ALGORITMOS EXATOS E HEURÍSTICOS PARA PROBLEMAS SELETIVOS DE ROTEAMENTO DE VEÍCULOS COM RESTRIÇÕES DE COBERTURA

RAMON PEREIRA LOPES

ALGORITMOS EXATOS E HEURÍSTICOS PARA PROBLEMAS SELETIVOS DE ROTEAMENTO DE VEÍCULOS COM RESTRIÇÕES DE COBERTURA

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

Orientador: Alexandre Salles da Cunha

Belo Horizonte

Dezembro de 2012

Lopes, Ramon Pereira

L864a Algoritmos exatos e heurísticos para problemas seletivos de roteamento de veículos com restrições de cobertura / Ramon Pereira Lopes. — Belo Horizonte, 2012

xx, 62 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais - Departamento de Ciência da Computação.

Orientador: Alexandre Salles da Cunha

1. Computação – Teses. 2. Otimização Combinatória – 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

Algoritmos exatos e heurísticos para problemas seletivos de roteamento de veículos com restrições de cobertura

RAMON PEREIRA LOPES

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

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

PROFA. EDNA AYAKO HOSHINO
Departamento de Teoria da Computação - UFMS

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

PROF. RICARDO SARAIVA DE CAMARGO
Departamento de Engenharia de Produção - UFMG

Belo Horizonte, 17 de dezembro de 2012.

Agradecimentos

Ao meu orientador, Prof. Alexandre Salles da Cunha, agradeço pelos ensinamentos, dedicação, incentivo e boas risadas durante as reuniões. Agradeço também por ter acreditado no meu potencial, ter me guiado nesta jornada e ter ajudado a me tornar tanto um profissional quanto uma pessoa melhor. Simplesmente, muito obrigado.

A Carol, minha noiva, agradeço por ser minha cúmplice ao longo desses anos e por ter suportado esses anos de distância. Este é mais um passo que damos juntos desde a época do colegial, e parte desta vitória é sua.

Aos meus pais, Raimundo e Edna, irmã, Larissa, e Toffi agradeço por serem responsáveis pelo que sou hoje. Vocês são o meu alicerce.

Aos Profs. Ninfa e Euclimar, agradeço pelo ato de generosidade que foi fundamental para a realização deste trabalho.

A Welington e Thiago, agradeço por serem a minha família em Belo Horizonte.

A Phillippe Samer, agradeço pela amizade e por ter me ajudado em um dos momentos mais difíceis que enfrentei durante esta jornada.

Ao Prof. Thiago Noronha, Vitor e Vinícius, agradeço pela experiência de ter trabalhado com vocês no Google ROADEF/EURO Challenge 2012, cujo aprendizado foi importante para o desenvolvimento deste trabalho.

Aos amigos do LaPO, especialmente Vitor, Januario e Amadeu, agradeço pelas boas risadas e força nas horas de dificuldade.

Por fim, agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) que financiou meu mestrado e todos aqueles que direta ou indiretamente contribuíram para o desenvolvimento deste trabalho.

Resumo

Esta dissertação dedica-se ao estudo de dois problemas de Otimização Combinatória: o Problema de Roteamento de Veículos Seletivo Min-Max e o Problema de Cobertura Multi-Veículo. O primeiro surge como modelo para projetar redes de sensores eficientes do ponto de vista energético; o segundo surge como modelo para distribuir unidades móveis de saúde. Este trabalho propõe um algoritmo de Geração de Colunas e duas heurísticas para o Problema de Roteamento de Veículos Seletivo Min-Max. Com base nos experimentos computacionais realizados, os métodos propostos não se mostraram competitivos em relação àqueles existentes na literatura. No contexto do Problema de Cobertura Multi-Veículo, este trabalho propõe um algoritmo *Branch-and-Price* e uma heurística. Nos experimentos computacionais realizados para avaliar os algoritmos propostos, nove dentre as trinta instâncias consideradas foram resolvidas na otimalidade. Apesar de as técnicas empregadas para os dois problemas terem o mesmo fundamento, nota-se que o método de Geração de Colunas quando aplicado ao segundo problema obteve um desempenho melhor em comparação ao primeiro.

Palavras-chave: Otimização Combinatória, Roteamento de Veículos, *Branch-and-Price*, Heurísticas.

Abstract

In this work, we investigate two Combinatorial Optimization problems: the Min-Max Selective Vehicle Routing Problem and the Multi-vehicle Covering Tour Problem. The former models the design of energy-efficient sensor networks; the latter models the delivery of mobile health-care facilities. We propose a Column Generation algorithm and two heuristics for the Min-Max Selective Vehicle Routing Problem. Computational experiments were performed to assess the quality of the solutions obtained by these heuristics in comparison to those given by the methods found in the literature. The experiments indicate that the methods proposed here for the Min-Max Selective Vehicle Routing Problem are not as efficient and effective as those found in the literature for the majority of the considered instances. In the context of the Multi-vehicle Covering Tour Problem, we propose a Branch-and-Price algorithm and a heuristic. The Branch-and-Price algorithm solved to proven optimality nine out of thirty instances considered in the computational experiments. In conclusion, one can observe that the algorithm proposed for the second problem had superior performance when compared with the algorithm proposed for the first one, although the same set of techniques were applied for solving both problems.

Keywords: Combinatorial Optimization, Vehicle Routing, Branch-and-Price, Heuristics.

Lista de Figuras

2.1 Instância do PRVSMM com 2 veículos e parâmetro **R**. 7
2.2 Porcentagem de variáveis básicas degeneradas ao longo das iteraçãoes para a instância eil51. 35
2.3 Porcentagem de variáveis básicas degeneradas ao longo das iteraçãoes para a instância st70. 37
2.4 Porcentagem de variáveis básicas degeneradas ao longo das iteraçãoes para a instância eil76. 37
2.5 Porcentagem de variáveis básicas degeneradas ao longo das iterações para a instância rat99. 38
2.6 Porcentagem de variáveis básicas degeneradas ao longo das iterações para a instância kroA100. 38
2.7 Porcentagem de variáveis básicas degeneradas ao longo das iterações para a instância eil101. 39

3.1 Porcentagem de variáveis básicas degeneradas ao longo das iterações quando $|V| = 50$ e $|W| = 50$. 53
3.2 Porcentagem de variáveis básicas degeneradas ao longo das iterações quando $|V| = 50$ e $|W| = 100$. 53
3.3 Porcentagem de variáveis básicas degeneradas ao longo das iterações quando $|V| = 100$ e $|W| = 100$. 54
3.4 Porcentagem de variáveis básicas degeneradas ao longo das iterações quando $|V| = 100$ e $|W| = 200$. 54
3.5 Porcentagem de variáveis básicas degeneradas ao longo das iterações quando $|V| = 200$ e $|W| = 200$. 55

Lista de Tabelas

2.1 Resultados computacionais com densidade igual a 0.75. 33
2.2 Resultados computacionais com densidade igual a 10. 34
2.3 Resultados computacionais para as instâncias PRVSMM. 36

3.1 Resultados computacionais com $p = 4$, $q = 200$. 50
3.2 Resultados computacionais com $p = \infty$ e $q = \infty$. 51
3.3 Resultados computacionais para as instâncias PCMV. 52

Lista de Siglas

BC	*Branch-and-Cut*
BP	*Branch-and-Price*
CGH	*Column Generation Heuristic*
CIA	*Cheapest Insertion Algorithm*
GRASP	*Greedy Randomized Adaptive Search Procedure*
GSEC	*Generalized Subtour Elimination Constraints*
HGC	Heurística de Geração de Colunas
HL	Heurística Lagrageana
ILS	*Iterated Local Search*
LRC	Lista Restrita de Candidatos
MS	Método de Subgradiente
MSD	Método de Subgradiente Defletido
PCMERR	Problema de Caminho Mínimo Elementar com Restrições de Recurso
PCMV	Problema de Cobertura Multi-Veículo
PM	Problema Mestre
PML	Problema Mestre Linear
PMLR	Problema Mestre Linear Restrito
PPL	Problema de Programação Linear
PRL	Problema de Relaxação Lagrangeana

PRV Problema de Roteamento de Veículos

PRVSMM Problema de Roteamento de Veículos Seletivo Min-Max

RSSF Redes de Sensores sem Fio

VNS *Variable Neighborhood Search*

Sumário

Agradecimentos vii

Resumo ix

Abstract xi

Lista de Figuras xiii

Lista de Tabelas xv

Lista de Siglas xvii

1 Introdução 1

2 Algoritmo de Geração de Colunas para o Problema de Roteamento de Veículos Seletivo Min-max 5

2.1 Definição do Problema 5
2.2 Revisão Bibliográfica 6
2.3 Formulações Matemáticas para o PRVSMM 10
2.3.1 Formulação GSEC 11
2.3.2 Reformulação por Recobrimento de Conjuntos 12
2.4 Resolvendo o Problema de Precificação 16
2.4.1 Algoritmo Exato 19
2.4.2 Algoritmo Heurístico 20
2.4.3 Paralelização dos Algoritmos 23
2.5 Métodos de Estabilização 24
2.5.1 Estabilização por Pontos Interiores 24
2.5.2 Estabilização por Relaxação Lagrangeana 26
2.6 Heurísticas para o PRVSMM 29
2.6.1 Heurística Lagrangeana 30

2.6.2 Heurística de Geração de Colunas 30
2.7 Experimentos Computacionais 31
2.7.1 Ambiente Computacional e Metodologia 31
2.7.2 Avaliação de Limites Primais 32
2.7.3 Desempenho do Algoritmo de Geração de Colunas 32
2.8 Conclusões 39

3 Algoritmo *Branch-and-Price* para o Problema de Cobertura Multi-Veículo 41
3.1 Definição do Problema 41
3.2 Revisão Bibliográfica 42
3.3 Reformulação por Recobrimento de Conjuntos 43
3.4 Resolvendo o Problema de Precificação 45
3.4.1 Algoritmo Exato 45
3.4.2 Algoritmo Heurístico 47
3.5 Heurística de Geração de Colunas 47
3.6 Algoritmo *Branch-and-Price* 47
3.7 Experimentos Computacionais 48
3.7.1 Ambiente Computacional e Metodologia 48
3.7.2 Resultados Computacionais 49
3.8 Conclusões 54

4 Considerações Finais 57

Referências Bibliográficas 59

Capítulo 1

Introdução

O Problema de Roteamento de Veículos (PRV) foi introduzido em Dantzig & Ramser [1959] e é um problema clássico de Otimização Combinatória. Naquele problema, deve-se determinar um conjunto de rotas, geralmente compartilhando a mesma origem ou depósito, que minimize o custo de deslocamento da frota de veículos.

