolha do parâmetro γ, usado na atualização da temperatura do SA, testes preliminares foram realizados com valores entre $0,9$ e $0,99$. Para a escolha do parâmetro λ, usado como peso para as penalidades no método GLS, testes preliminares foram realizados com valores entre $0,5$ e 5 . As heurísticas foram executadas 10 vezes para cada parâmetro e instância. Os valores $\gamma = 0,92$ e $\lambda = 1,5$ obtiveram os melhores resultados e foram usados nos experimentos subsequentes.

Cada método foi executado por 100 vezes para cada instância. Em cada execução, 100 iterações GRASP/VND e 500 iterações ILS/VND são realizadas.

Nas tabelas a seguir DT indica a distância total, NV o número de veículos e Gap indica a distância percentual para a melhor solução conhecida. O Gap foi calculado como $(\mathrm{DT} - \mathrm{DT}_{\mathrm{melhor}})100/\mathrm{DT}_{\mathrm{melhor}}$, em que $\mathrm{DT}_{\mathrm{melhor}}$ é o melhor resultado conhecido.

As tabelas 4.1 e 4.2 apresentam os melhores resultados no trabalho original e atualmente na literatura, considerando a distância total percorrida, para as instâncias de [42] e [17], respectivamente. A tabelas exibem a distância total percorrida (DT) e o número de veículos (NV) das soluções. Apenas o melhor resultado entre cada instância X e Y é exibido, como consequência da propriedade 5. Os melhores resultados para as instâncias de [42] são encontrados nos trabalhos:

- [43]: cmt5.
- [48]: cmt1, cmt2, cmt6, and cmt7.

- [50]: cmt3 and cmt5.

Os melhores resultados para as instâncias de [17] são encontrados nos trabalhos de:

- [12]: CON8-9.
- [32]: SCA3-1, SCA3-2, SCA3-3, SCA3-4, SCA3-5, SCA3-6, SCA3-7, SCA3-8, SCA3-9, SCA8-3, CON3-1, CON3-3, CON3-5, CON3-8, CON8-1, CON8-3, CON8-4, e CON8-6.
- [40]: SCA3-1, SCA3-2, SCA3-3, SCA3-4, SCA3-5, SCA3-6, SCA3-8, SCA3-9, SCA8-2, SCA8-4, SCA8-5, SCA8-8, SCA8-9, CON3-0, CON3-1, CON3-3, CON3-4 ,CON3-5, CON3-6, CON3-7, CON3-8, CON3-9, CON8-0, CON8-1 , CON8-3, e CON8-4.
- [43]: Obtem o melhor resultado em todas as instâncias, exceto CON8-9.
- [50]: SCA3-1, SCA3-2, SCA3-3, SCA3-4, SCA3-5, SCA3-6, SCA3-7, SCA3-8, SCA3-9, SCA8-0, SCA8-2, SCA8-3, SCA8-4, SCA8-5, SCA8-6, SCA8-8, SCA8-9, CON3-0, CON3-1, CON3-3, CON3-5, CON3-7, CON3-8, CON8-0, CON8-1, CON8-3, CON8-4, CON8-6, CON8-7, e CON8-8.

		Orig.		Melhor Lit.	
Problema	N_c	DT	NV	DT	NV
cmt1	50	603	5	458,96	3
cmt2	75	924	12	663,25	6
cmt3	100	923	10	721,27	5
cmt4	150	1178	15	852,35	7
cmt5	199	1509	19	1030,55	10
cmt11	120	1500	11	830,39	4
cmt12	100	820	10	644,7	5
Média		1065,28	11,7	743,06	5,7

Tabela 4.1. Melhores resultados no trabalho original e atualmente para as instâncias de Salhi e Nagy [42].

A tabela 4.3 exibe os melhores resultados encontrados por cada heurística proposta para as instâncias de [42], e a tabela 4.4 exibe os resultados e tempos computacionais médios.

Analisando a tabela 4.3, pode ser visto que todos os três métodos igualaram o melhor resultado da literatura na instância cmt3. O método 1 melhorou o melhor resultado da literatura na instância cmt5, esse resultado foi posteriormente melhorado

	Orig.	Melhor Lit.	
Problema	DT	DT	NV
SCA3-0	689	635,62	4
SCA3-1	765,6	697,84	4
SCA3-2	742,8	659,34	4
SCA3-3	737,2	680,04	4
SCA3-4	747,1	690,5	4
SCA3-5	784,4	659,9	4
SCA3-6	720,4	651,09	4
SCA3-7	707,9	659,17	4
SCA3-8	807,2	719,47	4
SCA3-9	764,1	681	4
SCA8-0	1132,9	961,5	9
SCA8-1	1150,9	1049,65	9
SCA8-2	1100,8	1039,64	9
SCA8-3	1115,6	983,34	9
SCA8-4	1235,4	1065,49	9
SCA8-5	1231,6	1027,08	9
SCA8-6	1062,5	971,82	9
SCA8-7	1217,4	1051,28	9
SCA8-8	1231,6	1071,18	9
SCA8-9	1185,6	1060,5	9
CON3-0	672,4	616,52	4
CON3-1	570,6	554,47	4
CON3-2	534,8	518	4
CON3-3	656,9	591,19	4
CON3-4	640,2	588,79	4
CON3-5	604,7	563,7	4
CON3-6	521,3	499,05	4
CON3-7	602,8	576,48	4
CON3-8	556,2	523,05	4
CON3-9	612,8	578,24	4
CON8-0	967,3	857,17	9
CON8-1	828,7	740,85	9
CON8-2	770,2	712,89	9
CON8-3	906,7	811,07	10
CON8-4	876,8	772,25	9
CON8-5	866,9	754,88	9
CON8-6	749,1	678,92	9
CON8-7	929,8	811,96	9
CON8-8	833,1	767,53	9
CON8-9	877,3	806,72	-
Média	842,71	758,47	6,4

Tabela 4.2. Melhores resultados no trabalho original e atuamente para as instâncias de Dethloff [17]. O número de veículos não é apresentado em [17]. O melhor resultado conhecido para a instância CON8-9 é apresentado em [12], no qual não é dado o número de veículos da solução.

	Met. 1			Met. 2			Met. 3		
Problema	DT	NV	Gap	DT	NV	Gap	DT	NV	Gap
cmt1	466,77	3	0,00	466,77	3	0,00	466,77	3	0,00
cmt2	684,21	6	3,16	684,21	6	3,16	684,21	6	3,16
cmt3	721,27	5	0,00	721,27	5	0,00	721,27	5	0,00
cmt4	852,46	7	0,01	852,46	7	0,01	852,83	7	0,06
cmt5	1030,27	10	-0,03	1032,59	10	0,20	1029,80	10	-0,07
cmt11	836,22	4	0,70	840,77	4	1,25	837,07	4	0,81
cmt12	662,22	5	2,72	662,22	5	2,72	662,22	5	2,72
Média	750,49	5,7	0,93	751,47	5,7	1,04	750,59	5,7	0,95

Tabela 4.3. Melhores resultados das heurísticas para as instâncias de Salhi e Nagy [42].

	Met. 1				Met. 2				Met. 3			
Problema	DT	NV	T	Gap	DT	NV	T	Gap	DT	NV	T	Gap
cmt1	467,37	3	2	1,83	467,64	3	1,8	1,89	467,97	3	1,9	1,96
cmt2	689,42	6,1	4,3	3,94	690,62	6,1	3,7	4,12	689,23	6,1	3,8	3,91
cmt3	723,47	5	10,7	0,3	723,98	5	10	0,37	723,73	5	10,2	0,34
cmt4	859,04	7	33,9	0,78	861,09	7	32,9	1,02	859,77	7	32,1	0,87
cmt5	1042,6	10	69,9	1,16	1044,96	10	68,5	1,39	1044,94	10	67,8	1,39
cmt6	847,54	4	28,2	2,06	853,31	4,2	26,4	2,76	846,82	4	31,2	1,97
cmt7	666,87	5,7	9,6	3,43	668,35	5,7	9,5	3,66	667,49	5,8	9,9	3,53
Média	756,61	5,8	22,6	1,92	758,56	5,8	21,8	2,17	757,13	5,8	22,4	1,99

Tabela 4.4. Resultados médios das heurísticas para as instâncias de Salhi e Nagy [42].

pelo método 3. O pior gap entre os três métodos, em relação ao melhor da literatura foi de 3,16%, na instância cmt2. Examinando os resultados médios, na tabela 4.4, vê-se que os três métodos tem desempenho bom e similar na média, com o pior gap em relação ao melhor da literatura em torno de 4%, na instância 2, e gap médio em torno 2%.

Na tabela 4.5 os melhores resultados encontrados por cada método nas instâncias de [17] são exibidos. A tabela 4.6 exibe os resultados e tempos computacionais médios.

Analizando os melhores resultados das heurísticas na tabela 4.5 para as instâncias de [17], vê-se que os três métodos obtiveram bom desempenho. O Método 2 não conseguiu encontrar o melhor resultado da literatura em apenas 1 instância (CON8-8), o método 3 não igualou o melhor resultado da literatura em apenas duas instâncias (SCA8-2 e CON8-9) e o método 1 não foi capaz de encontrar o melhor resultado da literatura em três instâncias (SCA8-2, SCA8-7, e CON8-9). Na média, observando-se a tabela 4.6, os três métodos obtiveram desempenho similar, com um pequeno gap médio

	Met. 1			Met. 2			Met. 3		
Problema	DT	NV	Gap	DT	NV	Gap	DT	NV	Gap
SCA3-0	635,62	4	0,00	635,62	4	0,00	635,62	4	0,00
SCA3-1	697,83	4	0,00	697,83	4	0,00	697,83	4	0,00
SCA3-2	659,33	4	0,00	659,33	4	0,00	659,33	4	0,00
SCA3-3	680,04	4	0,00	680,04	4	0,00	680,04	4	0,00
SCA3-4	690,50	4	0,00	690,50	4	0,00	690,50	4	0,00
SCA3-5	659,90	4	0,00	659,90	4	0,00	659,90	4	0,00
SCA3-6	651,08	4	0,00	651,08	4	0,00	651,08	4	0,00
SCA3-7	659,16	4	0,00	659,16	4	0,00	659,16	4	0,00
SCA3-8	719,47	4	0,00	719,47	4	0,00	719,47	4	0,00
SCA3-9	680,99	4	0,00	680,99	4	0,00	680,99	4	0,00
SCA8-0	961,49	9	0,00	961,49	9	0,00	961,49	9	0,00
SCA8-1	1049,65	9	0,00	1049,65	9	0,00	1049,65	9	0,00
SCA8-2	1039,71	9	0,01	1039,64	9	0,00	1044,48	9	0,46
SCA8-3	983,33	9	0,00	983,33	9	0,00	983,33	9	0,00
SCA8-4	1065,49	9	0,00	1065,49	9	0,00	1065,49	9	0,00
SCA8-5	1027,08	9	0,00	1027,08	9	0,00	1027,08	9	0,00
SCA8-6	971,82	9	0,00	971,82	9	0,00	971,82	9	0,00
SCA8-7	1053,84	9	0,24	1051,28	9	0,00	1051,28	9	0,00
SCA8-8	1071,18	9	0,00	1071,18	9	0,00	1071,18	9	0,00
SCA8-9	1060,50	9	0,00	1060,50	9	0,00	1060,50	9	0,00
CON3-0	616,52	4	0,00	616,52	4	0,00	616,52	4	0,00
CON3-1	554,47	4	0,00	554,47	4	0,00	554,47	4	0,00
CON3-2	518,00	4	0,00	518,00	4	0,00	518,00	4	0,00
CON3-3	591,18	4	0,00	591,18	4	0,00	591,18	4	0,00
CON3-4	588,79	4	0,00	588,79	4	0,00	588,79	4	0,00
CON3-5	563,69	4	0,00	563,69	4	0,00	563,69	4	0,00
CON3-6	499,05	4	0,00	499,05	4	0,00	499,05	4	0,00
CON3-7	576,48	4	0,00	576,48	4	0,00	576,48	4	0,00
CON3-8	523,05	4	0,00	523,05	4	0,00	523,05	4	0,00
CON3-9	578,24	4	0,00	578,24	4	0,00	578,24	4	0,00
CON8-0	857,17	9	0,00	857,17	9	0,00	857,17	9	0,00
CON8-1	740,85	9	0,00	740,85	9	0,00	740,85	9	0,00
CON8-2	712,88	9	0,00	712,88	9	0,00	712,88	9	0,00
CON8-3	811,06	10	0,00	811,06	10	0,00	811,06	10	0,00
CON8-4	772,25	9	0,00	772,25	9	0,00	772,25	9	0,00
CON8-5	754,88	9	0,00	754,88	9	0,00	754,88	9	0,00
CON8-6	678,92	9	0,00	678,92	9	0,00	678,92	9	0,00
CON8-7	811,95	9	0,00	811,95	9	0,00	811,95	9	0,00
CON8-8	767,52	9	0,00	767,52	9	0,00	767,52	9	0,00
CON8-9	809,00	9	0,28	809,00	9	0,28	809,00	9	0,28
Média	758,60	6,5	0,01	758,53	6,5	0,00	758,65	6,5	0,01

Tabela 4.5. Melhores resultados das heurísticas nas instâncias de Dethloff [17].