Devido à sua relevância tanto prática quanto teórica, são encontradas diversas variações do PRV na literatura. Dentre as principais, destacam-se: (i) PRV com restrição de distância [Laporte et al., 1984], em que cada rota não deve exceder um determinado limite de distância; (ii) PRV com janela de tempo [Bräysy & Gendreau, 2005], em que cada cliente deve ser visitado dentro de um intervalo de tempo específico, (iii) PRV com coleta e entrega [Savelsbergh & Sol, 1995], em que cada veículo deve realizar coletas e entregas de mercadorias em pontos específicos, (iv) PRV com *Back-hauls* [Goetschalckx & Jacobs-Blecha, 1989], que pode ser visto como um PRV com coleta e entrega em que todas as entregas devem ser realizadas antes de qualquer coleta, (v) PRV com *satellites* [Bard et al., 1998], em que cada veículo pode visitar pontos específicos, denominados *satellites*, para reabastecer sua carga em vez de retornar ao depósito, e (vi) PRV estocástico [Laporte et al., 2002], em que os dados do problema são estocásticos e seguem uma determinada distribuição de probabilidade.

Neste trabalho, duas variações do PRV são estudadas. As duas possuem natureza seletiva, no sentido de que nem todos os vértices precisam ser visitados, e restrições de cobertura, no sentido de que um vértice não visitado deve estar a uma distância pré-definida de algum vértice que o seja. Essas duas características não são muito comuns dentre as variações do PRV existentes na literatura, embora representem situações muitas vezes verificadas em aplicações.

A primeira variação, denominada Problema de Roteamento de Veículos Seletivo Min-Max (PRVSMM), difere das demais da literatura por, além de possuir restrições

de cobertura e natureza seletiva, ter uma função objetivo que consiste em minimizar o comprimento da maior rota. Sua aplicação está relacionada à eficiência energética no projeto de redes de sensores, as quais têm sido tema de intensa pesquisa nas últimas décadas [Akyildiz et al., 2002].

Diferentemente das inúmeras variações do PRV encontradas na literatura, para as quais são encontrados trabalhos que datam do final da década de 50 [Dantzig & Ramser, 1959], o PRVSMM foi recentemente proposto em Valle et al. [2009]. Com base nos resultados apresentados em [Valle et al., 2009, 2011], a otimalidade de instâncias de tamanho relativamente pequeno pôde ser atestada; para instâncias maiores, há *gap* de dualidade de mais de 40%, mesmo os algoritmos propostos tendo sido executados por aproximadamente quatro horas. Em Valle et al. [2011], foi apontada a dificuldade em resolver instâncias maiores do PRVSMM na otimalidade, como também foi sugerido que outras abordagens fossem investigadas a fim de tornar este problema melhor resolvido na prática. Com base nisso, há uma carência de métodos menos custosos para o PRVSMM.

A segunda variação, denominada Problema de Cobertura Multi-Veículo (PCMV), difere das demais variações encontradas na literatura porque (i) cada vértice de um conjunto específico não pode ser visitado, mas deve ser coberto e (ii) cada vértice de um outro conjunto específico deve ser obrigatoriamente visitado por alguma rota, enquanto os demais vértices podem ou não ser visitados. Sua aplicação está relacionada com a distribuição de unidades móveis de saúde, de modo que uma região não visitada deve estar a uma distância pré-definida de alguma região que a seja. O PCMV foi proposto em [Hachicha et al., 2000], no qual também foram apresentadas três heurísticas para o problema. Com base na revisão bibliográfica realizada, é desconhecida a existência de algoritmo exato para resolver instâncias do PCMV na otimalidade.

Algoritmos de Geração de Colunas e *Branch-and-Price* (BP) têm sido aplicados com sucesso a diversos problemas de otimização, especialmente no contexto do PRV [Barnhart et al., 1998]. Por conta disso, o presente trabalho tem como objetivo investigar se, assim como para outras variações do PRV, algoritmos de Geração de Colunas e BP produzem bons resultados no contexto do PRVSMM e do PCMV.

Para o PRVSMM, as contribuições deste trabalho são: (i) análise da aplicação de técnicas de estabilização no algoritmo de Geração de Colunas, (ii) desenvolvimento de uma heurística lagrangeana e (iii) desenvolvimento de uma heurística baseada no algoritmo de Geração de Colunas. Para o PCMV, as contribuições deste trabalho são: (i) desenvolvimento de um algoritmo de Geração de Colunas e um BP, (ii) desenvolvimento de uma heurística para resolver o problema de precificação, (iii) desenvolvimento de novas estratégias para acelerar a resolução do problema de precificação de forma

exata e (iv) desenvolvimento de uma heurística baseada no algoritmo de Geração de Colunas.

Este trabalho está organizado da seguinte forma. No Capítulo 2, o Problema de Roteamento de Veículos Seletivo Min-Max é tratado. No Capítulo 3, o Problema de Cobertura Multi-Veículo é estudado. Por fim, considerações finais são feitas no Capítulo 4.

Capítulo 2

Algoritmo de Geração de Colunas para o Problema de Roteamento de Veículos Seletivo Min-max

Neste capítulo, o Problema de Roteamento de Veículos Seletivo Min-Max é estudado. Uma reformulação por Recobrimento de Conjuntos, um algoritmo de Geração de Colunas e duas heurísticas são apresentadas para o problema. Por fim, resultados dos experimentos computacionais conduzidos para avaliar o desempenho e a qualidade da soluções dos algoritmos propostos são apresentados.

2.1 Definição do Problema

Seja $\mathcal{K} = \{1, ..., K\}$ um conjunto de veículos idênticos com capacidade de carga considerada ilimitada. Seja $G = (V, E)$ um grafo simples, completo, finito e não-direcionado com conjunto de vértices $V = \{1, ..., n\}$ e conjunto de arestas $E = \{\{i, j\} : i, j \in V\}$, sendo $m = |E|$. O vértice $1 \in V$ denota o *depósito*, o qual é o ponto de origem de todos os veículos, enquanto E denota todos os possíveis movimentos dos veículos entre pares de vértices. Uma matriz de distância $D = (d_{ij})$ entre todo par de vértices de V, obedecendo à propriedade de desigualdade triangular, é definida sobre E; assume-se que $d_{ij} = d_{ji}$ e $d_{ii} = 0$, $\forall i, j \in V$. Por fim, sejam $R \geq 0$ um valor real não negativo e $\omega(i) = \{j \in V : d_{ij} \leq R\}, \forall i \in V$, o conjunto formado pelos vértices que são cobertos pelo vértice i, no sentido de que a distância euclidiana entre o vértice i e aqueles em $\omega(i)$ não excede R.

Com base no que foi exposto, o PRVSMM consiste em encontrar um conjunto

de K rotas (uma para cada veículo) de modo que o comprimento da maior delas seja minimizado. Neste problema, uma rota é um ciclo com origem e destino no depósito, visitando no mínimo dois outros vértices além do depósito. O subgrafo de G induzido por um veículo k é denotado por $H_k = (V_k, E_k)$, no qual V_k denota o conjunto dos vértices visitados e E_k o conjunto das arestas percorridas pelo veículo k. No PRVSMM, nem todos os vértices precisam ser visitados por alguma rota. No entanto, um vértice $i \in V$ que não é visitado deve ser coberto por algum vértice visitado por alguma rota, isto é, para algum $k \in \mathcal{K}$, deve haver um vértice $j \in V_k$ tal que $i \in \omega(j)$. Com exceção do depósito, um vértice só pode ser visitado por no máximo uma única rota. O PRVSMM é $\mathcal{NP}$-Difícil, já que o Problema do Caixeiro Viajante [Dantzig et al., 1954] é um de seus casos particulares quando $K = 1$ e $\omega(i) = \{i\}, \forall i \in V$.

2.2 Revisão Bibliográfica

O PRVSMM foi recentemente proposto em [Valle et al., 2008] como modelo para projetar Redes de Sensores sem Fio (RSSF) eficientes do ponto de vista energético. O modelo proposto tem como objetivo minimizar o tempo de entrega (do inglês, *delay*) das mensagens e o consumo de energia da rede, os quais são fatores críticos no projeto de redes de sensores [Akyildiz et al., 2002]. Para esse fim, alguns nós especiais, denominados *sorvedouros* (do inglês, *sinks*), se deslocam pela região monitorada a fim de coletar os dados sensoriados pelos nós da rede. Esse deslocamento é feito de uma forma tal que, em vez de visitar todos os nós da rede, cada sorvedouro visita um subconjunto de nós. Àquele nó visitado por algum sorvedouro é dado o nome *cluster head*. Cada cluster head engloba um conjunto de nós que dele distam não mais de R. Vale ressaltar que a comunicação direta entre nós sensores é proibida; apenas a comunicação entre sorvedouros e nós sensores pode ocorrer, e esta ocorre quando o cluster head associado é visitado por algum sorvedouro. Uma ilustração da aplicação do PRVSMM no projeto de uma rede de sensores é apresentada a seguir.

A Figura 2.1 ilustra a aplicação do PRVSMM a uma rede de sensores e uma possível solução. No cenário, há dois sorvedouros ($K = 2$) e o parâmetro R é geometricamente indicado pelo raio **R** das circunferências. As duas rotas da solução são indicadas por um ciclo formado por arestas contínuas. Nós visitados por uma rota são representados pelos *Cluster Heads* (vértices em negrito). Por sua vez, nós que não foram visitados estão incluídos dentro de algum círculo tracejado, cujo centro é algum cluster head. Vale notar que cada nó é visitado por no máximo uma rota, com exceção do depósito que é o ponto de partida das K rotas. Deste modo, o depósito possui grau

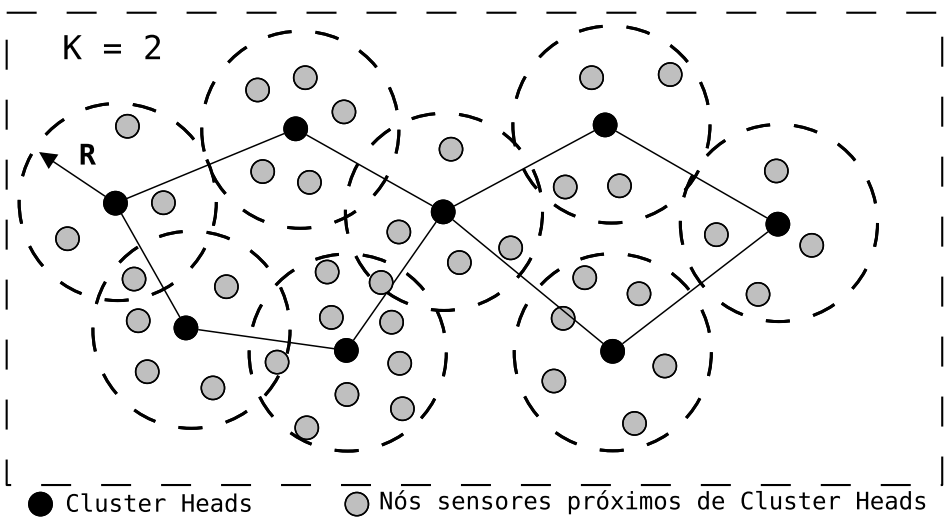


Figura 2.1: Instância do PRVSMM com 2 veículos e parâmetro **R**.

quatro, e os demais nós possuem grau ou zero ou dois na solução. Além disso, cada rota visita no mínimo dois outros vértices além do vértice de origem.

De acordo com os resultados das simulações realizadas em um simulador de redes, apresentados em [Valle et al., 2008], o modelo de organização de RSSF proposto se mostrou superior aos trabalhos comparados em todas as métricas avaliadas. A abordagem de solução proposta naquele trabalho é baseada em uma metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP) [Feo & Resende, 1995], sendo composta por duas fases: a fase de construção da solução inicial e a fase de busca local.

A fase de construção da solução inicial do método proposto é uma adatapção do algoritmo *Cheapest Insertion Algorithm* (CIA) [Julstrom, 1999]. A adaptação em questão consiste basicamente em adicionar à rota o vértice que resulta no menor incremento em termos de distância dentre aqueles disponíveis, levando em consideração o número de vértices que podem ser cobertos caso o vértice em questão venha a ser inserido. Os autores propuseram três versões diferentes do GRASP, as quais aplicam o mesmo método na fase de construção da solução inicial, mas se diferenciam na fase de busca local.

Na primeira versão do GRASP apresentada, a fase de busca local consiste na aplicação de duas buscas locais à solução inicial. A primeira busca local tenta remover nós das rotas; a segunda busca local é uma versão adaptada da busca local 2-OPT [Croes, 1958], sendo aplicada à maior das rotas. Por sua vez, na segunda versão do GRASP, a fase de busca local consiste de uma metaheurística *Iterated Local Search* (ILS) [Lourenço et al., 2002]. No ILS implementado, a fase de perturbação consiste na aplicação da busca local SWAP, que implementa a troca de dois vértices

de rotas distintas. A fase de busca local, por sua vez, consiste na busca local 2-OPT citada anteriormente. Por fim, a terceira versão do GRASP utiliza-se da metaheurística *Variable Neighborhood Search* (VNS) [Mladenovic & Hansen, 1997] na fase de busca local. O VNS é composto pelas buscas locais 2-OPT e SWAP já discutidas.

Em [Valle et al., 2009], foi proposta a primeira formulação matemática para o PRVSMM. No trabalho, uma formulação de Programação Inteira com um número exponencial de restrições do tipo *Generalized Subtour Elimination Constraints* (GSEC) e duas abordagens exatas foram propostas. Uma das abordagens consiste em um algoritmo *Branch-and-Cut* (BC), no qual desigualdades *Blossom* [Edmonds, 1965] e GSEC são separadas. A outra abordagem, por sua vez, consiste em um algoritmo *Local Branching* [Fischetti & Lodi, 2003] que recebe como entrada uma solução viável para o problema e utiliza o BC proposto como resolvedor de cada subproblema. Testes computacionais foram conduzidos em 11 instâncias da biblioteca TSPLIB [TSP, 2012]. Nos experimentos reportados, a otimalidade de poucas instâncias foi atestada, sendo que, para aquelas em que não foi possível obter um certificado de otimalidade, houve *gap* de dualidade de até 40%. Vale ressaltar que, para as instâncias analisadas, o *gap* de dualidade crescia à medida que o número de veículos aumentava, o que termina por ser um gargalo do algoritmo proposto.

Recentemente, Valle et al. [2011] apresentaram novas contribuições e novos resultados para o PRVSMM, além dos métodos propostos em [Valle et al., 2009]. Naquele trabalho, é apresentada uma metaheurística híbrida baseada em GRASP e em ILS, como também um método heurístico denominado *Column Generation Heuristic* (CGH). O GRASP híbrido proposto utiliza o algoritmo CIA na fase construtiva, enquanto a fase de busca local utiliza um ILS. No ILS proposto, o mecanismo de perturbação consiste na remoção (com uma certa probabilidade) de um vértice de uma rota, enquanto que os procedimentos 2-OPT e SWAP são aplicados na fase de busca local, de modo a compor um VNS. A heurística CGH é descrita a seguir.

A heurística CGH é fundamentada na resolução de um Problema de Programação Inteira obtido a partir da reformulação por recobrimento de conjuntos para o PRVSMM. Assuma que $\mathcal{Q}$ denota o conjunto de todas as rotas viáveis para o PRVSMM. A heurística baseia-se na ideia de resolver o problema associado à reformulação com base em um conjunto restrito de colunas $\mathcal{Q}_r \subset \mathcal{Q}$, tal que $|\mathcal{Q}_r| \ll |\mathcal{Q}|$, sob a condição de que uma solução viável para o problema possa ser encontrada pela combinação de K rotas dentre aquelas em $\mathcal{Q}_r$. A heurística é composta por duas etapas que são descritas a seguir.

Na primeira etapa, o Problema de Programação Inteira formulado a partir do conjunto restrito de colunas $\mathcal{Q}_r$ é resolvido na otimalidade. Na segunda etapa, o ILS e

o GRASP são utilizados para gerar novas rotas a partir da solução obtida na primeira etapa. As novas rotas passam a compor um novo conjunto $\mathcal{Q}_r$, com base no qual um novo Problema de Programação Inteira é formulado e resolvido na otimalidade. Todo o processo é repetido enquanto a resolução do problema aprimorar a melhor solução conhecida, que é constantemente atualizada ao longo do algoritmo. Apesar das novas contribuições, os autores apontaram a dificuldade de resolver o PRVSMM com base nos experimentos conduzidos.

Valle et al. [2011] reportaram experimentos com um conjunto de vinte instâncias com o objetivo de avaliar as abordagens exatas e heurísticas propostas para o PRVSMM. Para o tratamento exato, o BC, dentro do limite de 4 horas de execução, foi capaz de atestar a otimalidade de três instâncias, enquanto que, para aquelas em que não foi possível fazê-lo, os *gaps* de dualidade obtidos variaram entre 0.7% e 47.3% para os cenários avaliados. Por sua vez, o *Local Branching* foi capaz de melhorar a solução inicial de entrada na maioria dos casos, principalmente à medida que o número de veículos aumentava. Para o tratamento heurístico, a heurística CGH não apenas forneceu soluções melhores que os limites superiores fornecidos pelo BC (quando este não foi capaz de atestar a otimalidade) e pelo *Local Branching*, como também obteve tais soluções em menos tempo. A saber, a heurística CGH demorou aproximadamente um décimo do tempo necessário pelas outras duas abordagens. Em conclusão, os autores apontaram a dificuldade de resolver instâncias do PRVSMM na otimalidade, permanecendo diversas instâncias sem seu valor ótimo conhecido. Não foi apresentado estudo algum do comportamento dos algoritmos propostos à medida que a densidade da rede varia.

Um problema relacionado ao PRVSMM, intitulado *Min-max K-Vehicles Windy Rural Postman Problem* (MM K-WRPP), foi proposto em Benavent et al. [2009]. Seja um grafo $G = (V, E)$ com custos não-negativos c_{ij} e c_{ji} atribuídos para cada aresta $\{i, j\} \in E$. Dados um vértice como depósito, um subconjunto $E_r \subseteq E$ de arestas e um número K de veículos com capacidade considerada ilimitada, no MM K-WRPP, busca-se um conjunto de K rotas (uma para cada veículo), de modo que cada rota inicie e termine no depósito e cada aresta em E_r seja percorrida por um único veículo. O problema tem como objetivo minimizar o comprimento da maior rota a fim de encontrar um conjunto de rotas balanceadas em termos de comprimento. Naquele trabalho, os autores propuseram uma formulação de Programação Inteira com um número exponencial de restrições e um algoritmo BC. O PRVSMM difere do MM K-WRPP por dois aspectos: há restrições de cobertura e não há restrição que determinadas arestas devam ser visitadas.