	Met. 1				Met. 2				Met. 3			
Problema	DT	NV	T	Gap	DT	NV	T	Gap	DT	NV	T	Gap
SCA3-0	638,06	4,0	1,6	0,38	637,90	4,0	1,3	0,35	638,35	4,0	1,6	0,42
SCA3-1	697,83	4,0	1,5	0,00	697,83	4,0	1,4	0,00	697,83	4,0	1,6	0,00
SCA3-2	659,33	4,0	1,4	0,00	659,33	4,0	1,3	0,00	659,33	4,0	1,5	0,00
SCA3-3	680,04	4,0	1,4	0,00	680,04	4,0	1,3	0,00	680,04	4,0	1,5	0,00
SCA3-4	690,50	4,0	1,4	0,00	690,50	4,0	1,3	0,00	690,50	4,0	1,4	0,00
SCA3-5	659,90	4,0	1,6	0,00	659,90	4,0	1,5	0,00	659,90	4,0	1,6	0,00
SCA3-6	651,12	4,0	1,5	0,00	651,08	4,0	1,4	0,00	651,08	4,0	1,5	0,00
SCA3-7	665,43	4,0	1,5	0,94	665,39	4,0	1,4	0,94	664,82	4,0	1,5	0,85
SCA3-8	719,47	4,0	1,5	0,00	719,47	4,0	1,4	0,00	719,47	4,0	1,5	0,00
SCA3-9	680,99	4,0	1,4	0,00	680,99	4,0	1,2	0,00	680,99	4,0	1,4	0,00
SCA8-0	964,41	9,0	0,8	0,30	966,32	9,0	0,7	0,50	964,45	9,0	0,8	0,30
SCA8-1	1056,56	9,0	0,8	0,65	1059,67	9,0	0,8	0,95	1056,84	9,0	0,9	0,68
SCA8-2	1049,27	9,8	0,7	0,92	1049,50	9,8	0,7	0,94	1049,80	9,9	0,8	0,97
SCA8-3	991,21	9,0	0,7	0,80	1002,90	9,0	0,7	1,98	986,78	9,0	0,8	0,34
SCA8-4	1067,18	9,0	0,8	0,15	1067,01	9,0	0,7	0,14	1067,18	9,0	0,8	0,15
SCA8-5	1035,18	9,0	0,8	0,78	1032,92	9,0	0,7	0,56	1034,26	9,0	0,8	0,69
SCA8-6	972,43	9,0	0,8	0,06	972,88	9,0	0,8	0,10	972,43	9,0	0,9	0,06
SCA8-7	1063,13	9,9	0,8	1,12	1062,34	9,9	0,8	1,05	1062,76	9,9	0,8	1,09
SCA8-8	1071,18	9,0	0,9	0,00	1071,18	9,0	0,8	0,00	1071,18	9,0	0,8	0,00
SCA8-9	1062,66	9,0	0,7	0,20	1062,43	9,0	0,7	0,18	1063,32	9,0	0,7	0,26
CON3-0	616,58	4,0	1,6	0,00	616,57	4,0	1,5	0,00	616,54	4,0	1,5	0,00
CON3-1	554,71	4,0	1,9	0,04	554,71	4,0	1,7	0,04	554,61	4,0	1,9	0,02
CON3-2	520,84	4,0	1,7	0,54	520,53	4,0	1,7	0,48	520,95	4,0	1,7	0,56
CON3-3	591,18	4,0	1,6	0,00	591,18	4,0	1,5	0,00	591,18	4,0	1,7	0,00
CON3-4	590,02	4,0	1,7	0,20	589,92	4,0	1,5	0,19	589,96	4,0	1,6	0,19
CON3-5	563,69	4,0	1,7	0,00	563,69	4,0	1,5	0,00	563,69	4,0	1,7	0,00
CON3-6	500,15	4,0	1,8	0,22	499,95	4,0	1,6	0,18	500,21	4,0	1,8	0,23
CON3-7	576,58	4,0	2,0	0,01	576,60	4,0	1,8	0,02	576,52	4,0	1,9	0,00
CON3-8	523,05	4,0	1,7	0,00	523,05	4,0	1,5	0,00	523,05	4,0	1,7	0,00
CON3-9	579,06	4,0	1,9	0,14	579,65	4,0	1,5	0,24	578,66	4,0	1,8	0,07
CON8-0	860,05	9,0	0,9	0,33	860,23	9,0	0,8	0,35	861,09	9,0	0,9	0,45
CON8-1	740,93	9,0	1,0	0,01	742,76	9,0	0,9	0,25	741,43	9,0	0,9	0,07
CON8-2	713,76	9,0	1,0	0,12	713,81	9,0	0,8	0,12	713,34	9,0	0,9	0,06
CON8-3	811,77	10,0	1,0	0,08	814,22	10,0	0,8	0,38	811,25	10,0	0,9	0,02
CON8-4	772,82	9,0	0,9	0,07	772,59	9,0	0,8	0,04	772,46	9,0	0,8	0,02
CON8-5	756,69	9,0	0,9	0,23	757,43	9,0	0,8	0,33	757,46	9,0	0,9	0,34
CON8-6	683,14	9,0	1,0	0,62	686,34	9,0	0,8	1,09	683,37	9,0	0,9	0,65
CON8-7	812,12	9,0	0,9	0,01	811,99	9,0	0,7	0,00	811,98	9,0	0,8	0,00
CON8-8	771,83	9,0	1,0	0,56	771,44	9,0	0,8	0,50	773,59	9,0	0,9	0,78
CON8-9	810,54	9,0	1,0	0,47	810,56	9,0	0,9	0,47	810,70	9,0	0,9	0,49
Média	760,63	6,5	1,2	0,24	761,17	6,5	1,1	0,30	760,58	6,5	1,2	0,24

Tabela 4.6. Resultados médios das heurísticas para as instâncias de Dethloff [17].

de cerca de 0,26%.

As tabelas entre 4.7 e 4.14 exibem resultados computacionais para as instâncias usadas no trabalho de Salani [41], para as quais são conhecidas soluções ótimas. Esse conjunto de instâncias consiste em instâncias de quatro tipos: 1S, 1C, 2S e 2C de [41]. O conjunto de instâncias 1S é composto de 18 instâncias de 20 e 40 consumidores. Os consumidores têm sua coordenadas x e y distribuídas uniformemente nos intervalos $[0, 2400]$ e $[0, 3600]$ respectivamente. O depósito é localizado na posição (1200,1600). O custo dos arcos é definido pela distância euclidiana arredondada para o inteiro acima. As demandas são geradas aleatoriamente a partir de uma distribuição normal com média 50 e desvio padrão 20. A fração de consumidores de entrega é de 1/3, 2/3 ou 4/5. Nessa classe de instãncias os consumidores só recebem ou só entregam. O veículo assume capacidade 100, 150 ou 200. A classe 1C é obtida a partir da classe 1S da seguinte maneira: todas as demandas são consideradas demanda de entrega e para cada consumidor i computa-se uma demanda de coleta $p_i = (0, 5 + r)d_i$, em que r é obtido a partir de uma distribuição uniforme no intervalo $[0, 1]$. A classe 2S consiste em 18 instâncias obtidas através de instâncias do mundo real para o CVRP com 20 e 40 consumidores. Cada instãncia do VRPSPD é gerada a partir de uma instância do CVRP com uma quantidade de consumidores de entrega igual a 1/2, 2/3 ou 4/5 e capacidade do veículo igual a 150, 200 ou 300. A classe 2S é obtida a partir da classe 2C pelo mesmo método usado na obtenção da classe 1C. O primeiro ítem no nome de cada instância indica sua classe, o segundo indica a quantidade de consumidores, o terceiro indica a porcentagem de consumidores de entrega e o quarto indica a capacidade do veículo.

Foram usadas apenas as instâncias contendo 40 consumidores. Os resultados computacionais são comparados com a solução ótima. Não temos conhecimento de outro trabalho envolvendo heurísticas que tenha feito testes computacionais nessas instâncias. Observando as tabelas com os melhores resultados para as instâncias, vemos que as heurísticas que desenvolvemos se comportam muito bem, as heurísticas 2 e 3 não encontram a solução ótima em apenas uma instância (2S_40_66_2), enquanto a heurística 1 não encontrou o resultado ótimo em apenas três instâncias (1C_40_66_2, 2S_40_50_3 e 2S_40_66_2). Na média, os três algoritmos apresentam resultados bastante próximos da solução ótima, a heurística 1 apresenta um gap médio de 0,3%, a heurística 2 0,2% e a heurística 3 0,3%. O pior gap médio em uma determinada instância tem valor de apenas 2,4%, apresentado pelo algoritmo 3 na instância 1S_40_50_3.

Problema	Opt	Met. 1			Met. 2			Met. 3		
		DT	NV	Gap	DT	NV	Gap	DT	NV	Gap
1S_40_50_1	357430	357430	10	0	357430	10	0	357430	10	0
1S_40_50_2	269590	269590	7	0	269590	7	0	269590	7	0
1S_40_50_3	229044	229044	5	0	229044	5	0	229044	5	0
1S_40_66_1	377279	377279	13	0	377279	13	0	377279	13	0
1S_40_66_2	291008	291008	9	0	291008	9	0	291008	9	0
1S_40_66_3	241347	241347	7	0	241347	7	0	241347	7	0
1S_40_80_1	425911	425911	16	0	425911	16	0	425911	16	0
1S_40_80_2	324920	324920	11	0	324920	11	0	324920	11	0
1S_40_80_3	270313	270313	8	0	270313	8	0	270313	8	0
Média	309649,1	309649,1	9,5	0,0	309649,1	9,5	0,0	309649,1	9,5	0,0

Tabela 4.7. Melhores resultados encontrados pelas heurísticas para as instâncias do tipo 1S.

Problema	Met. 1				Met. 2				Met. 3			
	DT	NV	T	Gap	DT	NV	T	Gap	DT	NV	T	Gap
1S_40_50_1	359282,0	10,0	0,5	0,5	358207,0	10,0	0,5	0,2	359286,0	10,0	0,5	0,5
1S_40_50_2	269590,0	7,0	0,5	0	269590,0	7,0	0,4	0	269590,0	7,0	0,5	0
1S_40_50_3	234492,0	5,9	0,5	2,3	232056,0	5,5	0,6	1,3	234548,0	5,9	0,5	2,4
1S_40_66_1	377315,0	13,0	0,5	0	377303,0	13,0	0,5	0	377323,0	13,0	0,5	0
1S_40_66_2	293108,0	9,0	0,5	0,7	291470,0	9,0	0,5	0,1	293152,0	9,0	0,5	0,7
1S_40_66_3	241615,0	7,0	0,5	0,1	241424,0	7,0	0,5	0	241615,0	7,0	0,5	0,1
1S_40_80_1	425911,0	16,0	0,4	0	425911,0	16,0	0,4	0	425911,0	16,0	0,4	0
1S_40_80_2	325040,0	11,0	0,5	0	325099,0	11,0	0,4	0	325040,0	11,0	0,5	0
1S_40_80_3	270822,0	8,0	0,5	0,1	270798,0	8,0	0,6	0,1	270807,0	8,0	0,6	0,1
Média	310797,2	9,6	0,4	0,4	310206,4	9,6	0,4	0,1	310808,0	9,6	0,5	0,4

Tabela 4.8. Resultados médios encontrados pelas heurísticas para as instâncias do tipo 1S.

4.4 Conclusão

Nesta seção, foram propostas três heurísticas para o VRPSPD, baseadas nas metaheurísticas GRASP/ILS e VND. Os métodos usam GRASP/VND para gerar um ponto de partida para uma busca ILS/VND e se diferem no critério de aceitação do método ILS. O primeiro tem um critério simples, aceitar se a nova solução for melhor. O segundo aceita de acordo com SA, o terceiro aceita de acordo com uma estratégia baseada no método GLS. Na média, os três métodos têm desempenhos similares, com bons gaps médios em relação aos melhores resultados da literatura. Acredita-se que esse fato está associado a qualidade da busca VND. Porém, a heurística com aceitação baseada em GLS levou o terceiro método a obter o melhor resultado da literatura em uma das

Problema	Opt	Met. 1			Met. 2			Met. 3		
		DT	NV	Gap	DT	NV	Gap	DT	NV	Gap
1C_40_50_1	601817	601817	23	0	601817	23	0	601817	23	0
1C_40_50_2	402309	402309	15	0	402309	15	0	402309	15	0
1C_40_50_3	334830	334830	11	0	334830	11	0	334830	11	0
1C_40_66_1	630257	630257	24	0	630257	24	0	630257	24	0
1C_40_66_2	426517	429125	16	0,6	426517	15	0	426517	15	0
1C_40_66_3	331459	331459	11	0	331459	11	0	331459	11	0
1C_40_80_1	647539	647539	25	0	647539	25	0	647539	25	0
1C_40_80_2	424368	424368	15	0	424368	15	0	424368	15	0
1C_40_80_3	332957	332957	11	0	332957	11	0	332957	11	0
Média	459117,0	459406,7	16,7	0,0	459117,0	16,6	0,0	459117,0	16,6	0,0

Tabela 4.9. Melhores resultados encontrados pelas heurísticas para as instâncias do tipo 1C.

Problema	Met. 1				Met. 2				Met. 3			
	DT	NV	T	Gap	DT	NV	T	Gap	DT	NV	T	Gap
1C_40_50_1	601817,0	23,0	0,3	0	601817,0	23,0	0,3	0	601817,0	23,0	0,3	0
1C_40_50_2	402314,0	15,0	0,3	0	402309,0	15,0	0,4	0	402313,0	15,0	0,4	0
1C_40_50_3	336262,0	11,0	0,4	0,4	336995,0	11,0	0,4	0,6	336405,0	11,0	0,4	0,4
1C_40_66_1	630257,0	24,0	0,3	0	630257,0	24,0	0,3	0	630257,0	24,0	0,3	0
1C_40_66_2	429125,0	16,0	0,4	0,6	428593,0	15,7	0,4	0,4	429099,0	15,9	0,4	0,6
1C_40_66_3	332592,0	11,0	0,4	0,3	332618,0	11,0	0,4	0,3	332376,0	11,0	0,4	0,2
1C_40_80_1	647539,0	25,0	0,3	0	647539,0	25,0	0,3	0	647539,0	25,0	0,3	0
1C_40_80_2	424560,0	15,0	0,4	0	424514,0	15,0	0,4	0	424600,0	15,0	0,4	0
1C_40_80_3	333315,0	11,0	0,4	0,1	333801,0	11,0	0,4	0,2	333360,0	11,0	0,4	0,1
Média	459753,4	16,7	0,3	0,1	459827,0	16,7	0,3	0,1	459751,7	16,7	0,3	0,1

Tabela 4.10. Resultados médios encontrados pelas heurísticas para as instâncias do tipo 1C.

instâncias de Salhi e Nagy.

Problema	Opt	Met. 1			Met. 2			Met. 3		
		DT	NV	Gap	DT	NV	Gap	DT	NV	Gap
2S_40_50_1	16917	16917	12	0	16917	12	0	16917	12	0
2S_40_50_2	14603	14603	8	0	14603	8	0	14603	8	0
2S_40_50_3	11343	11407	5	0,5	11343	5	0	11343	5	0
2S_40_66_1	17932	17932	13	0	17932	13	0	17932	13	0
2S_40_66_2	15307	15349	10	0,2	15349	10	0,2	15349	10	0,2
2S_40_66_3	11725	11725	6	0	11725	6	0	11725	6	0
2S_40_80_1	20660	20660	18	0	20660	18	0	20660	18	0
2S_40_80_2	17201	17201	13	0	17201	13	0	17201	13	0
2S_40_80_3	13208	13208	9	0	13208	9	0	13208	9	0
Média	15432,8	15444,6	10,4	0,0	15437,5	10,4	0,0	15437,5	10,4	0,0

Tabela 4.11. Melhores resultados encontrados pelas heurísticas para as instâncias do tipo 2S.

Problema	Met. 1				Met. 2				Met. 3			
	DT	NV	T	Gap	DT	NV	T	Gap	DT	NV	T	Gap
2S_40_50_1	16937,2	11,6	0,5	0,1	16958,6	11,2	0,5	0,2	16943,4	11,6	0,5	0,1
2S_40_50_2	14698,8	8,0	0,5	0,6	14704,6	8,0	0,5	0,6	14690,9	8,0	0,5	0,6
2S_40_50_3	11485,1	5,9	0,5	1,2	11497,6	5,9	0,5	1,3	11479,9	5,9	0,5	1,2
2S_40_66_1	17934,6	13,0	0,5	0	17933,6	13,0	0,5	0	17934,2	13,0	0,5	0
2S_40_66_2	15508,7	10,0	0,5	1,3	15487,0	9,9	0,4	1,1	15497,5	10,0	0,5	1,2
2S_40_66_3	11974,1	6,8	0,5	2,1	11943,0	6,7	0,5	1,8	11985,9	6,8	0,5	2,2
2S_40_80_1	20662,6	17,9	0,4	0	20663,4	17,8	0,4	0	20664,1	17,9	0,4	0
2S_40_80_2	17219,5	13,0	0,4	0,1	17219,4	13,0	0,4	0,1	17224,2	13,0	0,5	0,1
2S_40_80_3	13208,1	9,0	0,4	0	13208,0	9,0	0,5	0	13208,0	9,0	0,5	0
Média	15514,3	10,5	0,4	0,6	15512,8	10,5	0,4	0,5	15514,2	10,5	0,4	0,6

Tabela 4.12. Resultados médios encontrados pelas heurísticas para as instâncias do tipo 2S.

Problema	Opt	Met. 1			Met. 2			Met. 3		
		DT	NV	Gap	DT	NV	Gap	DT	NV	Gap
2C_40_50_1	26988	26988	23	0	26988	23	0	26988	23	0
2C_40_50_2	21710	21710	17	0	21710	17	0	21710	17	0
2C_40_50_3	15523	15523	10	0	15523	10	0	15523	10	0
2C_40_66_1	25981	25981	22	0	25981	22	0	25981	22	0
2C_40_66_2	21317	21317	16	0	21317	16	0	21317	16	0
2C_40_66_3	15293	15293	10	0	15293	10	0	15293	10	0
2C_40_80_1	26122	26122	22	0	26122	22	0	26122	22	0
2C_40_80_2	20652	20652	16	0	20652	16	0	20652	16	0
2C_40_80_3	15365	15365	10	0	15365	10	0	15365	10	0
Média	20994,5	20994,5	16,2	0,0	20994,5	16,2	0,0	20994,5	16,2	0,0

Tabela 4.13. Melhores resultados encontrados pelas heurísticas para as instâncias do tipo 2C.

Problema	Met. 1				Met. 2				Met. 3			
	DT	NV	T	Gap	DT	NV	T	Gap	DT	NV	T	Gap
2C_40_50_1	26988,0	23,0	0,3	0	26988,0	23,0	0,3	0	26988,0	23,0	0,3	0
2C_40_50_2	21744,9	17,0	0,4	0,1	21740,2	17,0	0,4	0,1	21745,1	17,0	0,4	0,1
2C_40_50_3	15554,8	10,7	0,4	0,2	15549,8	10,6	0,4	0,1	15552,7	10,7	0,4	0,1
2C_40_66_1	25981,0	22,0	0,3	0	25981,0	22,0	0,3	0	25981,0	22,0	0,3	0
2C_40_66_2	21422,3	15,9	0,4	0,4	21388,8	15,9	0,4	0,3	21406,4	15,9	0,4	0,4
2C_40_66_3	15300,6	10,0	0,5	0	15299,9	10,0	0,4	0	15300,6	10,0	0,5	0
2C_40_80_1	26123,8	22,0	0,3	0	26122,0	22,0	0,3	0	26124,4	22,0	0,3	0
2C_40_80_2	20697,7	16,8	0,4	0,2	20700,3	16,7	0,4	0,2	20695,5	16,8	0,4	0,2
2C_40_80_3	15400,1	10,1	0,4	0,2	15411,7	10,0	0,4	0,3	15398,6	10,1	0,4	0,2
Média	21023,6	16,3	0,3	0,1	21020,1	16,3	0,3	0,1	21021,3	16,3	0,3	0,1

Tabela 4.14. Resultados médios encontrados pelas heurísticas para as instâncias do tipo 1C.

Capítulo 5

Branch-and-Price aplicado ao VRPSPD

O algoritmo *Branch-and-Price* (BP) é um algoritmo *Branch-and-Bound* onde a Relaxação Linear, calculada por Geração de Colunas, de uma reformulação do problema é utilizada como limite inferior em cada nó da árvore, ao invés de outra relaxação.

Nesta seção, introduzimos o algoritmo *Branch-and-price* desenvolvido. Na seção 5.1 é dada uma breve introdução ao método de Geração de Colunas e Decomposição Dantzig-Wolfe. Posteriormente, na seção 5.2, é apresentado o modelo matemático e sua reformulação para o VRPSPD. A seção 5.3 trata da resolução do Subproblema de Precificação, nesta seção são propostas algumas estratégias para a obtenção de rotas de custo reduzido negativo. Os elementos do algoritmo *Branch-and-price* desenvolvido são apresentados na seção 5.4, nesta seção é proposta uma estratégia em que são utilizadas algumas iterações de resolução do Problema Mestre Restrito para a escolha da variável de *branching*. Experimentos computacionais são relatados na seção 5.5 e o capítulo é concluído na seção 5.6.

5.1 Geração de Colunas/Dantzig-Wolfe

Dado um problema de otimização

$$\min cx \tag{5.1}$$

Sujeito a:

$$Ax \geq b \tag{5.2}$$

$$x \in X \tag{5.3}$$

em que $X = P \cap \mathbb{Z}^n$ é um conjunto de pontos inteiros pertencentes a um poliedro P, que por simplicidade assumimos ser fechado, assim, X é um conjunto finito de pontos $\{x_i\}_{i=1}^{I}$. Um ponto $x \in X$ tem a forma

$$x = \sum_{i=1}^{I} x_i \lambda_i \tag{5.4}$$

Sujeito a:

$$\sum_{i=1}^{I} \lambda_i = 1 \tag{5.5}$$

$$\lambda_i \in \mathbb{B}, 1 \leq i \leq I \tag{5.6}$$

Substituindo x em (5.1) e (5.2), tem-se uma formulação equivalente (reformulação Dantzig-Wolfe), esse problema é chamado de Problema Mestre:

$$\min \sum_{i=1}^{I} (cx_i)\lambda_i \tag{5.7}$$

Sujeito a:

$$\sum_{i=1}^{I} (Ax_i)\lambda_i \geq b \tag{5.8}$$

$$\sum_{i=1}^{I} \lambda_i = 1 \tag{5.9}$$

$$\lambda_i \in \mathbb{B}, 1 \leq i \leq I \tag{5.10}$$

Geralmente, as variáveis em X não são conhecidas ou são muitas para que sejam explicitadas. Na prática, trabalha-se com um subconjunto pequeno de colunas. Dessa maneira, tem-se um Problema Mestre Restrito. O Problema Mestre é, então, resolvido através de Geração de Colunas, isto é, novas colunas são geradas caso seja necessário. Assim, no passo k do processo apenas um subconjunto restrito $X^k = \{x_i\}_{i=1}^{I^k}$ de colunas é conhecido, tem-se então o Problema Mestre Restrito

$$\min \sum_{i=1}^{I^k} (cx_i)\lambda_i \tag{5.11}$$

Sujeito a:

$$\sum_{i=1}^{I^k} (Ax_i)\lambda_i \geq b \tag{5.12}$$

$$\sum_{i=1}^{I^k} \lambda_i = 1 \tag{5.13}$$

$$\lambda_i \in \mathbb{B}, 1 \leq i \leq I^k \tag{5.14}$$

que tem como saída uma solução primal $x^k = \sum_{i=1}^{I^k} x_i\lambda_i$ e dual $(\theta^k, \mu^k) \in \mathbb{R}_+^m \times \mathbb{R}$. O custo reduzido de uma coluna $x \in X$ é dado por $(c - \theta^k A)x - \mu^k$, se uma coluna em X tiver custo reduzido negativo, a inserção da variável correspondente na base traz um ganho na função objetivo (na ausência de degeneração), essa coluna é então inserida no conjunto de colunas do Problema Mestre Restrito, gerando X^{k+1}. Se não houver coluna com custo reduzido negativo, a solução do Problema Mestre corrente é otima. Portanto, a cada passo é resolvido o problema

$$\hat{x} = \arg\min_{x \in X} (c - \theta_k A)x. \tag{5.15}$$

Esse problema é denominado Subproblema de Precificação. Se $\hat{x}$ tiver custo reduzido negativo, é inserido no Problema Mestre Restrito e o processo é repetido, senão, a solução atual é ótima. Resolver o problema Mestre é equivalente a resolver a formulação inteira original. Entretanto, a Relaxação Linear do Problema Mestre geralmente leva a limites mais justos (igual no pior dos casos) do que a Relaxação Linear da formulação original. Para mais detalhes sobre geração de colunas e decomposição DW veja [2, 8, 45].