Assim como o PRVSMM, uma variação do PRV que também possui natureza

min-max foi proposta em [Glaab, 2002]. No problema, intitulado *Laser Multi-Scanner Problem* (*LMSP*), busca-se definir um conjunto de m rotas, uma para cada laser, de modo a minimizar o comprimento da maior rota. Diferentemente do PRVSMM, naquele problema, todos os vértices devem ser visitados por algum veículo (laser) e cada veículo pode ter um ponto de partida diferente. Os autores propuseram duas heurísticas: uma heurística de inserção e uma heurística gulosa de arcos. A fim de avaliar a qualidade das heurísticas propostas, os autores propuseram duas relaxações: uma relaxação combinatória denominada $m - forest$ e outra que relaxa as restrições do número de rotas e da inexistência de subciclo.

Um outro problema com natureza min-max foi proposto em [Yakici & Karasakal, 2012]. No problema, intitulado *Vehicle Routing Problem with Split Delivery and Heterogeneous Demand* (VRPSDHD), deve-se definir um conjunto de K rotas, uma para cada veículo, de modo que cada consumidor tenha sua demanda atendida. O problema tem como objetivo minimizar o maior tempo gasto pelas rotas. O tempo gasto por uma rota consiste na soma entre o tempo dispendido com o deslocamento do veículo associado e aquele dispendido no atendimento de cada cliente visitado. Vale ressaltar que os veículos podem possuir velocidade de deslocamento diferente e o custo de atendimento de um cliente por veículos distintos pode variar. Naquele trabalho, os autores propuseram uma formulação de Programação Inteira com um número exponencial de restrições e uma heurística. O PRVSMM difere do VRPSDHD por dois aspectos principais: possui restrições de cobetura e natureza seletiva.

Por fim, Baldacci et al. [2007] propuseram um problema que, assim como o PRVSMM, surgiu como modelo para projetar redes de comunicação. O *Capacitated m-Ring-Star Problem* (CmRSP) consiste em definir um conjunto de m ciclos disjuntos com origem no depósito e um conjunto de atribuições entre cada vértice não visitado e aquele visitado por algum ciclo. O número de nós visitados por um ciclo ou a ele atribuídos não deve exceder um valor pré-estabelecido. Naquele problema, deseja-se minimizar o custo de construção dos ciclos e o custo de conexão dos vértices não visitados. O PRVSMM difere do CmRSP por três aspectos: possui natureza min-max, não há custo para atribuir (cobrir) vértices não visitados a alguma rota e, por fim, não há restrições de capacidade.

2.3 Formulações Matemáticas para o PRVSMM

Esta seção apresenta duas formulações de Programação Inteira para o PRVSMM. A primeira delas envolve um número exponencial de restrições e foi proposta em [Valle

et al., 2011]. Por sua vez, a segunda formulação, proposta neste trabalho, é uma reformulação do PRVSMM por Recobrimento de Conjuntos que possui um número exponencial de variáveis. No restante desta seção, dada uma formulação P para o PRVSMM, assuma que $z(P)$ denote o limite de Programação Linear fornecido por P. Considere que $\mathbb{B} = \{0, 1\}$, $\mathbb{Z}$ denote o conjunto dos números inteiros e, por fim, $\mathbb{R}$ denote o conjunto dos números reais.

2.3.1 Formulação GSEC

Dado um subconjunto $W \subseteq V$ de vértices, $E[W, V \setminus W] = \{\{i, j\} \in E : i \in W, j \in V \setminus W\}$ representa o conjunto de arestas no corte $[W, V \setminus W]$, e $E[W] = \{\{i, j\} \in E : i, j \in W\}$ representa o conjunto de arestas com ambas as extremidades em W. Além disso, $\delta(i) = E[\{i\}, V \setminus \{i\}]$ representa o conjunto de arestas incidentes ao vértice $i \in V$. As variáveis de decisão a seguir são utilizadas na formulação: (i) $x_{ij}^k \in \mathbb{B}, \forall \{i, j\} \in E, \forall k \in \mathcal{K}$, assumindo valor 1 se a aresta $\{i, j\}$ pertencer à rota percorrida pelo veículo k ou valor 0 caso contrário, (ii) $y_i^k \in \mathbb{B}, \forall i \in V, \forall k \in \mathcal{K}$, assumindo valor 1 se o vértice i for visitado pela rota percorrida pelo veículo k ou valor 0 caso contrário, e (iii) $w \in \mathbb{R}$, denotando o comprimento da maior rota. Com base nisso, a primeira formulação matemática para o PRVSMM, proposta em [Valle et al., 2011], é dada por:

$$\min \quad \{w : (w, x, y) \in P_{GSEC} \cap (\mathbb{R}, \mathbb{B}^{K|E|}, \mathbb{B}^{K|V|})\} \tag{2.1}$$

sendo o poliedro P_{GSEC} definido como:

$$w \geq \sum_{\{i,j\} \in E} d_{ij} x_{ij}^k, \forall k \in \mathcal{K} \tag{2.2}$$

$$\sum_{\{i,j\} \in \delta(i)} x_{ij}^k = 2y_i^k, \ \ \forall i \in V, \ \ \forall k \in \mathcal{K}, \tag{2.3}$$

$$\sum_{k \in \mathcal{K}} \sum_{j \in \omega(i)} y_j^k \geq 1, \ \ \forall i \in V, \tag{2.4}$$

$$-\sum_{k \in \mathcal{K}} y_i^k \geq -1, \ \ \forall i \in V \setminus \{1\}, \tag{2.5}$$

$$\sum_{\{i,j\} \in E[W, V \setminus W]} x_{ij}^k \geq 2y_z^k, \forall W \subset V, 1 \in W, z \notin W, \forall k \in \mathcal{K}, \tag{2.6}$$

$$y_1^k = 1, \ \ \forall k \in \mathcal{K}. \tag{2.7}$$

As restrições (2.2) fazem com que w assuma um valor maior ou igual ao com-

primento de cada rota. As restrições (2.3) garantem que se o vértice i for visitado pelo veículo k, então exatamente duas das arestas utilizadas por este veículo devem incidir sobre i. As restrições (2.4) referem-se ao requisito de cobertura, garantindo que todo vértice é coberto. Por sua vez, as restrições (2.5) asseguram que um vértice será visitado por no máximo um veículo. As restrições (2.6) referem-se às restrições de eliminação de subciclo GSEC, prevenindo o modelo de formar rotas que não visitam o depósito. Por fim, as restrições (2.7) obrigam que o depósito seja sempre visitado para cada veículo.

2.3.2 Reformulação por Recobrimento de Conjuntos

A formulação proposta neste trabalho é uma reformulação do PRVSMM por Recobrimento de Conjuntos. Esta reformulação possui um número exponencial de variáveis. Portanto, faz-se necessário o uso do método de Geração de Colunas para obter limites inferiores válidos para a reformulação proposta. Antes de apresentar a reformulação, fazem-se necessárias algumas definições e notação.

Seja $\mathcal{P}$ o conjunto de todas as rotas viáveis para o PRVSMM sobre um grafo $G = (V, E)$, de modo que $\mathcal{P}_k$ representa o conjunto das rotas em $\mathcal{P}$ associadas ao veículo k. Para cada rota $p \in \mathcal{P}$, os seguintes parâmetros são definidos: (i) d^p representa o comprimento da rota, (ii) β_i^p assume valor 1 se o vértice i for visitado pela rota ou valor 0 caso contrário, (iii) δ_i^p assume valor 1 se o vértice i for coberto pela rota ou valor 0 caso contrário, e (iv) ς_{ij}^p assume valor 1 se a aresta $\{i, j\} \in E$ for utilizada pela rota ou valor 0 caso contrário. Por fim, seja $\lambda_k^p \in \mathbb{B}$ uma variável que assume valor 1 se a rota $p \in \mathcal{P}_k$ for utilizada pelo veículo k ou valor 0 caso contrário. Deste modo, o Problema Mestre (PM) é definido como:

$$\min \quad \{w : (w, \lambda) \in P_c \cap (\mathbb{R}_+, \mathbb{B}^{K|\mathcal{P}|})\}, \tag{2.8}$$

sendo o poliedro P_c definido como:

$$w \geq \sum_{p \in \mathcal{P}_k} d^p \lambda_k^p, \forall k \in \mathcal{K}, \tag{2.9}$$

$$\sum_{p \in \mathcal{P}_k} \lambda_k^p = 1, \forall k \in \mathcal{K}, \tag{2.10}$$

$$\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k} \delta_i^p \lambda_k^p \geq 1, \forall i \in V \setminus \{1\}, \tag{2.11}$$

$$-\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k} \beta_i^p \lambda_k^p \geq -1, \forall i \in V \setminus \{1\}. \tag{2.12}$$

As restrições (2.9) forçam w a assumir um valor maior ou igual ao comprimento das rotas atribuídas. As restrições (2.10) obrigam que exatamente uma única rota seja atribuída para cada veículo. As restrições (2.11) garantem que todo vértice deve ser coberto. Por fim, as restrições (2.12) garantem que cada vértice só pode ser visitado por no máximo um veículo.