5.2 Formulação Matemática e Reformulação DW para o VRPSPD

O modelo matemático, apresentado a seguir, referente ao VRPSPD, foi desenvolvido por Dell'amico et al. [15].
Os dados do problema são:

- Um grafo $G = (V, A)$.

- Um depósito, representado pelo vértice $0 \in V$.
- Clientes, representados pelos vértices em $V \setminus \{0\}$. Cada consumidor i, com demandas p_i de coleta e d_i de entrega.
- Conexões entre os consumidores, representadas pelos arcos em A, com custos c_{ij}, satisfazendo a desigualdade triangular, para todo (i, j) em A.
- Número máximo de veículos K. Cada veículo com capacidade Q.

As variáveis de decisão são:

- x_{ij}, variável binária, assumindo valor 0 se o arco $(i, j) \in A$ não estiver na solução e valor 1 caso contrário.
- P_{ij}, variável real não negativa, indicando a quantidade de carga coletada passando pelo arco (i, j).
- D_{ij}, variável real não negativa, indicando a quantidade de carga a ser entregue passando pelo arco (i, j).

Formulação Matemática:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{5.16}$$

Sujeito a:

$$\sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\} \tag{5.17}$$

$$\sum_{j\in V\setminus\{0\}} x_{0j} \leq K \tag{5.18}$$

$$\sum_{j\in V} x_{ij} = \sum_{j\in V} x_{ji} \qquad \forall i \in V \tag{5.19}$$

$$\sum_{j\in V} P_{ij} - \sum_{j\in V} P_{ji} = p_i \qquad \forall i \in V \setminus \{0\} \tag{5.20}$$

$$\sum_{j\in V} D_{ji} - \sum_{j\in V} D_{ij} = d_i \qquad \forall i \in V \setminus \{0\} \tag{5.21}$$

$$P_{ij} + D_{ij} \leq Qx_{ij} \qquad \forall (i, j) \in A \tag{5.22}$$

$$P_{ij}, D_{ij} \geq 0 \qquad \forall (i, j) \in A \tag{5.23}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A \tag{5.24}$$

A restrição (5.17) faz com que cada consumidor seja visitado uma única vez. A restrição (5.18) garante que não sejam usados mais de K veículos. Restrições (5.19),

(5.20) e (5.21) são restrições de conservação de fluxo no número de veículos, quantidade de coleta e de entrega, respectivamente. Note que essas restrições são responsáveis pela eliminação de subrotas. Restrição (5.22) garante que a capacidade do veículo não seja violada.

Seja R o conjunto de todas as rotas viáveis, λ_r uma variável binária indicando se a rota $r \in R$ faz parte da solução e y_i^r um parâmetro que indica se o consumidor i é atendido na rota r. Pela decomposição Dantzig-Wolfe, a seguinte formulação é obtida.
Problema Mestre:

$$\min \sum_{r \in R} c_r \lambda_r \tag{5.25}$$

Sujeito a:

$$\sum_{r \in R} y_i^r \lambda_r \geq 1 \qquad \forall i \in V \setminus \{0\} \tag{5.26}$$

$$\sum_{r \in R} \lambda_r \leq K \tag{5.27}$$

$$\lambda_r \in \{0, 1\} \qquad \forall r \in R \tag{5.28}$$

A restrição (5.26) exige que cada consumidor seja atendido pelo menos uma vez. A restrição (5.27) proíbe o uso de mais de K veículos. É usada uma restrição de cobertura de conjuntos (5.26) ao invés de uma restrição de particionamento de conjutos, pois, na solução ótima do problema um consumidor não será visitado mais de uma vez e o modelo com cobertura tem um espaço dual menor quando a restrição de integralidade (5.28) é relaxada. Além disso, dado um conjunto restrito de colunas, usando cobertura é mais fácil de se encontrar soluções viáveis [15].

Apenas um conjunto limitado de colunas (rotas) $R' \subseteq R$ é conhecido, dessa maneira, tem-se um Problema Mestre Restrito, e sua relaxação linear é resolvida por geração de colunas. A cada otimização do Problema Mestre Restrito com o conjunto de colunas atual, busca-se uma nova coluna que tenha custo reduzido negativo para que sua variável λ associada seja introduzida na base. Essa rota é gerada através da resolução do subproblema abaixo. Na formulação, x_{ij} é uma variável binária que indica o uso do arco (i, j), a variável binária y_i assume valor 1 caso o consumidor i seja visitado pela rota e 0 caso contrário e as variáveis P_{ij} e D_{ij} correspondem respectivamente à carga de coleta e entrega passando pelo arco (i, j).
Subproblema:

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij} - \sum_{i \in V \setminus \{0\}} \theta_i y_i + \mu \tag{5.29}$$

Sujeito a:

$$\sum_{j \in V} x_{ij} = y_i \qquad \forall i \in V \setminus \{0\} \tag{5.30}$$

$$\sum_{j \in V} x_{ji} = y_i \qquad \forall i \in V \setminus \{0\} \tag{5.31}$$

$$\sum_{j \in V \setminus \{0\}} x_{0j} = 1 \tag{5.32}$$

$$\sum_{j \in V \setminus \{0\}} x_{j0} = 1 \tag{5.33}$$

$$\sum_{j \in V} P_{ij} - \sum_{j \in V} P_{ji} = p_i y_i \qquad \forall i \in V \setminus \{0\} \tag{5.34}$$

$$\sum_{j \in V} D_{ji} - \sum_{j \in V} D_{ij} = d_i y_i \qquad \forall i \in V \setminus \{0\} \tag{5.35}$$

$$P_{ij} + D_{ij} \leq Q x_{ij} \qquad \forall (i,j) \in A \tag{5.36}$$

$$P_{ij}, D_{ij} \geq 0 \qquad \forall (i,j) \in A \tag{5.37}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{5.38}$$

$$y_i \in \{0,1\} \qquad \forall i \in V \setminus \{0\} \tag{5.39}$$

Se a rota passa pelo consumidor i então deve chegar e sair um arco desse consumidor como indicado nas restrições (5.30) e (5.31). Deve sair e chegar um arco no depósito (5.32) e (5.33) . As restrições (5.34) e (5.35) são responsáveis pela conservação de fluxo de carga de coleta e entrega respectivamente. A restrição (5.36) garante que a capacidade do veículo não seja violada na rota.

A função objetivo do Subproblema corresponde ao custo reduzido da rota. O vetor de variáveis θ corresponde às variáveis duais associadas as restrições de cobertura (5.26) e a variável μ é a variável dual associada a restrição do número de veículos (5.27). Se uma rota de custo $c < 0$ for encontrada, ela é inserida no Problema Mestre Restrito, senão, a solução do Mestre é ótima. O Subproblema corresponde ao Problema do Caminho Elementar Mínimo com Restrições de Recurso e será discutido mais detalhadamente na próxima seção.

Em nossa implementação, a restrição (5.27) não é usada, ao invés disso, deixamos o algoritmo encontrar o número ótimo de veículos por conta própria. Dessa maneira, não se tem a variável μ na função objetivo do Subproblema.

5.3 Resolvendo o Subproblema

5.3.1 O Problema do Caminho Elementar Mínimo com Restrições de Recurso

No Problema do Caminho Elementar Mínimo com Restrições de Recurso (*Elementary Resource Constrained Shortest Path Problem*, ERCSPP), o caminho elementar (sem ciclos) entre dois vértices específicos deve ser encontrado em um digrafo com possíveis arcos de custo negativo. Um conjunto de restrições sobre um conjunto de recursos deve ser satisfeito. Recursos podem representar tempo, carga do veículo, ou outras quantidades. Um consumo de cada recurso está associado à cada arco e as restrições de recurso restringem os valores das quantidades de recursos nos nós do caminho [6, 19, 27, 39]. Se a restrição de elementaridade for relaxada, isto é, se forem permitidos ciclos na solução, tem-se o Problema do Caminho Mínimo com Restrições de Recurso (*Resource Constrained Shortest Path Problem*, RCSPP), e existe algoritmo pseudo-polinomial para resolvê-lo. Porém, o ERCSPP é NP-difícil no sentido forte caso hajam ciclos de custo negativo no grafo [6, 16, 39].

A versão básica do ERCSPP é definida da seguinte maneira. Um digrafo $D = (V, A)$ com custos $\hat{c}_{ij}$, para todo $(i, j) \in A$, e um conjunto R de recursos com quantidades disponíveis $q_r \geq 0$, para todo $r \in R$, são dados. O consumo do recurso $r \in R$ quando o arco (i, j) é atravessado é dado pela função $w_{ij}(r)$. O objetivo do problema é encontrar o caminho elementar de menor custo de s a t, $s, t \in V$, tal que o consumo total de cada recurso r não seja maior que q_r [6].

Geralmente, para evitar ciclos, um conjunto adicional U de recursos com disponibiliade unitária, chamados *node resources* [6] é usado. Cada recurso $u \in U$ é associado a um vértice $v \in V$. Quando um vértice é visitado, seu recurso associado é consumido e o vértice não pode ser visitado novamente. Uma abordagem de Programação Dinâmica comum usando essa formulação terá tempo de execução exponencial no número desses recursos [6].

No caso do VRPSPD, tem-se as restrições de que a carga de coleta e entrega não podem ultrapassar a capacidade do veículo. Essas restrições são modeladas através de dois recursos dependentes, r_p e r_{max}, que representam a carga total de coleta no veículo e a carga máxima em algum ponto do caminho até o ponto atual, respectivamente. Dado um caminho qualquer terminado no vértice i, seja $\mathcal{R}_i(r), r \in \{r_p, r_{max}\}$ a quantidade total consumida do recurso r até o momento por este caminho. Ao ser estendido a um vértice j, tem-se que $\mathcal{R}_j(r_p) = \mathcal{R}_i(r_p) + p_j$, em que p_j é a quantidade de coleta do consumidor j. Já a nova carga máxima será dada por $\mathcal{R}_j(r_{max}) = \max\{\mathcal{R}_i(r_{max}) +$

$d_j, \mathcal{R}_i(r_p) + p_j\}$, em que d_j é a quantidade de entrega do consumidor j. Isso é, a carga total de coleta do veículo aumenta em p_j e a carga máxima do veículo ao longo do caminho, logo antes de passar por j, será aumentada em d_j, portanto, em j a carga máxima será dada pelo máximo entre a carga total de coleta atual e a carga máxima antes de chegar em j. Dessa maneira, tem-se uma função não-linear, porém não decrescente para o recurso que representa a carga máxima. O veículo parte com $r_p = 0$ e $r_{max} = 0$ e em nenhum ponto do caminho r_p ou r_{max} devem ser superiores à capacidade do veículos [27, 39].

Problemas de caminho mínimo com restrições de recurso são geralmente resolvidos por programação dinâmica [16, 27, 39]. A eficiência desses algoritmos depende fortemente do número de caminhos parciais gerados. Então, é desejável que caminhos inúteis, isto é, caminhos que não levarão a algum caminho ótimo, sejam identificados e descartados [27, 39].

O (E)RCSPP é de grande importância para os métodos de Geração de Colunas para VRPs. Nesses métodos o VRP é decomposto em um problema de particionamento ou cobertura de conjuntos, chamado Problema Mestre, e um problema de geração de rotas viáveis, chamado Subproblema ou Problema de *Pricing*. O Subproblema pode ser formulado como um ERCSPP, os custos reduzidos são modelados pelos custos dos arcos e as restrições que definem o tipo de VRP são modeladas pelos recursos [6, 39]. O processo de *Pricing* procura por rotas com custo (reduzido) negativo para serem adicionadas no Problema Mestre, se não houver, a solução atual é ótima. Se a restrição de elementaridade for relaxada, o RCSPP é resolvido ao invés do ERCSPP (como na maioria dos algoritmos da literatura), a solução inteirá ótima ainda assim não conterá ciclos. Entretanto, embora os algoritmos para resolver o RCSPP sejam mais rápidos, caminhos elementares produzem limites mais fortes [9].

Devido aos trabalhos publicados nos últimos anos, é possível resolver os subproblemas envolvendo ERCSPP de maneira menos custosa, produzindo limites melhores em cada nó. Neste trabalho, o interesse é a resolução do ERCSPP para geração de rotas com custo reduzido negativo para o VRPSPD. Porém, o conteúdo discutido ao longo desta seção é válido para qualquer tipo de VRP. Nas próximas seções, discutimos estratégias para resolução do ERCSPP e contribuições próprias são apresentadas para o caso em que se está interessado em caminhos de custo negativo.

5.3.2 Estratégias para resolução do ERCSPP

Em algoritmos de programação dinâmica para o ERCSPP, são atribuídos *labels* a caminhos. Um *label* l é uma tupla que pode assumir a seguinte forma $(c, \mathcal{R}, \mathcal{U})$, sendo

c o custo do caminho, $\mathcal{R}$ um vetor com o consumo atual de cada recurso em R e $\mathcal{U}$ um vetor com o consumo atual de recursos unitários em U. Cada vértice $i \in V$ tem associado um conjunto de *labels* cujos caminhos terminam em i. *Labels* são sistematicamente estendidos em todas as direções viáveis até que nenhum *label* possa mais ser estendido. Quando nenhum *label* puder mais ser estendido, o caminho ótimo é dado pelo *label* de mínimo custo no conjunto de *labels* associado ao vértice destino t. A eficiência desses algoritmos depende do número de *labels* gerado. Assim, estratégias para identificar e descartar *labels* que subsequentemente não podem ser estendidos ao caminho ótimo devem ser desenvolvidas. A principal regra para eliminação de *labels* é a Regra de Dominância. Sejam $l = (c, \mathcal{R}, \mathcal{U})$ e $l' = (c', \mathcal{R}', \mathcal{U}')$ dois labels associados ao vértice v, l domina l' se:

$$c \leq c' \tag{5.40}$$

$$\mathcal{R} \leq \mathcal{R}' \tag{5.41}$$

$$\mathcal{U} \leq \mathcal{U}' \tag{5.42}$$

e pelo menos uma das desigualdades é estrita [39].

Outras estratégias podem ser usadas. Por exemplo, quando os consumos de recurso $w_{ij}(r)$ obedecem a desigualdade triangular, se um *label* l associado ao vértice i não puder ser estendido ao vértice j devido à escassez de algum recurso, isto é, se $\mathcal{R}(r) + w_{ij}(r) > q_r$, o recurso unitário do vértice j no *label* l pode ser ajustado em 1, pois o vértice j não pode mais ser visitado [6, 39]. Com essa estratégia, um número maior de *labels* será dominado, uma vez que a satisfação da restrição (5.42) se tornará um pouco mais fácil.

De acordo com a regra de dominância, um *label* l com caminho p não será dominado por um *label* l' com caminho p' caso p não passe por algum vértice de p'. Mesmo que l' consuma uma quantidade menor de recursos e tenha menor custo, p ainda pode passar por vértices que p' não pode mais, assim, pode levar a melhores caminhos que p'. Porém, se um limite superior no lucro que p pode obter passando por esses vértices for conhecido, e seu custo atual menos esse lucro ainda for maior que o custo de l', então p pode ser descartado. Essa idéia foi introduzida em [9], no qual é apresentada uma estimativa para o caso em que a diferença é de um vértice.

Outra estratégia, que traz grande redução ao tempo computacional, foi desenvolvida independentemente por Boland et al. [6] com o nome *State Space Augmenting Algorithm* e por Righini e Salani [39] com o nome *Decremental State Space Relaxation*. A idéia é usar recursos unitários apenas para alguns vértices, ao invés de todos. Inicialmente o algoritmo começa com um conjunto U vazio e o problema é resolvido,

como $U = \emptyset$, esse é o RCSPP. Após isso, se a solução contiver ciclos, U é atualizado com a inclusão de recursos unitários para alguns dos vértices no ciclo e o problema é resolvido novamente. O processo é repetido até que uma solução sem ciclos seja obtida. Righini e Salani testaram a inclusão de todos os vértices de todos os ciclos em cada iteração. Boland et al. testaram diferentes estratégias, a inclusão apenas do vértice com maior multiplicidade (visitado mais vezes) no caminho ótimo foi a estratégia com melhor performance.

Quando o RCSPP é resolvido ao invés do ERCSPP, ciclos de tamanho 2 (ciclos contendo 2 vértices) são tipicamente prevenidos (ver [14] para mais detalhes em prevenção de ciclos de tamanho 2). Outra estratégia é o uso de *State Space Relaxation* [39]. O espaço de estados é projetado em um espaço de dimensão menor, isto é, cada *label* $(c, \mathcal{R}, \mathcal{U})$ é mapeado em um *label* $(c, \mathcal{R}, \sigma)$, sendo $\sigma = \sum_{i \in U} \mathcal{U}(i)$ o número de vértices já visitados, e deve respeitar a desigualdade $0 \leq \sigma \leq |V|$. Dessa forma, o número de *labels* é drasticamente reduzido, porém, a informação sobre quais vértices já foram visitados é perdida nos *labels* de dimensão menor e a solução pode conter ciclos. *State Space Relaxation* e eliminação de ciclos de tamanho 2 podem ser usadas em conjunto com o *State Space Augmenting Algorithm*(*Decremental State Space Relaxation*) discutido acima, criando um algoritmo bastante robusto.

5.3.3 Novas estratégias para aceleração da obtenção de colunas negativas

Nesta seção são propostas três novas estratégias, que visam encontrar mais rapidamente colunas de custo negativo.

5.3.3.1 Eliminação de vértices do grafo

Em algoritmos de Geração de Colunas, durante a resolução do (E)RCSPP como Subproblema, os custos dos arcos são dados por $\hat{c}_{ij} = c_{ij} - \theta_j$, em que θ_j é uma variável dual (prêmio) associada ao vértice j. Se θ_j for igual a 0, e os custos $c_{ij} \geq 0$ dos arcos e consumos de recursos obedecerem a desigualdade triangular, o vértice j pode ser eliminado do subproblema. Isto é válido, pois, nunca haverá redução do custo passando-se pelo vértice j, ou seja, sempre será mais barato ir direto a um vértice k do que passar por j primeiro, uma vez que $c_{ij} - \theta_j + c_{jk} - \theta_k \geq c_{ik} - \theta_k$, já que $\theta_j = 0$ e os custos obedecem a desigualdade triangular.

5.3.3.2 Uma regra de eliminação baseada no limite inferior do caminho mínimo

Suponha que seja mantido nos *labels* o número de vértices que o caminho ainda pode visitar. Um *label* terá a forma $(c, \mathcal{R}, \mathcal{S}, n)$, sendo n o número de vértices que ainda podem ser visitados. O algoritmo é inicializado com um *label* inicial de custo e consumo de recursos nulos e $n = |V| - 1$ no vértice inicial s. Cada vez que o caminho for estendido, n é decrescido de uma unidade, se um *label* possuir $n = 0$, não pode mais ser estendido, similar ao que é feito na *State Space Relaxation* (veja seção 5.3.2).

Dado um *label* $l = (c, \mathcal{R}, \mathcal{U}, n)$ associado ao vértice i, considere que um caminho de custo negativo esteja sendo procurado. Se um limite inferior α no custo do caminho mínimo de i ao vértice destino t usando no máximo n vértices for conhecido e $c+\alpha \geq 0$, então l pode ser descartado, já que não levará a um caminho de custo negativo.

Um limite inferior pode ser facilmente computado. Seja $\hat{c}_{ij} = c_{ij} - \theta_j$ e A^- o conjunto *ordenado* de maneira não descrescente dos arcos (i, j) com $\hat{c}_{ij} < 0$, o limite é dado por:

$$\alpha(n) = \begin{cases} \sum_{(i,j)\in A^-} \hat{c}_{ij} & \text{se } n \geq |A^-| \\ \sum_{(i,j)\in A_n^-} \hat{c}_{ij} & \text{se } n < |A^-|. \end{cases}$$

Sendo A_n^- o conjunto com os n primeiros arcos de A^-. É fácil ver que $\alpha(n)$ representa um limite inferior no caminho mínimo com até n arcos pois, para qualquer arco negativo no caminho mínimo, é possível associar um arco dentre os usados no cálculo de α com custo menor ou igual e o restante dos arcos do caminho mínimo são positivos.

5.3.3.3 Um algoritmo que usa uma nova regra de dominância

A complexidade do RCSPP é aumentada quando os recursos unitários e a desigualdade (5.42) são adicionados para modelar o ERCSPP. A desigualdade (5.42) é difícil de satisfazer e, assim, o número de *labels* dominados cai drasticamente. A idéia da presente estratégia é usar uma regra de dominância incompleta, testando a desigualdade (5.42) apenas para alguns dos recursos em U. Isto é feito na tentativa de eliminar mais *labels*. Entretanto, *labels* que levariam ao caminho ótimo podem também ser eliminados. Ainda assim, essa idéia pode ser aplicada aos casos em que se está interessado em caminhos de custo negativo, porque mesmo que a solução ótima e outras sejam eliminadas, caminhos de custo negativo ainda podem ser encontrados e adicionados ao problema mestre. Se caminhos de custo negativo não forem encontrados, isso não significa que eles não existam, é preciso resolver o problema de novo, usando um maior subconjunto de U na desigualdade (5.42) até que se encontre o caminho desejado ou todos os recursos em U já estejam sendo usados na regra de dominância, e nesse caso, se não

for encontrado um caminho de custo negativo, então não existe nenhum. Note que, todos os recursos em U continuam sendo verificados normalmente durante a extensão de *labels* e, dessa maneira, os vértices com recurso em U continuarão sendo visitados no máximo uma vez.

Para formalizar a idéia acima. Seja $\hat{U} \subseteq U$ um subconjunto de recursos unitários. Esses são os recursos cujas quantidades serão testadas na regra de dominância. Dado um *label* l, o vetor $\hat{\mathcal{U}}$ possui as quantidades atuais de recursos de $\hat{U}$ nesse *label*. Os *labels* terão então a forma $l = (c, \mathcal{R}, \mathcal{U}, \hat{\mathcal{U}})$. Um *label* l dominará um *label* l' se

$$c \leq c'$$
$$\mathcal{R} \leq \mathcal{R}'$$
$$\hat{\mathcal{U}} \leq \hat{\mathcal{U}}'$$

e uma das desigualdades for estrita.

Toda vez que não forem encontrados caminhos de custo negativo, mais recursos de U são adicionados em $\hat{U}$ e o problema é resolvido outra vez, até que $U = \hat{U}$. O algoritmo 6 exibe o procedimento.

algoritmo 6 Dominância Incompleta

Entrada: Instância do ERCSPP em que se está interessado em um caminho de custo negativo
Saída: Conjunto C de caminhos de custo negativo

```
1: Û = ∅
2: C = ∅
3: repita
4:    resolva o ERCSPP usando a regra de dominância modificada
5:    C = caminhos de custo negativo encontrados
6:    se C = ∅ então
7:       atualize Û
8:    fim se
9: até C ≠ ∅ ou Û = U
10: retorne C
```

O conjunto $\hat{U}$ pode ser atualizado de diversas maneiras. Uma maneira possível é dividir U em n subconjuntos disjuntos de tamanhos iguais $U = U_1 \cup U_2 \cup \ldots \cup U_n$. Então, o método tem início com $\hat{U} = \emptyset$ e, ao fim da iteração k, se não forem encontrados caminhos de custo negativo, um novo subconjunto é adicionado, $\hat{U} = \hat{U} \cup U_k$. Se $k = n + 1$ e nenhum caminho negativo foi encontrado, então não há nenhum, já que $\hat{U} = U$.

Ao invés de se resolver um ERCSPP completo ($|U| = |V|$), pode ser usado o

State Space Augmenting Algorithm (*Decremental State Space relaxation*) descrito na seção 5.3.2. Deve ser notado que, neste algoritmo é usado um conjunto de recursos unitários U, tal que os vértices associados a recursos em U formam um subconjunto de V, na extensão e dominância de *labels*. Enquanto na estratégia proposta, é usado U na extensão e $\hat{U} \subseteq U$ na dominância.

5.3.4 Avaliação das Estratégias Propostas na Resolução da Relaxação Linear do Problema Mestre

Nesta seção são apresentados experimentos computacionais para avaliar as estratégias propostas. A Relaxação Linear da reformulação do VRPSPD introduzida na seção 5.2 é resolvida por Geração de Colunas.

São testadas cinco implementações diferentes. A implementação básica inclui todas as estratégias discutidas na seção 5.3.2. A segunda inclui a implementação básica e a estratégia de eliminação de vértices baseada nos custos duais. A terceira, além da implementação básica, faz uso de limites inferiores no caminho mínimo para a eliminação de *labels*. A quarta inclui a implementação básica e a estratégia de uso de regra de dominância mais simples (o conjunto de *node resources* é dividido em dois subconjuntos disjuntos). Finalmente, a quinta implementação utiliza todas as estratégias discutidas.