Define-se como Problema Mestre Linear (PML) o problema $\{\min w : (w, \lambda) \in P_c \cap (\mathbb{R}_+, \mathbb{R}_+^{K|\mathcal{P}|})\}$. Associando-se variáveis duais γ_k^0, τ_k, γ_i^1 e γ_i^2 respectivamente às restrições (2.9)-(2.12) do PML, obtém-se o Problema Dual do Mestre Linear:

$$\max \quad \sum_{k \in K} \tau_k - \sum_{i \in V \setminus \{1\}} \gamma_i^2 + \sum_{i \in V \setminus \{1\}} \gamma_i^1, \tag{2.13}$$

Sujeito a:

$$-\sum_{k \in K} \gamma_k^0 \geq -1, \tag{2.14}$$

$$d^p \gamma_k^0 - \sum_{i \in V \setminus \{1\}} \delta_i^p \gamma_i^1 + \sum_{i \in V \setminus \{1\}} \beta_i^p \gamma_i^2 - \tau_k \geq 0, \forall k \in K, p \in \mathcal{P}_k, \tag{2.15}$$

$$\gamma_k^0 \geq 0, \gamma_i^1 \geq 0, \gamma_i^2 \geq 0. \tag{2.16}$$

Devido ao grande número de variáveis no PML, faz-se necessário um procedimento de Geração de Colunas. Para este fim, formula-se um Problema Mestre Linear Restrito (PMLR) em que $\mathcal{P}$ é substituído por um conjunto $\tilde{\mathcal{P}} \subset \mathcal{P}$ ($|\tilde{\mathcal{P}}| \ll |\mathcal{P}|$) no PML. Por fim, considere $\tilde{\mathcal{P}}_k$ como o conjunto restrito das rotas em $\tilde{\mathcal{P}}$ associadas ao veículo k.

Sejam $\bar{\lambda}$ uma solução básica para o PMLR e $\bar{\gamma}_k^0$, $\bar{\tau}_k$, $\bar{\gamma}_i{}^1$ e $\bar{\gamma}_i^2$ as respectivas variáveis duais ótimas. A fim de fornecer um certificado de que $\bar{\lambda}$ resolve o PML, é necessário garantir que as restrições (2.15) sejam satisfeitas. Com o objetivo de fornecer esse certificado ou de encontrar uma restrição dual violada, deve-se resolver um problema de precificação para cada veículo $k \in \mathcal{K}$, definido como:

$$p_k^* = \arg\min_{p \in \mathcal{P}_k} \quad \phi(p), \tag{2.17}$$

em que $\phi(p) = d^p \bar{\gamma_k}{}^0 - \sum_{i \in V \setminus \{1\}} \delta_i^p \bar{\gamma_i}{}^1 + \sum_{i \in V \setminus \{1\}} \beta_i^p \bar{\gamma_i^2} - \bar{\tau}_k$ representa o custo reduzido de um rota p. Se $\phi(p_k^*) \geq 0, \forall k \in \mathcal{K}$, a solução básica $\bar{\lambda}$ resolve o PML. Caso contrário, foi encontrada alguma rota p_k^*, para algum k, à qual associa-se uma desigualdade (2.15) violada e, portanto, essa rota deve ser inserida no PMLR, o qual será resolvido novamente.

O problema de precificação em questão corresponde ao Problema de Caminho

Mínimo Elementar com Restrições de Recurso (PCMERR) (do inglês, *Elementary Shortest Path Problem with Resource Constraints*) e é decomposto em K subproblemas, um para cada veículo. Este problema será discutido detalhadamente na próxima seção. Antes disso, cabe apresentar uma prova de que os limites de Programação Linear fornecidos pela Reformulação por Recobrimento de Conjuntos são tão fortes quanto aqueles fornecidos pela Formulação GSEC.

Proposição 1. $z(P_c) \geq z(P_{GSEC})$.

Demonstração. Seja um ponto qualquer $(\bar{w}, \bar{\lambda}) \in P_c$, em que $\bar{\lambda} = (\bar{\lambda}_1, ..., \bar{\lambda}_K)$. Deve-se demonstrar que esse ponto possui um correspondente $(\bar{w}, \bar{x}, \bar{y}) \in P_{GSEC}$, em que $\bar{x} = (\bar{x}^1, ..., \bar{x}^K)$ e $\bar{y} = (\bar{y}^1, ..., \bar{y}^K)$. Para este fim, para cada $k \in \mathcal{K}$, considere a definição das variáveis $\bar{x}_{ij}^k$ e $\bar{y}_i^k$ abaixo.

$$\bar{x}_{ij}^k = \sum_{p \in \mathcal{P}_k} \varsigma_{ij}^p \bar{\lambda}_k^p, \forall k \in \mathcal{K}, i, j \in V, \tag{2.18}$$

$$\bar{y}_i^k = \sum_{p \in \mathcal{P}_k} \beta_i^p \bar{\lambda}_k^p, \forall k \in \mathcal{K}, i \in V. \tag{2.19}$$

Como (i) $\beta_i^p \in \mathbb{B}, \forall i \in V, p \in \mathcal{P}$, (ii) $\varsigma_{ij}^p \in \mathbb{B}, \forall \{i, j\} \in E, p \in \mathcal{P}$ e (iii) com base na restrição (2.10), tem-se que $\bar{x}^k \in \mathbb{B}^{|E|}$ e $\bar{y}^k \in \mathbb{B}^{|V|}$ $\forall k \in \mathcal{K}$.

Considere que a restrição (2.9) seja reescrita equivalentemente da seguinte forma:

$$\bar{w} \geq \sum_{\{i,j\} \in E} d_{ij} \sum_{p \in \mathcal{P}_k} \varsigma_{ij}^p \bar{\lambda}_k^p, \forall k \in \mathcal{K}, \tag{2.20}$$

Pela substituição de (2.18) em (2.20), a restrição (2.2) é claramente satisfeita.

Somando-se a equação (2.19) para todo $k \in \mathcal{K}$, obtém-se:

$$\sum_{k \in \mathcal{K}} \bar{y}_i^k = \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k} \beta_i^p \bar{\lambda}_k^p, i \in V. \tag{2.21}$$

Como $\bar{\lambda}$ satisfaz à desigualdade (2.12), chega-se a seguinte desigualdade:

$$\sum_{k \in \mathcal{K}} \bar{y}_i^k \leq 1, i \in V. \tag{2.22}$$

Deste modo, a restrição (2.5) é satisfeita.

Como o vértice $i = 1$ é sempre visitado ($\beta_1^p = 1$) por toda rota $p \in \mathcal{P}_k$, tem-se

que

$$\bar{y}_1^k = \sum_{p \in \mathcal{P}_k} \bar{\lambda}_k^p, \forall k \in \mathcal{K}. \tag{2.23}$$

Como $\bar{\lambda}$ satisfaz à restrição (2.10), conclui-se que

$$\bar{y}_1^k = 1, \forall k \in \mathcal{K}. \tag{2.24}$$

Deste modo, a restrição (2.7) é satisfeita.

Considere que a restrição (2.12) seja reescrita equivalentemente da seguinte forma:

$$\sum_{k \in \mathcal{K}} \sum_{j \in \omega(i)} \sum_{p \in \mathcal{P}_k} \beta_j^p \lambda_k^p \geq 1, \forall i \in V \setminus \{1\}. \tag{2.25}$$

Como $\bar{\lambda}$ satisfaz à restrição (2.12), consequentemente satisfaz à desigualdade (2.25), considere a substituição de (2.19) em (2.25)

$$\sum_{k \in \mathcal{K}} \sum_{j \in \omega(i)} \bar{y}_j^k \geq 1, \forall i \in V \setminus \{1\}. \tag{2.26}$$

Deste modo, a restrição (2.4) é satisfeita, exceto quando $i = 1$.

Para $i = 1$, uma vez que $i \in \omega(i)$, considere que a restrição (2.4) seja reescrita da seguinte forma

$$\sum_{k \in \mathcal{K}} y_1^k + \sum_{k \in \mathcal{K}} \sum_{j \in \omega(1) \setminus 1} y_j^k \geq 1. \tag{2.27}$$

Substituindo a equação (2.24) em (2.27), chega-se a:

$$K + \sum_{j \in \omega(1) \setminus 1} \bar{y}_j^k \geq 1. \tag{2.28}$$

Deste modo, a restrição (2.4) é satisfeita quando $i = 1$.

Por definição, uma rota $p \in \mathcal{P}_k$ é um ciclo em que cada vértice é visitado uma única vez e há exatamente duas arestas incidentes a um vértice i qualquer visitado por p. Com base nisso,

$$2\bar{y}_i^k = \sum_{\{i,j\} \in \delta(i)} \sum_{p \in \mathcal{P}_k} \varsigma_{ij}^p \bar{\lambda}_k^p, \forall i \in V, k \in \mathcal{K}. \tag{2.29}$$

Substituindo (2.18) em (2.29), obtém-se

$$2\bar{y}_i^k = \sum_{\{i,j\}\in\delta(i)} \bar{x}_{ij}^k, \forall i \in V, k \in \mathcal{K}. \tag{2.30}$$

Portanto, a restrição (2.3) é satisfeita.