A cada iteração da geração de colunas, apenas a primeira coluna negativa encontrada é introduzida no problema mestre. Note que, com o uso da estratégia *State Space Augmenting Algorithm*(*Decremental State Space Relaxation*), discutida na seção 5.3.2, é necessário resolver o problema diversas vezes até que seja encontrado um caminho elementar. Esperar por n caminhos negativos, sem garantias de que serão elementares, em todas as etapas do processo, pode sair muito caro e portanto esse tipo de estratégia não foi adotada.

As instâncias de Salhi e Nagy [42] são utilizadas para os experimentos. Como o número de consumidores em cada problema pode ser muito grande, são considerados apenas os primeiros 30 e 40 consumidores de cada problema para os experimentos. Além disso, apenas as instâncias sem restrição na distância máxima percorrida são usadas. O problema mestre de cobertura de conjuntos é resolvido com o software ILOG CPLEX 10.2 sem ajustes especiais de parâmetros.

Os algoritmos foram codificados em c++ e compilados com o compilador g++. Experimentos computacionais foram conduzidos em uma máquina Intel Xeo Core 2 Quad, 2,00GHz, 8GB de RAM e sistema operacional Ubuntu.

O tempo limite para a resolução de cada problema foi de 3600 segundos. Como

o CPLEX pode usar diferentes parâmetros em execuções diferentes, resultando em diferentes números de resoluções do subproblema, cada implementação foi executada 10 vezes para cada problema.

Na tabela 5.1, as médias dos tempos computacionais nas 10 execuções para problemas com 30 consumidores são exibidas, para cada implementação. Celulas com "*"indicam que o problema não foi resolvido dentro de 3600 segundos em nenhuma das 10 execuções. Quando uma instância não é resolvida, é considerado um tempo computacional de 3600 segundos. Na tabela 5.2, as melhoras médias dos tempos computacionais em relação à implementação básica são exibidas. As melhoras são calculadas como $(t_i/t_1) * 100/t_1$, em que t_i é o tempo computacional da implementação i

É visto que, na média, a segunda implementação teve comportamento similar à implementação básica. A terceira implementação foi capaz de resolver a Relaxação Linear do Problema Mestre na instância cmt11, que a implementação básica não foi capaz de resolver, em 16,5 segundos e reduziu o tempo computacional médio em 27,2%. Um bom desempenho foi demonstrado pela quarta implementação, essa implementação melhorou o tempo computacional médio em todas as instâncias resolvidas pela implementação básica, e obteve uma melhora média de 58,5%, porém, não foi capaz de resolver a instância cmt11. Finalmente, a combinação de todas as estratégias na quinta implementação ocasionou em uma redução bastante significativa de 81,3% do tempo computacional médio, e também foi capaz de resolver a instância cmt11. Nenhuma das implementações foi capaz de resolver a instância cmt12 em menos de 3600 segundos.

Na tabela 5.3 são exibidos os tempos computacionais médios para problemas com 40 consumidores. A tabela 5.4 mostra as melhoras médias do tempo computacional em relação à implementação básica.

Novamente, a segunda implementação se comportou de maneira similar à implementação básica. A terceira implementação reduziu o tempo computacional em 23,4% na média e resolveu as instâncias cmt05 e cmt11, que não foram resolvidas pela implementação básica. Já a quarta implementação resolveu as instâncias cmt04 e cmt05, não resolvidas pela implementação básica, e reduziu o tempo computacional médio em 61,9% na média. A quinta implementação reduziu o tempo computacional em 72,4% e foi capaz de resolver as instâncias cmt04, cmt05 e cmt11 em menos de 3600 segundos. Nenhuma implementação foi capaz de resolver as instâncias cmt03 e cmt12 dentro do tempo limite especificado.

	Implementações				
	1	2	3	4	5
cmt01	106,7	125,8	103,0	46,8	**25,9**
cmt02	33,7	36,6	29,7	11,2	**9,2**
cmt03	3334,0	3274,3	3222,8	1200,0	**806,8**
cmt04	3143,9	1667,9	1686,4	**853,9**	892,3
cmt05	2394,1	2491,1	2422,5	197,6	**163,1**
cmt11	3600*	3600*	**16,5**	3600*	19,3
cmt12	3600*	3600*	3600*	3600*	3600*
média	2316,0	2113,6	1582,9	1358,5	**788,1**

Tabela 5.1. Tempos computacionais médios (em segundos) para as instâncias com 30 consumidores

	Implementações			
	2	3	4	5
cmt01	17,9	-3,4	-56,1	**-75,7**
cmt02	8,6	-11,8	-66,7	**-72,7**
cmt03	-1,7	-3,3	-64	**-75,8**
cmt04	-46,9	-46,3	**-72,8**	-71,6
cmt05	4	1,1	-91,7	**-93,1**
cmt11	0	**-99,5**	0	-99,4
cmt12	0	0	0	0
média	-3	-27,2	-58,5	**-81,3**

Tabela 5.2. Melhoras médias do tempo computacional (%) para instâncias com 30 consumidores.

5.3.5 Conclusão

Nesta seção, foram propostas três novas estratégias para acelerar a resolução do ERCSPP, particularmente no caso em que o ERCSPP é usado em algoritmos de Geração de Colunas, dessa maneira, busca-se caminhos de custo negativo. A primeira estratégia consiste em remover alguns vértices do grafo baseado nos custos duais associados. A segunda estratégia consiste em utilizar um limite inferior no caminho mínimo até o vértice destino para eliminar *labels*. A terceira estratégia consiste em usar uma regra de dominância incompleta. Para avaliar a efetividade das estratégias propostas, elas foram empregadas na resolução da Relaxação Linear do Problema Mestre do VRPSPD por Geração de Colunas. Na média, a primeira estratégia proposta não demonstrou efeito significativo nas instâncias testadas, a segunda e terceira estratégias tiveram um bom desempenho e a combinação das estratégias propostas levou a uma grande redução do tempo computacional.

	Implementações				
	1	2	3	4	5
cmt01	3095,3	2818,1	2529,8	563,8	**440,8**
cmt02	866,0	947,4	778,6	206,2	**162,6**
cmt03	3600*	3600*	3600*	3600*	3600*
cmt04	3600*	3600*	3600*	**1327,5**	2970,4
cmt05	3600*	3600*	3582,0	402,0	**385,6**
cmt11	3600*	3600*	**411,0**	3600*	411,1
cmt12	3600*	3600*	3600*	3600*	3600*
média	3137,3	3109,3	2585,9	1899,9	**1652,9**

Tabela 5.3. Tempos computacionais médios (em segundos) para as instâncias com 40 consumidores

	Implementações			
	2	3	4	5
cmt01	-8,9	-18,2	-81,7	**-85,7**
cmt02	9,3	-10	-76,1	**-81,2**
cmt03	0	0	0	0
cmt04	0	0	**-63,1**	-17,4
cmt05	0	-0,5	-88,8	**-89,2**
cmt11	0	**-88,5**	0	-88,5
cmt12	0	0	0	0
média	0,0	-23,4	-61,9	**-72,4**

Tabela 5.4. Melhoras médias do tempo computacional (%) para instâncias com 40 consumidores.

5.4 *Branch-and-price*

5.4.1 Introdução

Nas seções anteriores foram apresentados o modelo e a reformulação do problema e os métodos para a resolução do Subproblema. A seguir, são discutidos os principais elementos do algoritmo *Branch-and-price* desenvolvido. A figura 5.1 exibe um diagrama com o funcionamento do algoritmo, no qual x^i denota a Relaxação Linear do Problema Mestre Restrito do problema/nó i e $c(x^i)$ o seu custo, x^* denota a melhor solução inteira conhecida, z_{ub} o custo da melhor solução inteira conhecida, π^i o pai de um problema/nó i, R^i o conjunto de rotas gerado em um problema/nó i da árvore e L o conjunto de problemas candidatos.

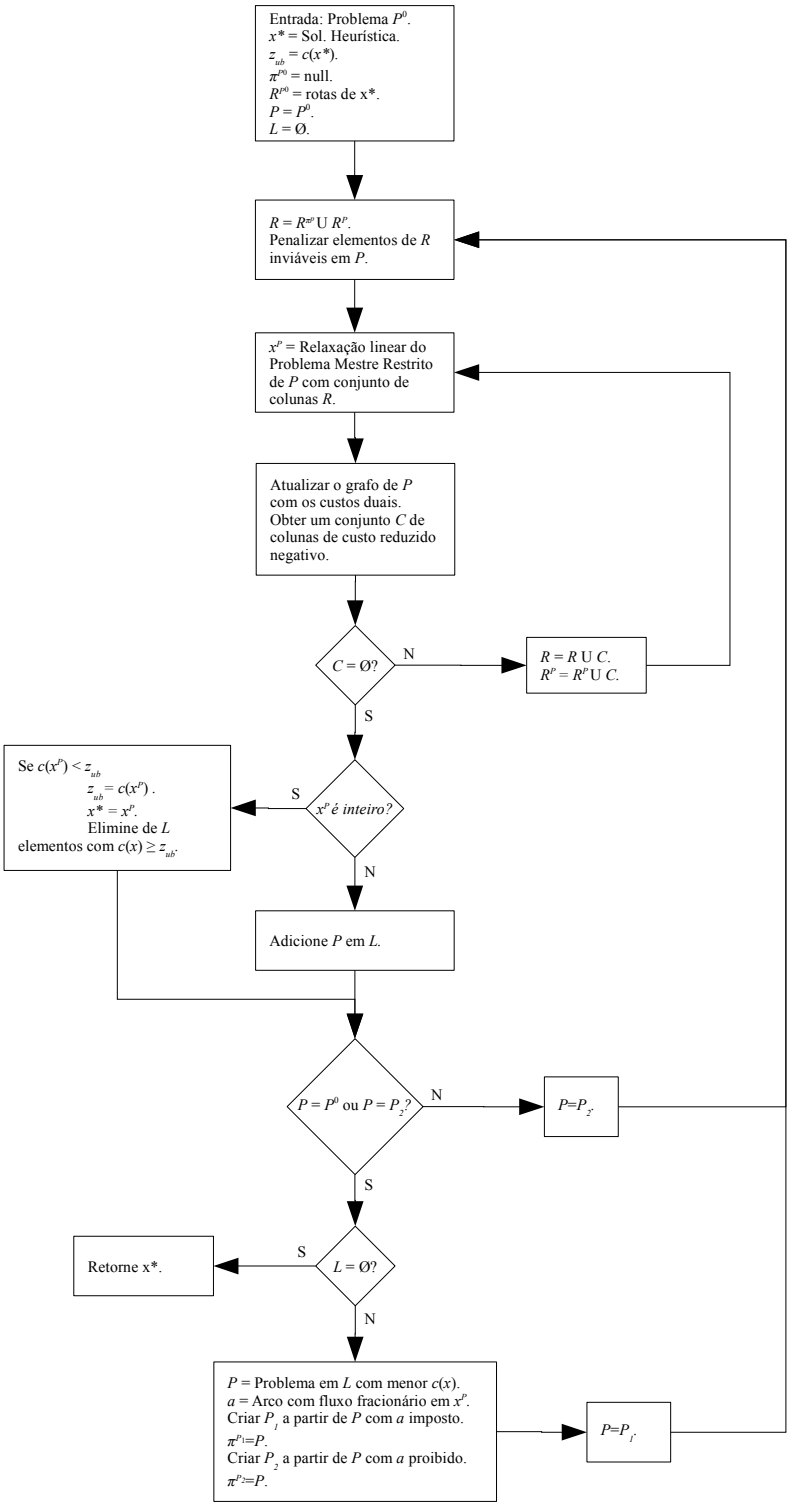

Figura 5.1. Fluxograma do algoritmo *Branch-and-price* desenvolvido.

5.4.2 Inicialização

Como dito anteriormente, o conjunto de todas as rotas possíveis não é explicitamente conhecido. Desse modo, o algoritmo *Branch-and-price* implementado é inicializado com o conjunto de rotas dado por uma solução heurística gerada pela heurística descrita no capítulo 4, que usa a estratégia baseada em GLS. Assim, um bom limite superior é conhecido e espera-se que mais nós sejam podados na árvore de busca.

5.4.3 Branching

Se a solução ótima da Relaxação Linear do Problema Mestre possui variáveis λ fracionárias, é necessário uma estratégia para se chegar à solução inteira. É adotada uma estratégia de *branching* nos arcos, isto é, um arco com fluxo fracionário é escolhido e são criados dois problemas filhos, esse arco é imposto na solução de um deles e proibido na do outro. Seja esse arco denotado por (i, j). Para impor o arco, todo o arco (i, k) tal que $k \neq j$ e (k, j) tal que $k \neq i$ é removido do grafo. Para proibir o arco (i, j) em si, basta removê-lo do grafo.

Para escolher o arco no qual ocorrerá o *branching* monta-se um conjunto F de arcos candidatos, composto por arcos com fluxo fracionário, ou seja, $F \subseteq \{(i, j)|0 < \omega_{ij} < 1\}$, em que $\omega_{ij} = \sum_{r \in R'} \lambda_r x_{ij}^r$ representa o fluxo no arco. O conjunto F é composto por arcos de fluxo fracionário próximo ao valor de $0,5$ (arcos distantes da integralidade). Inicialmente, são encontrados os arcos com menor valor positivo $\delta_l = 0,5 - \omega_{ij}$ e $\delta_r = \omega_{ij} - 0,5$. O conjunto F é dado por $F = \{(i, j)|0,85 \times \delta_l \leq \omega_{ij} \leq (1 - \delta_r) \times 0,15 + \delta_r; 0 < \omega_{ij} < 1; (i, j) \in A\}$, isto é, são escolhidos os arcos com fluxo até 15% menor que δ_l e maior que δ_r. Foram testados outros valores, este porém proporcionou melhores resultados.

Para cada arco, $(i, j) \in F$ calcula-se f_{ij}^1 e f_{ij}^2, limites inferiores no custo da solução com o arco imposto e com o arco proibido, respectivamente. É escolhido então o arco $e = \arg\max_{(i,j) \in F}\{\min f_{ij}^1, f_{ij}^2\}$, ou seja, tenta-se escolher o arco que proporcionaria o maior aumento do limite inferior na sub-árvore do nó atual, levando a maiores chances de poda dos nós subsequentes. Esta é uma estratégia clássica para a escolha do arco, porém, é usado como limite inferior um limite calculado a partir da realização de algumas iterações da geração de colunas com o arco imposto ou proibido. Não foi encontrado qualquer outro trabalho que use esse limite para escolher o arco. Seja m^k o valor da solução ótima do Mestre Restrito na iteração k, s^k o custo da coluna ótima na iteração k, θ_i^k a variável dual associada a restrição de cobertura do consumidor i na iteração k e seja K um limite superior no número ótimo de veículos nesse nó da árvore, tem-se o limite: $Ks^k + \sum_{i \in V \setminus \{0\}} \theta_i^{k-1} + \mu = Ks^k + m^{k-1}$ [15]. Obviamente,

quanto mais iterações, mais preciso é esse limite. Dessa maneira, para cada arco em F, a idéia é fazer algumas iterações de geraçã o de colunas com o arco imposto e proibido e calcular os limites inferiores f_{ij}^1 e f_{ij}^2 nesses problemas como descrito.

Gera-se uma coluna ótima para cada um dos dois problemas filhos correspondentes a cada arco. Entretanto, gerar uma coluna elementar pode levar muito tempo, o RCSPP é então usado ao invés do ERCSPP. Como o valor de K não é conhecido, é usado o valor da solução heurística inicial. Porém, ao se usar o número de veículos dado pela solução da heurística, não existe mais garantia de que o valor calculado represente um limite inferior. Dessa maneira, os valores f_{ij}^1 e f_{ij}^2 são tratados apenas como estimativas dos limites inferiores.

5.4.4 Escolha do nó

Na ramificação, é usada uma estratégia de busca em largura, uma vez que o algoritmo já é inicalizado com uma boa solução primal. O nó com menor limite inferior é escolhido para reduzir mais rapidamente o *gap* entre os limites superior e inferior atuais. Dessa maneira o algoritmo pode ser parado mais rapidamente com uma determinada tolerância da otimalidade caso seja necessário.

Para cada nó, é mantido um conjunto com as colunas de seus descendentes na árvore. Antes da resolução do nó, essas colunas são carregadas e são impostas penalizações de valor infinito nas colunas que não podem ser utilizadas naquele nó devido a bloqueios e imposições de arcos.

5.4.5 Estabilização

Métodos de estabilização de Geração de Colunas buscam acelerar a convergência dos métodos de Geração de Colunas (Veja [8]). Foi implementado o método proposto por Du Merle et al. [18], no qual a idéia é colocar uma caixa ao redor das variáveis duais e penalizar linearmente caso as variáveis saiam dessa caixa. Em experimentos preliminares, a estratégia não demonstrou redução no tempo computacional e foi removida da implementação final.

5.5 Resultados Computacionais

Nossos experimentos foram realizados em uma máquina máquina Intel Xeon Core 2 Quad, 2,00GHz, 8GB de RAM e sistema operacional Ubuntu, foi usado o compila-

dor g++. Para resolução do problema mestre é usado o software CPLEX 10.2, sem nenhuma configuração especial.

Foram usadas as mesmas instâncias do trabalho de Salani [41], já descritas na seção 4.3.

A heurística desenvolvida na seção 4 foi testada para a obtenção de colunas negativas na resolução do subproblema. Porém, a heurística encontra poucas rotas com custo reduzido negativo, e acaba aumentando o tempo computacional na resolução dos subproblemas. O fato da heurística conseguir poucas rotas de custo negativo se deve ao balanceamento que faz no custo da solução como um todo, ao invés de focar na otimização de apenas uma das rotas.

Como descrito na seção 5.3.4, apenas o primeiro caminho de custo negativo encontrado em cada resolução do Subproblema é adicionado ao Problema Mestre Restrito. Foi usada a implementação com todas as estratégias básicas e as estratégias propostas.

A seguir, o algoritmo proposto é comparado com o algoritmo *Branch-and-cut-and-price* desenvolvido por Salani [41]. As tabelas 5.5, 5.6, 5.7 e 5.8 exibem os resultados do algoritmo proposto (coluna BP) e os resultados obtidos pelo algoritmo com resolução exata do *pricing* e uso de cortes *2-path* daquele trabalho (coluna Salani). Os resultados de Salani foram obtidos em uma máquina Intel Pentium IV, 1,6GHz, 512Mb de Ram e sistema operacional Red Hat. As colunas lb_{root} e t_{root} indicam respectivamente o limite inferior obtido e tempo gasto (em segundos) no nó raiz, ub^* indica o valor da solução inteira ótima, t_{tree} e n_{nodes} indicam respectivamente o tempo gasto (em segundos) na resolução do problema como um todo e o número de nós explorados na árvore de busca.

Embora tenha gasto um maior tempo computacional em praticamente todas as instâncias, apenas uma instância não foi resolvida em menos de uma hora e a maioria foi resolvida em menos de dez segundos. O algoritmo proposto geralmente precisa resolver uma quantidade muito superior de nós, devido principalmente ao emprego dos cortes *2-path* no algoritmo de Salani, que elevam bastante os limites inferiores, possibilitando a poda dos nós. Na instância 1S_40_80_3 para cada nó resolvido pelo algoritmo de Salani, o algoritmo proposto resolveu cerca de 2899 nós. Embora não seja uma comparação leal, uma vez que em [41] são resolvidos problemas de separação em cada nó, em média, o algoritmo proposto leva 0,2s para resolver cada nó, enquanto o algoritmo de [41] leva 3,4s. Isso leva a acreditar que é possível obter uma implementação mais rápida que a de Salani com a incoporação dos cortes em nosso algoritmo.

Um fato interessante é que, apesar das soluções de Salani supostamente serem ótimas, foram encontradas soluções melhores para as instâncias 2S_20_80_1, 2S_40_50_1, 2S_40_50_3, 2S_40_80_1 e 2S_40_80_3, e a viabilidade de cada uma foi verificada.

	BP					Salani [41]		
instância	lb_{root}	t_{root}	ub^*	t_{tree}	n_{nodes}	ub^*	t_{tree}	n_{nodes}
1S_20_50_1	181689,00	0,08	181689	0,08	1	181689	0,04	1
1S_20_50_2	150332,00	0,21	151472	1,11	19	151472	0,58	5
1S_20_50_3	135474,00	0,84	136107	2,77	11	136107	2,13	1
1S_20_66_1	188700,00	0,07	189396	0,12	9	189396	0,09	1
1S_20_66_2	155226,42	0,20	155853	0,66	13	155853	0,59	1
1S_20_66_3	136489,00	0,96	136489	0,96	1	136489	1,41	1
1S_20_80_1	205167,00	0,06	210732	0,06	293	210732	0,26	11
1S_20_80_2	162726,83	0,13	166408	3,17	187	166408	0,4	3
1S_20_80_3	143101,00	0,63	147820	48,94	1013	147820	2,61	1
1S_40_50_1	346314,25	0,68	357430	614,69	12225	357430	82,41	377
1S_40_50_2	266517,07	4,39	269590	96,47	181	269590	20,09	5
1S_40_50_3	225390,59	15,88	229044	320,84	157	229044	148,06	1
1S_40_66_1	375089,00	0,39	377279	1,08	29	377279	1,49	5
1S_40_66_2	285113,29	2,73	291008	3500,2	18463	291008	38,44	11
1S_40_66_3	238784,87	5,05	241347	237,01	283	241347	90,7	17
1S_40_80_1	425911,00	0,31	425911	0,31	1	425911	0,13	1
1S_40_80_2	321866,57	1,11	324920	104,33	1339	324920	9,46	1
1S_40_80_3	267206,50	3,03	270313	8444,94	26093	270313	39,62	9

Tabela 5.5. Resultados computacionais para as instâncias do tipo 1S

	BP					Salani [41]		
instância	lb_{root}	t_{root}	ub^*	t_{tree}	n_{nodes}	ub^*	t_{tree}	n_{nodes}
1C_20_50_1	264917,50	0,04	265504	0,05	7	265504	0,01	1
1C_20_50_2	203300,75	0,07	206425	0,1	7	206425	0,05	3
1C_20_50_3	170306,33	0,13	171236	0,34	15	171236	0,12	1
1C_20_66_1	296098,50	0,06	298493	0,06	5	298493	0,01	1
1C_20_66_2	192727,00	0,06	192727	0,06	1	192727	0,02	1
1C_20_66_3	173500,00	0,13	178629	8,29	577	178629	1,13	43
1C_20_80_1	304412,00	0,04	304412	0,04	1	304412	0,01	1
1C_20_80_2	216147,50	0,06	218072	0,12	9	218072	0,03	1
1C_20_80_3	175258,00	0,09	177215	0,44	31	177215	0,3	1
1C_40_50_1	597295,00	0,19	601817	0,31	21	601817	0,05	1
1C_40_50_2	396943,88	0,34	402309	8,06	457	402309	3,03	51
1C_40_50_3	331432,10	0,72	334830	67,22	565	334830	16,4	31
1C_40_66_1	628599,00	0,19	630257	0,23	11	630257	0,02	1
1C_40_66_2	420402,42	0,35	426517	24,71	1179	426517	2,94	37
1C_40_66_3	327651,93	1,23	331459	467,78	5039	331459	43,22	145
1C_40_80_1	644874,50	0,17	647539	0,19	5	647539	0,01	1
1C_40_80_2	421200,10	0,37	424368	4,58	173	424368	2,33	25
1C_40_80_3	332208,50	1,3	332957	27,47	195	332957	1,94	1

Tabela 5.6. Resultados computacionais para as instâncias do tipo 1C

instância	BP					Salani [41]		
	lb_{root}	t_{root}	ub^*	t_{tree}	n_{nodes}	ub^*	t_{tree}	n_{nodes}
2S_20_50_1	8546,50	0,09	8769	0,58	45	8769	0,22	5
2S_20_50_2	7348,00	0,12	7348	0,12	1	7348	0,05	1
2S_20_50_3	6386,80	0,96	6445	2,70	9	6445	0,88	1
2S_20_66_1	9001,00	0,06	9129	0,12	7	9129	0,04	1
2S_20_66_2	7470,00	0,09	7470	0,09	1	7470	0,04	1
2S_20_66_3	6657,00	0,58	6890	28,79	361	6890	0,94	3
2S_20_80_1	10379,67	0,06	**10643**	0,36	49	10707	0,03	1
2S_20_80_2	8753,33	0,08	8773	0,14	5	8773	0,09	1
2S_20_80_3	7058,00	0,18	7058	0,18	1	7058	0,16	1
2S_40_50_1	16827,58	0,58	**16917**	3,05	27	18282	11,14	31
2S_40_50_2	14533,09	2,82	14603	8,76	15	14603	7,20	5
2S_40_50_3	11239,86	15,26	**11343**	68,19	25	11610	110,03	5
2S_40_66_1	17772,83	0,48	17932	4,48	63	17932	1,50	5
2S_40_66_2	15036,44	2,32	15307	252,35	1095	15307	31,51	31
2S_40_66_3	11686,78	12,25	11725	59,25	35	11725	25,80	1
2S_40_80_1	20587,50	0,38	**20660**	0,84	19	20665	0,29	1
2S_40_80_2	16937,00	0,76	17201	245,79	2803	17201	166,34	445
2S_40_80_3	13001,20	3,09	**13208**	1932,32	4983	13317	147,41	73