Por definição, uma rota $p \in \mathcal{P}_k$ é um ciclo com origem no vértice 1 em que cada vértice é visitado uma única vez e há pelo menos duas arestas em um corte $[W, V \setminus W]$ qualquer tal que $W \subset V, 1 \in W$. Com base nisso,

$$\sum_{\{i,j\}\in E[W,V\setminus W]} \varsigma_{ij}^p \bar{\lambda}_k^p \geq 2\beta_z^p \bar{\lambda}_k^p, \forall W \subset V, 1 \in W, z \notin W, \forall p \in \mathcal{P}_k, k \in \mathcal{K}. \tag{2.31}$$

Somando-se a desigualdade (2.31) para cada $p \in \mathcal{P}_k$, obtém-se

$$\sum_{\{i,j\}\in E[W,V\setminus W]} \sum_{p\in\mathcal{P}_k} \varsigma_{ij}^p \bar{\lambda}_k^p \geq 2 \sum_{p\in\mathcal{P}_k} \beta_z^p \bar{\lambda}_k^p, \forall W \subset V, 1 \in W, z \notin W, k \in \mathcal{K}. \tag{2.32}$$

Substituindo (2.18) e (2.19) em (2.32), chega-se a

$$\sum_{\{i,j\}\in E[W,V\setminus W]} \bar{x}_{ij}^k \geq 2\bar{y}_z^k, \forall W \subset V, 1 \in W, z \notin W, k \in \mathcal{K}. \tag{2.33}$$

Portanto, a restrição (2.6) é satisfeita. □

2.4 Resolvendo o Problema de Precificação

O PCMERR consiste em encontrar um caminho mínimo elementar (em que não há repetição de vértices) entre um vértice de origem e outro de destino, de modo que os recursos consumidos ao percorrer o caminho não excedam suas disponibilidades. Recursos podem estar relacionados a fatores de tempo (intervalo de tempo em que um vértice pode ser visitado ou tempo de duração da rota), de capacidade (capacidade máxima que um veículo pode transportar), entre outros. Ao percorrer uma aresta ou visitar um vértice, uma quantidade pré-estabelecida de cada recurso é consumida. O PCMERR é $\mathcal{NP}$-Difícil se o grafo contiver ciclos de custo negativo [Dror, 1994].

A notação apresentada em [Boland et al., 2006] é utilizada para definir o PCMERR. Seja $D = (V, A)$ um grafo direcionado com conjunto V de vértices ($n = |V|$) e conjunto A de arcos ($m = |A|$). Sejam $R \in \mathbb{Z}_+^*$ o número de recursos considerados e uma tupla $\Psi = \langle \Psi^1, ..., \Psi^R \rangle$ de recursos tal que $\Psi^r \in \mathbb{Z}_+$ denota a capacidade do recurso r. Para cada arco $(i, j) \in A$, são definidos um custo $c_{ij} \in \mathbb{R}$ de utilização e uma tupla

$\psi_{ij} = \langle \psi_{ij}^1, ..., \psi_{ij}^R \rangle$ que denota a quantidade a ser consumida para cada recurso. Para um caminho p, o conjunto de arcos utilizados é designado por $A(p)$, seu custo é representado por $\phi(p)$ e o consumo de recursos é denotado por $\psi(p) = \langle \psi^1(p), ..., \psi^R(p) \rangle$, no qual $\psi^r(p) = \sum_{\{i,j\} \in A(p)} \psi_{ij}^r$ denota o consumo do recurso r. O PCMERR consiste em determinar um caminho elementar p de custo mínimo que satisfaça as restrições de capacidade $\psi^r(p) \leq \Psi^r$ para cada recurso r.

A fim de garantir a elementaridade dos caminhos, é considerada uma tupla $\Upsilon = \langle \Upsilon^1, ..., \Upsilon^n \rangle$ de recursos tal que Υ^i possui capacidade unitária para o vértice i. Para um caminho p, o consumo dos recursos unitários é denotado por $\upsilon(p) = \langle \upsilon^1(p), ..., \upsilon^n(p) \rangle$, no qual $\upsilon^i(p)$ indica se o vértice i é visitado. Quando um vértice i qualquer é visitado, sua capacidade unitária Υ^i é consumida ($\upsilon^i(p) = 1$), impedindo, assim, que este vértice seja visitado novamente.

A abordagem comumente utilizada para resolver o PCMERR é fundamentada em Programação Dinâmica. De um modo geral, a abordagem consiste em estender caminhos parciais por meio da inserção de cada vértice não visitado e adjacente àquele último visitado pelo caminho, até que nenhum novo caminho possa ser criado. Como resultado, um novo caminho é criado a cada extensão. O algoritmo usualmente tem início no vértice de origem, e caminhos são estendidos em direção ao vértice de destino. Associa-se, ainda, um rótulo $l_p = \langle \phi(p), \upsilon(p), \psi(p) \rangle$ (do inglês, *label*) a cada caminho parcial p com o objetivo de indicar seu custo e o consumo de recursos. Quando nenhum caminho parcial puder ser estendido, o caminho ótimo é aquele de menor custo terminando no vértice de destino.

Feillet et al. [2004] propuseram o primeiro algoritmo de Programação Dinâmica para o PCMERR quando o grafo contém ciclos de custo negativo. Naquele trabalho, os autores estenderam o uso da tupla Υ a fim de identificar vértices que não podem ser alcançados por limitações de recurso. A saber, consome-se o recurso Υ^i associado ao vértice i caso este possa ser visitado, mas sua visitação viole a capacidade disponível de algum recurso. Como resultado, segundo os autores, há uma redução no tempo de execução do algoritmo, já que estados são podados. Por fim, tendo em vista que o desempenho do algoritmo proposto é intrinsecamente dependente da quantidade de rótulos gerados, os autores introduziram o conceito de regras de dominância para impedir a geração de novos rótulos a partir da extensão daqueles que garantidamente levem a caminhos sub-ótimos. Sejam $\hat{p}_i$ e $\tilde{p}_i$ dois caminhos distintos que terminam no vértice i. O caminho $\hat{p}_i$ domina $\tilde{p}_i$ se: (i) $\phi(\hat{p}_i) \leq \phi(\tilde{p}_i)$, (ii) $\psi^r(\hat{p}_i) \leq \psi^r(\tilde{p}_i)$ para $r = 1, ..., R$, (iii) $\upsilon^j(\hat{p}_i) \leq \upsilon^j(\tilde{p}_i)\ \forall j \in V$ e (iv) pelo menos uma das desigualdades for estritamente menor.

Outro algoritmo de Programação Dinâmica para o PCMERR foi proposto

em [Boland et al., 2006]. Naquele algoritmo, vértices de um determinado subconjunto possuem recurso com capacidade unitária enquanto os demais possuem recurso com capacidade ilimitada em relação à tupla Υ. Como resultado, o algoritmo proposto resolve uma relaxação do PCMERR, já que a solução ótima retornada pode conter ciclos. Em caso de a solução retornada possuir repetição de alguns vértices, o algoritmo é executado novamente considerando um recurso unitário para os vértices repetidos. Esse processo é repetido até que um caminho elementar seja encontrado. Adicionalmente, é proposta uma etapa de pré-processamento do grafo a fim de remover vértices ou arcos cujos consumos excedem os recursos disponíveis.

Outra abordagem para o PCMERR foi proposta por Righini & Salani [2006]. Naquela abordagem, um algoritmo de Programação Dinâmica baseado em rótulos realiza uma busca bidirecional no espaço de estados em conjunto com mecanismos de poda. Diferentemente das duas abordagens anteriores, neste algoritmo, caminhos são estendidos tanto a partir do vértice de origem em direção ao vértice de destino quanto no sentido inverso, resultando em uma busca de natureza bidirecional. Em cada vértice, são armazenados dois conjuntos disjuntos de rótulos: um que armazena os rótulos originados no vértice de origem e outro que armazena os rótulos originados no vértice de destino. Um caminho da origem ao destino é formado quando a união de dois caminhos parciais de direções contrárias que terminem em um mesmo vértice possa ocorrer sem violar as capacidades dos recursos disponíveis. Por fim, os autores apresentaram mecanismos de poda para eliminar rótulos que levem a caminhos sub-ótimos. Os mecanismos de poda propostos naquele trabalho são discutidos a seguir.

O primeiro mecanismo de poda apresentado utiliza-se de um limite superior para o ganho máximo que um caminho parcial pode obter (para um dado recurso) até chegar ao vértice de destino. Um caminho é podado caso a subtração entre seu custo corrente e o limite superior seja maior ou igual ao custo de uma solução viável previamente conhecida. Vale ressaltar que o limite superior supracitado é igual ao valor da relaxação linear de um problema da mochila binária formulado tendo como lucro os prêmios a serem coletados em cada vértice e como restrição da mochila a capacidade do recurso considerado. O segundo mecanismo de poda calcula um limite superior para o número de arcos que podem ser visitados por um caminho parcial sem violar a capacidade de algum recurso. Neste caso, diferentemente da primeira regra de poda, um problema da mochila binária é resolvido na otimalidade. Um terceiro mecanismo de poda consiste em interromper a extensão de um caminho caso a metade da capacidade disponível de um recurso considerado crítico seja consumida. Este último mecanismo requer que haja um consumo positivo do recurso crítico em cada aresta do grafo. Vale ressaltar que os próprios autores afirmaram que, sem os mecanismos de poda, o algoritmo proposto

produziria duas vezes a quantidade de rótulos produzidos por uma estratégia monodirecional.

Os algoritmos implementados no contexto do presente trabalho para resolver o problema de precificação são discutidos na próxima seção. A saber, foram implementados um algoritmo exato e um algoritmo heurístico.

2.4.1 Algoritmo Exato

O algoritmo exato utilizado neste trabalho para resolver o problema de precificação é aquele apresentado em [Feillet et al., 2004]. O algoritmo implementado teve de ser adaptado para levar em consideração as restrições de cobertura (2.12) do PRVSMM. A adaptação consistiu na introdução de uma tupla $\Pi = \langle \Pi^1, ..., \Pi^n \rangle$ de recursos tal que Π^i possui capacidade unitária e indica se o vértice $i \in V$ é coberto. Para um caminho p, $\pi(p)$ representa o consumo dos recursos associados a Π e $\pi^i(p)$ indica se o vértice i é coberto. Com base nisso, no algoritmo implementado, um rótulo para um caminho parcial p é determinado pela tupla $\langle d^p, \phi(p), \upsilon(p), \pi(p) \rangle$.

Como apontado anteriormente, o desempenho do algoritmo implementado depende da quantidade de caminhos parciais estendidos. Deste modo, regras de dominância e de poda foram desenvolvidas com o objetivo de antecipadamente caracterizar caminhos cuja extensão leve a um caminho sub-ótimo.

Proposição 2. *Sejam* $l_p = \langle d^p, \phi(p), \upsilon(p), \pi(p) \rangle$ *e* $l_{\tilde{p}} = \langle d^{\tilde{p}}, \phi(\tilde{p}), \upsilon(\tilde{p}), \pi(\tilde{p}) \rangle$ *rótulos associados respectivamente aos caminhos* p *e* $\tilde{p}$ *que terminam no vértice* i*. Se* $\pi^j(p) \leq \pi^j(\tilde{p})$*,* $\forall j \in V$*, e* $\phi(p) \leq \phi(\tilde{p})$*, então* l_p *domina* $l_{\tilde{p}}$*.*

Demonstração. Sejam j um vértice qualquer para o qual será considerada a extensão dos rótulos l_p e $l_{\tilde{p}}$, e c_{ij} o custo da aresta entre os vértices i e j. Assuma que $\hat{\omega}(u, q) = \{a \in \omega(u) \setminus \{1\} | \pi^a(q) = 0\}$ denota o conjunto de todos os vértices cobertos pelo vértice u que não são cobertos por uma rota q. Uma vez que $\pi^u(p) \leq \pi^u(\tilde{p})$, $\forall u \in V$, tem-se $\hat{\omega}(j, \tilde{p}) \subseteq \hat{\omega}(j, p)$. As possíveis extensões válidas para $\tilde{p}$ são analisadas a seguir.

Caso 1: Considere que $\upsilon^j(p) = 0$ e $\upsilon^j(\tilde{p}) = 0$, isto é, o vértice j não foi visitado por p nem por $\tilde{p}$. Como resultado, tanto a extensão de l_p quanto a de $l_{\tilde{p}}$ ao vértice j é viável. Isto posto, a extensão dos rótulos l_p e $l_{\tilde{p}}$ ao vértice j em termos de custo resulta em $\phi(p) + c_{ij}\bar{\gamma}^0_k - \sum_{u \in \hat{\omega}(j,p)} \bar{\gamma}^1_u + \bar{\gamma}^2_j \leq \phi(\tilde{p}) + c_{ij}\bar{\gamma}^0_k - \sum_{u \in \hat{\omega}(j,\tilde{p})} \bar{\gamma}_u{}^1 + \bar{\gamma}^2_j$, uma vez que $\hat{\omega}(j, \tilde{p}) \subseteq \omega(j, p)$.

Caso 2: Considere que $\upsilon^j(p) = 1$ e $\upsilon^j(\tilde{p}) = 0$, isto é, o vértice j foi visitado apenas por p. Portanto, a extensão de $l_{\tilde{p}}$ ao vértice j é viável. Isto posto, a extensão

do rótulo $l_{\tilde{p}}$ ao vértice j em termos de custo resulta em $\phi(\tilde{p}) + c_{ij}\bar{\gamma}_k^0 + \bar{\gamma}_j^2 \geq \phi(p)$, já que $\hat{\omega}(j, \tilde{p}) = \emptyset$ porque $\upsilon^j(p) = 1$ e $\pi^u(p) \leq \pi^u(\tilde{p})$, $\forall u \in V$.

Portanto, o caminho $\tilde{p}$ pode ser eliminado, já que sua extensão resulta em um caminho com custo sempre maior ou igual a $\phi(p)$. □

Proposição 3. *Seja j um vértice qualquer e considere $\omega(j)^+ = \{v \in \omega(j) : \bar{\gamma}_v^1 > 0\}$. Seja $l_p = \langle d^p, \phi(p), \upsilon(p), \pi(p) \rangle$ um rótulo associado ao caminho p que termina no vértice i tal que $\upsilon^j(p) = 0$ e $\pi^v(p) = 1$, $\forall v \in \omega(j)^+$. A extensão de l_p para o vértice j pode ser podada.*

Demonstração. Suponha que $\tilde{p}$ seja um caminho tal que p seja seu subcaminho e as arestas $\{i, j\}$ e $\{j, l\}$ sejam utilizadas, sendo l o vértice visitado imediatamente após o vértice j. Seja $\hat{p}$ o caminho obtido a partir de $\tilde{p}$ pela remoção do vértice j, de modo a remover as arestas $\{i, j\}$ e $\{j, l\}$, e reconectar o caminho por meio da aresta $\{i, l\}$. Com base na propriedade da desigualdade triangular e no fato de $\pi^v(p) = 1$, $\forall v \in \omega(j)^+$, temos que $\phi(\hat{p}) = \phi(\tilde{p}) - c_{ij} - c_{jl} + c_{jl} \leq \phi(\tilde{p})$. Portanto, a extensão de l_p para o vértice j pode ser podada por conduzir a um caminho com custo maior ou igual àquele em que o vértice j não é visitado. □

O algoritmo implementado tem sua execução interrompida quando o primeiro caminho com custo reduzido negativo for encontrado. Experimentos computacionais preliminares indicaram que o uso dessa estratégia resulta na diminuição do tempo total gasto com a resolução dos problemas de precificação.

2.4.2 Algoritmo Heurístico

O algoritmo heurístico implementado é baseado na metaheurística GRASP. Essa metaheurística constrói uma solução para o problema e, em seguida, realiza uma busca local em uma determinada vizinhança da solução construída. A fase de construção da solução é realizada combinando alguma aleatoriedade a uma estratégia gulosa. Para este objetivo, uma Lista Restrita de Candidatos (LRC) (do inglês, *Restricted Candiadate List*) é construída. Sejam C o conjunto de todos os elementos que, por atender aos aspectos de viabilidade, podem ser adicionados a uma solução em construção e $g(e)$ uma função que mede o custo incremental de adicionar um elemento $e \in C$ à solução. A LRC é formada pelos elementos do conjunto $\{e \in C : g_* \leq g(e) \leq g_* + \alpha(g^* - g_*)\}$, em que $g_* = min\{g(e) : e \in C\}$, $g^* = max\{g(e) : e \in C\}$ e α é um parâmetro satisfazendo $0 \leq \alpha \leq 1$ que controla o quão gulosa é a fase construtiva [Feo & Resende, 1995]. Por sua vez, a fase de busca local é aplicada para encontrar um mínimo local na vizinhança

analisada. Esse procedimento (fase construtiva seguida de busca local) é repetido até que uma determinada condição seja satisfeita.

A fase construtiva da heurística implementada é baseada no algoritmo CIA, o qual constrói uma rota por meio da inserção de um vértice não visitado a cada passo. Sejam S o conjunto de vértices visitados e $\bar{S} = V \setminus S$ o conjunto de vértices não visitados pela rota em construção. A fase construtiva preenche a LRC a partir de cada vértice $i \in \bar{S}$ e escolhe aleatoriamente um de seus elementos para compor a solução em construção. Esse processo é repetido até que uma rota que contenha pelo menos dois vértices além do depósito seja formada e não haja vértice $i \in \bar{S}$ cuja inserção diminua o custo reduzido da rota em construção. Ao término dessa fase, inicia-se a fase de busca local.

A fase de busca local é uma implementação da metaheurística VNS. A ideia básica de VNS reside na mudança sistemática de vizinhanças dentro de um algoritmo de busca local. Para este fim, as vizinhanças são ordenadas de acordo com sua complexidade, e há uma mudança de uma vizinhança para outra imediatamente mais custosa, se não houver melhoria no custo da solução corrente ao explorar aquela vizinhança. Sempre que a exploração de uma vizinhança resultar em uma melhoria da solução corrente, retorna-se para a vizinhança mais barata.

O pseudocódigo da metaheurística VNS é ilustrado no Algoritmo 1. Sejam $\mathcal{N}_h(x)$ uma vizinhança indexada por h e $f(x)$ uma função de custo de uma dada solução x. Na Linha 1, o índice da vizinhança h é inicializado e a solução corrente x é inicializada com base na solução x_0. As Linhas 2–6 são executadas até que o número máximo de vizinhanças h_{max} seja atingido. Na Linha 4, a solução corrente x e o índice h da vizinhança são atualizados, se houver uma solução aprimorante na vizinhança $\mathcal{N}_h(x)$. Caso contrário, o índice da vizinhança é incrementado na Linha 6, portanto uma vizinhança mais custosa passa a ser considerada. Por fim, a melhor solução encontrada é retornada na Linha 7.

No contexto da metaheurística VNS implementada, quatro vizinhanças ($h_{max} = 4$) foram utilizadas com base na política melhor-aprimorante, a qual consiste em avaliar todas as soluções da vizinhança e implementar o melhor movimento. Essas vizinhanças são discutidas a seguir.

A primeira vizinhança ($h = 1$), denominada *2-SWAP*, é composta por soluções obtidas pela permutação de dois dentre os vértices visitados. Sejam dois vértices quaisquer $i, j \in S$ visitados pela rota construída na fase construtiva. Uma solução da vizinhança em questão é obtida pela substituição de i por j e de j por i, resultando na mudança da ordem de visitação dos vértices em questão. Vale ressaltar que o cálculo do incremento no custo da solução corrente foi feito de forma eficiente, isto é, não se

Algorithm 1: VNS(x_0).

```
1 h = 1; x = x_0;
2 while h ≤ h_max do
3     if ∃s ∈ N_h(x) : f(s) < f(x) then
4         x = s; h = 1;
5     else
6         h = h + 1;
7 return x
```

fez necessário realizar a mudança dos vértices para obter a variação do custo.

A segunda vizinhança ($h = 2$), denominada *REMOVE*, é composta por soluções obtidas pela remoção de um dos vértices visitados. Seja um vértice qualquer $i \in S$ visitado pela rota construída na fase construtiva. Uma solução da vizinhança em questão é obtida pela remoção do vértice i da rota e pela reconexão das arestas entre o vértice predecessor e sucessor de i. O cálculo do incremento no custo da solução corrente foi feito de forma eficiente, levando em conta tanto o incremento na distância quanto o incremento associado à visitação do vértice i e à cobertura dos vértices cobertos somente por i.