Tabela 5.7. Resultados computacionais para as instâncias do tipo 2S

instância	BP					Salani [41]		
	lb_{root}	t_{root}	ub^*	t_{tree}	n_{nodes}	ub^*	t_{tree}	n_{nodes}
2C_20_50_1	12720	0,05	12720	0,05	1	12720	0,01	1
2C_20_50_2	9982,25	0,07	10054	0,26	25	10054	0,04	1
2C_20_50_3	8352,67	0,12	8387	0,39	11	8387	0,22	1
2C_20_66_1	14526	0,04	14578	0,05	7	14578	0,03	1
2C_20_66_2	10802,33	0,07	10861	0,14	9	10861	0,06	1
2C_20_66_3	7947,43	0,12	8160	4,36	237	8160	1,89	17
2C_20_80_1	12674,5	0,05	12802	0,07	9	12802	0,02	1
2C_20_80_2	9904,25	0,07	10087	0,73	95	10087	0,11	1
2C_20_80_3	8086,5	0,12	8317	3,77	163	8317	0,49	1
2C_40_50_1	26904,5	0,21	26988	0,52	39	26988	0,05	1
2C_40_50_2	21496,83	0,35	21710	40,34	1373	21710	4,22	31
2C_40_50_3	15272,09	1,63	15523	3121,46	15605	15523	48,12	155
2C_40_66_1	25981	0,21	25981	0,21	1	25981	0,06	1
2C_40_66_2	20989,43	0,34	21317	119,18	4119	21317	5,42	45
2C_40_66_3	15065,01	1,39	15293	1153,97	5013	15293	62,32	129
2C_40_80_1	26122	0,22	26122	0,22	1	26122	0,07	1
2C_40_80_2	20397,45	0,34	20652	18,50	445	20652	7,77	69
2C_40_80_3	15240,99	2,64	15365	36,85	115	15365	7,18	7

Tabela 5.8. Resultados computacionais para as instâncias do tipo 2C

5.6 Conclusão

Neste capítulo foi desenvolvido um algoritmo *Branch-and-price* para a resolução do VRPSPD e algumas técnicas para a aceleração da obtenção de colunas de custo reduzido negativo no Subproblema. Essas técnicas proporcionaram um grande ganho no tempo computacional da resolução da Relaxação Linear do Problema Mestre nos problemas testados. O algoritmo *Branch-and-price* desenvolvido utiliza limites obtidos na resolução de algumas iterações da geração de colunas para escolher sobre quais arcos fará *branching*, não se tem conhecimento de outro trabalho que utilize essa estratégia. Instâncias de até 40 consumidores foram resolvidas geralmente em tempo inferior a uma hora. Os experimentos mostram que a incorporação de cortes ao algoritmo proposto pode criar um algoritmo bastante robusto.

Capítulo 6

Conclusão e Trabalhos Futuros

Neste trabalho, foi abordado o Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas. Algumas propriedades importantes do problema foram estudas e foram desenvolvidas heurísticas e um método exato.

As heurísticas desenvolvidas se baseiam em idéias de diversas metaheurísticas (GRASP, VND, ILS, SA e GLS). Essas heurísticas foram testadas em instâncias adotadas na maioria dos trabalhos envolvendo heurísticas na literatura e obtiveram resultados semelhantes aos dos melhores algoritmos. Também foram realizados testes em um conjunto de instâncias para o qual se conhece soluções ótimas, nesse conjunto, duas das heurísticas não encontraram a solução ótima em apenas 1 das 36 instâncias testadas enquanto a outra não encontrou a solução ótima em 3 das instâncias.

O algoritmo exato desenvolvido consiste em um algoritmo *Branch-and-price*, onde foram utilizadas novas estratégias para resolução do Subproblema e escolha dos arcos para realização do *branching*. Esse algoritmo foi comparado a um algoritmo *Branch-and-cut-and-price* desenvolvido em [41], e embora tenha gasto maior tempo computacional na maioria das instâncias, foi capaz de resolver quase todas em tempo inferior a uma hora. Além disso, os experimentos sugerem que o algoritmo pode ser melhorado com a introdução de cortes, o que possivelmente trará um grande ganho em termos de tempo computacional.

Em trabalhos futuros, podem ser estudadas novas maneiras para o controle da intensidade da perturbação e para a aceitação de soluções candidatas que guiem mais eficientemente a busca ILS nas heurísticas desenvolvidas. Além disso, também podem ser estudadas outras estratégias para a fase de busca local da busca ILS, na qual foi usado o método VND, podem ser empregadas até mesmo outras metaheurísticas, como Busca Tabu, por exemplo.

Na área exata, podem ser estudados outros tipos de *branching*, a utilização de

outras regras pode ajudar na diminuição do número de nós explorados pelo algoritmo desenvolvido. Além disso, O emprego de cortes tem trazido bons resultados em outros problemas de roteamento (ver [22] por exemplo), essa estratégia ainda não foi adequadamente explorada no VRPSPD e pode trazer bons resultados.

Em instâncias mais difíceis como as propostas por Salhi e Nagy [42], o algoritmo exato proposto não foi capaz de resolver o nó raíz em tempo hábil nem sequer para instâncias de 50 consumidores, para a resolução eficiente desse tipo de problema por um algoritmo *Branch-and-price* ou *Branch-and-cut-and-price* é necessário, primeiramente, o desenvolvimento de novas estratégias para a aceleração da resolução do Subproblema, o que representa mais uma opção de trabalho futuro. Uma alternativa a ser estudada é a incorporação de mais caminhos de custo reduzido negativo no problema mestre em cada iteração. Conhecendo-se um caminho de custo negativo, outros caminhos poderiam ser gerados a partir deste através de operações elementares.

O Problema de Roteamento de Veículos com *Linehauls* e *Backhauls* Mistos (VRPMB) (seção 2.2) é um problema semelhante, no qual os consumidores entregam ou recebem mercadoria, mas não as duas operações simultaneamente. Este problema representa um caso especial do VRPSDP e os algoritmos para o VRPSDP são válidos para o VRPMB. Assim, os algoritmos desenvolvidos podem ser testados em instâncias para o VRPMB e comparados a algoritmos existentes na literatura para este problema.

Referências Bibliográficas

[1] G. B. Alvarenga. *Algoritmo híbrido para os problemas de roteamento de veículos estático e dinâmico com janela de tempo*. PhD thesis, Universidade Federal de Minas Gerais, 2004.

[2] F. C. Alvelos. *Branch-and-Price and Multicommodity Flows*. PhD thesis, Universidade do Minho, 2005.

[3] L. P. Assis. Algoritmos para o problema de roteamento de veículos com coleta e entrega simultâneas. Master's thesis, Universidade Federal de Minas Gerais, 2007.

[4] G. Berbeglia, J-F Courdeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: A classification scheme and survey. *TOP*, 15(1):1–31, 2007.

[5] N. Bianchessi and G. Righini. Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers & Operations Research*, 34(2):579–587, 2007.

[6] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34:58–68, 2006.

[7] O. Braÿsy and M. Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, 2005.

[8] O. Briand, C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck. Comparison of bundle and classical column generation. *Mathematical Programming*, 113:299–344, 2008.

[9] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33:2972–2990, 2006.

[10] J-F Chen. Approaches for the vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Chinese Institute of Industrial Engineers*, 23(2):141–150, 2006.

[11] J-F Chen and T-H Wu. Vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Operational Research Sociecty*, 57(5):579–587, 2006.

[12] P. Chen, H. Huang, and X. Dong. An ant colony system based heuristic algorithm for the vehicle routing problem with simultaneous delivery and pickup. In *2nd IEEE Conference on Industrial Electronics and Applications*, 2007.

[13] N. Christofides, A. Mingozzi, and P. Toth. *Combinatorial Optimization*, chapter The Vehicle Routing Problem, pages 315–338. Wiley, 1979.

[14] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–282, 1981.

[15] M. Dell'amico and G. Righini. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2):235–247, 2005.

[16] M. Desrochers, J. Desrosiers, and M Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992.

[17] J. Dethloff. Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. *OR Spectrum*, 23(1):79–96, 2001.

[18] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1):229–237, 1999.

[19] D. Feillet, D. Pierre, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[20] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–134, 1995.

[21] M. Fleischer. Simulated annealing: past, present, and future. In *Winter Simulation Conference*, 1995.

[22] R. Fukasawa, H. Longo, J. Lysgaard, M. P. Aragão, M. Reis, E. Uchoa, and R. F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006.

[23] M. Gendreau, G. Laporte, and D. Vigo. Heuristics for the traveling salesman problem with pickup and delivery. *Computers & Operations Research*, 26(7):699–714, 1999.

[24] E. I. Gökçe. A revised ant colony system approach to vehicle routing problems. Master's thesis, Graduate School of Engineering and Natural Sciences, Sabanci University, 2004.

[25] P. Hansen and N. Mladenovic. A tutorial on variable neighborhood search. Technical report, Les Cahiers du GERAD, HEC Montreal and GERAD, 2003.

[26] D. Henderson and S. H. Jacobson. *Handbook of Metaheuristics*, chapter The Theory and Pratice of Simulated Annealing, pages 287–319. Kluwer Academic Publishers, 2003.

[27] S. Irnich and G. Desaulniers. *Column Generation*, chapter Shortest Path Problems with Resource Constraints, pages 33–65. Springer, 2005.

[28] J. Li and J-Y Zhang. A genetic algorithm to vehicle routing problem in reverse logistics. In *International Conference on Management Science & Engineering*, 2007.

[29] H. R. Lourenço, O. C. Martin, and T. Stüzle. *Handbook of Metaheuristics*, chapter Iterated Local Search, pages 321–353. Kluwer Academic Publishers, 2003.

[30] H. Min. The vehicle routing problems with simultaneous delivery and pick-up points. *Transportation Research*, 23:377–386, 1989.

[31] F. A. T. Montané and R. D. Galvão. Vehicle routing problems with simultaneous pick-up and delivery service. *Journal of the Operational Research Society of India (OPSEARCH),*, 39:19–33, 2002.

[32] F. A. T. Montané and R. D. Galvão. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operations Research*, 33(3):595–619, 2006.

[33] G. Mosheiov. The traveling salesman problem with pick-up and delivery. *European Journal of Operational Research*, 79:299–310, 1994.

[34] G. Mosheiov. Vehicle routing with pick-up and delivery: Tour partitioning heuristics. *Computers & Industrial Engineering*, 34(3):669–684, 1998.

[35] G. Nagy and S. Salhi. Heuristic algorithms for single and multiple depot vehicle routing problems with pickup and deliveries. *European Journal of Operational Research*, 162(1):126–141, 2005.

[36] H. C. B. Oliveira, G. C. Vasconcelos, and G. B. Alvarenga. Reducing traveled distance in the vehicle routing problem with time windows using a multi-start simulated annealing. In *International Joint Conference on Neural Networks*, 2006.

[37] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[38] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008.

[39] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51:155–170, 2008.

[40] S. Ropke and D. Pisinger. An unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(6):750–775, 2006.

[41] M. Salani. *Branch-and-Price Algorithms for Vehicle Routing Problems*. PhD thesis, Università Degli Studi Di Milano, 2005.

[42] S. Salhi and G. Nagy. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 50:1034–1042, 1999.

[43] A. Subramanian, L. S. Ochi, and L. A. F. Cabral. An efficient ILS heuristic for the vehicle routing problem with simultaneous pickup and delivery. Technical report, Universidade Federal Fluminense, Universidade Federal da Paraíba, 2009.

[44] P. Toth and D. Vigo. *The Vehicle Routing Problem*, chapter An Overview of Vehicle Routing Problems. SIAM, 2001.

[45] F. Vanderbeck. *Decomposition and Column Generation for Integer Programs*. PhD thesis, Université Catholique de Louvain, 1994.

[46] C. Voudouris and E. P. K. Tsang. *Handbook of Metaheuristics*, chapter Guided Local Search, pages 186–218. Kluwer Academic Publishers, 2003.

[47] A.V. Vural. A GA based meta-heuristic for capacitated vehicle routing problem with simultaneous pick-up and deliveries. Master's thesis, Graduate School of Engineering and Natural Sciences, Sabanci University, 2003.

[48] N. A. Wassan, A. H. Wassan, and G. Nagy. A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimization*, 4(4):368–386, 2008.

[49] R. H. Xie, Z. Q. Qiu, and Y. Y. Zhang. A new heuristics for VRP with simultaneous delivery and pick-up. In *Proceedings of the First International Conference on Transportation Engineering*, 2007.

[50] E. E. Zachariadis, C. D. T., and C. T. Kiranoudis. A hybrid metaheuristic for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications*, 36(2):1070–1081, 2009.

[51] T. Zhang, W-X Tian, Y-J Zhang, and X-C Zheng. An improved ant colony algorithm for VRPSDP. In *Sixth International Conference on Machine Learning Cybernetics*, 2007.