A terceira vizinhança ($h = 3$), denominada *EXCHANGE*, é composta por soluções obtidas pela substituição de um dos vértices visitados por outro que ainda não o tenha sido. Seja $i \in S$ um vértice visitado pela rota construída na fase construtiva e $j \in \bar{S}$ um vértice que ainda não tenha sido visitado. Uma solução da vizinhança *EXCHANGE* é obtida pela substituição do vértice i pelo vértice j. Vale ressaltar que o cálculo do incremento foi feito de forma eficiente, levando em conta tanto o incremento na distância associado à remoção do vértice i e à inserção do vértice j, quanto o incremento associado à visitação dos vértices i e j e à cobertura dos vértices que deixam de ser cobertos pela remoção do vértice i e aqueles que passam a ser cobertos pela inserção do vértice j.

Por fim, a quarta e última vizinhança ($h = 4$), denominada *INSERTION*, é composta por soluções obtidas pela inserção de um vértice que ainda não tenha sido visitado. Seja $i \in \bar{S}$ um vértice que não tenha sido visitado pela rota construída na fase construtiva. Uma solução da vizinhança em questão é obtida pela inserção do vértice i nessa rota. A fim de determinar a melhor posição de inserção do vértice i, o algoritmo CIA implementado foi utilizado.

2.4.3 Paralelização dos Algoritmos

Paralelismo tem sido a solução recorrente para a limitação existente acerca do aumento da velocidade dos circuitos integrados que compõem os processadores de um computador. Em vez de construir processadores mais rápidos, a indústria tem colocado múltiplos processadores, cada um relativamente simples, em um único *chip*, formando, então, um processador *multicore* com várias unidades de processamento independentes [Pacheco, 2011]. Paralelismo consiste no particionamento da computação entre processadores.

Na literatura, são apresentados dois tipos de Paralelismo. No *Paralelismo de Dados*, os dados utilizados na resolução do problema são particionados entre os processadores, de modo que cada processador aplica o mesmo conjunto de operações em sua parte dos dados. Por sua vez, no *Paralelismo de Tarefas*, as tarefas a serem executadas para a resolução do problema são particionadas entre os processadores.

Tendo em vista a natureza do problema de precificação, deve-se resolver uma instância do PCMERR para cada veículo k em cada iteração do algoritmo. Portanto, a resolução do problema de precificação pode ser paralelizada com base em *Paralelismo de Dados*, já que o mesmo algoritmo (conjunto de operações) deve ser aplicado para cada veículo e conjunto de variáveis duais associadas (parte dos dados). Com base nessa observação, a resolução do problema de precificação foi paralelizada e é discutida a seguir.

A resolução do problema de precificação foi paralelizada utilizando OpenMP [Ope, 2012]. OpenMP é uma Interface de Programação Aplicada (do inglês, *Application Proramming Interface*-API) que fornece um conjunto de diretivas de compilação e rotinas para programação paralela. De um modo geral, o OpenMP permite que o programador apenas especifique o bloco de código que deve ser executado em paralelo, e aquele se encarrega de determinar em qual processador o bloco de código será executado. Com base nisso, a paralelização da resolução do problema de precificação foi bastante simples e consistiu apenas em especificar o bloco de código referente à resolução do problema de precificação como aquele a ser paralelizado, o que engloba a chamada ao algoritmo heurístico e ao algoritmo exato.

Em experimentos computacionais preliminares considerando K processadores, foi verificado uma diminuição de até 68% no tempo total gasto na resolução dos problemas de precificação com a versão paralela em relação àquela sequencial. Deste modo, fica clara a vantagem de paralelizar a resolução do problema de precificação.

2.5 Métodos de Estabilização

É senso comum que, de um modo geral, algoritmos de Geração de Colunas baseados no método Simplex possuem uma convergência lenta, uma vez que, em geral, um grande progresso é feito em direção à solução ótima nas primeiras iterações enquanto que o mesmo não se observa ao se aproximar do valor ótimo. Associado a isso, os valores das variáveis duais tendem a oscilar, demorando, então, a convergir para seus valores ótimos [Lübbecke & Desrosiers, 2004]. A fim de acelerar a convergência de algoritmos baseados em Geração de Colunas, foram desenvolvidos métodos de estabilização. Nas seções subsequentes, dois métodos de estabilização são discutidos e analisados no contexto do PRVSMM. Uma discussão sobre outros métodos de estabilização presentes na literatura pode ser encontrada em Lübbecke & Desrosiers [2004].

2.5.1 Estabilização por Pontos Interiores

Assumindo que o método Simplex tenha sido empregado para resolver o PML à otimalidade, as variáveis duais, fornecidas pelo resolvedor, são um ponto extremo na face ótima do poliedro dual. Ao longo das iterações de um algoritmo de Geração de Colunas, as variáveis duais podem oscilar, no sentido de assumir valores muito próximos ou muito distantes daqueles assumidos anteriormente, resultando em uma lenta convergência para seus valores ótimos.

Com o objetivo de amenizar a oscilação dos valores assumidos pelas variáveis duais, Rousseau et al. [2006] propuseram um método de estabilização que tem como objetivo gerar uma solução dual que seja um ponto interior da face ótima do problema dual associado ao Problema Mestre Restrito. Essa solução é utilizada em substituição daquela que é um dos pontos extremos da face ótima, resultando em uma melhor estimativa dos valores ótimos por estes não assumirem valores extremos [Rousseau et al., 2006]. Para este fim, o método consiste em gerar diversos pontos extremos na face dual ótima e realizar uma combinação linear convexa de todos os pontos encontrados. Essa combinação é utilizada como valor das variáveis duais no problema de precificação. A seguir, é apresentado como o método em questão foi aplicado ao PRVSMM.

Suponha que o PMLR tenha sido resolvido à otimalidade. Sejam $\tilde{\mathcal{P}}_k^*$ o conjunto de rotas p associadas ao veículo k para as quais $\lambda_k^p > 0$ (conjunto formado por variáveis básicas), L o conjunto de veículos k para os quais a restrição (2.10) não está justa, M o conjunto de vértices i para os quais a restrição (2.11) não está justa e, por fim, N o conjunto de vértices i para os quais a restrição (2.12) não está justa. Com base nas condições de folgas complementares, as variáveis duais ótimas devem satisfazer o

sistema abaixo:

$$\sum_{k\in K}\gamma_k^0 = 1, \tag{2.34}$$

$$-d^p\gamma_k^0 + \sum_{i\in V\setminus\{1\}}\delta_i^p\gamma_i^1 - \sum_{i\in V\setminus\{1\}}\beta_i^p\gamma_i^2 + \tau_k \leq 0, \forall p \in \tilde{\mathcal{P}}_k \setminus \tilde{\mathcal{P}}_k^* \tag{2.35}$$

$$-d^p\gamma_k^0 + \sum_{i\in V\setminus\{1\}}\delta_i^p\gamma_i^1 - \sum_{i\in V\setminus\{1\}}\beta_i^p\gamma_i^2 + \tau_k = 0, \forall p \in \tilde{\mathcal{P}}_k^* \tag{2.36}$$

$$\gamma_k^0 \geq 0, \forall k \in K \setminus L \tag{2.37}$$

$$\gamma_k^0 = 0, \forall k \in L \tag{2.38}$$

$$\gamma_i^1 \geq 0, \forall i \in V \setminus (M \cup \{1\}) \tag{2.39}$$

$$\gamma_i^1 = 0, \forall i \in M \setminus \{1\} \tag{2.40}$$

$$\gamma_i^2 \geq 0, \forall i \in V \setminus (N \cup \{1\}) \tag{2.41}$$

$$\gamma_i^2 \geq 0, \forall i \in N \setminus \{1\} \tag{2.42}$$

Para obter um ponto extremo na face dual ótima, resolve-se o seguinte problema de Programação Linear:

$$\max \quad \sum_{k\in K}\tau_k - \sum_{i\in V\setminus\{1\}}\gamma_i^2 + \sum_{i\in V\setminus\{1\}}\gamma_i^1 \tag{2.43}$$

sujeito às restrições (2.34)-(2.42). Na tentativa de obter pontos extremos distintos na face dual ótima, a função objetivo (2.43) é perturbada, no sentido de modificar seus coeficientes. Em Rousseau et al. [2006], os autores propuseram perturbar a função objetivo por meio da multiplicação dos vetores de custo da função objetivo por um vetor u de dimensão apropriada. Cada entrada do vetor u é um número aleatoriamente gerado no intervalo $[0, 1]$. Além disso, para cada vetor u considerado, os autores sugeriram também considerar o vetor $-u$ a fim de obter pontos extremos distantes. Deste modo, do problema de Programação Linear definido abaixo, é possível calcular pontos extremos na face dual ótima a partir de diversos valores de u:

$$\max \quad \sum_{k\in K}u_k\tau_k - \sum_{i\in V\setminus\{1\}}u_i^2\gamma_i^2 + \sum_{i\in V\setminus\{1\}}u_i^1\gamma_i^1, \tag{2.44}$$

sujeito às restrições (2.34)-(2.42).

Neste método, o único parâmetro a ser determinado é o número de pontos ex-

tremos a serem obtidos. Se por um lado a combinação linear convexa de um grande número de pontos extremos possivelmente resulta em um ponto mais central na face ótima, por outro um tempo maior é gasto já que um problema de Programação Linear deve ser resolvido para obter cada ponto. Em Rousseau et al. [2006], os autores apontaram que a obtenção de vinte pontos extremos se mostrou razoável para fins de estabilização, já que não houve ganho significativo em termos de estabilização ao considerar um número ainda maior de pontos.

Experimentos preliminares foram executados para avaliar a eficácia desta técnica de estabilização no algoritmo de Geração de Colunas proposto. Para todas as instâncias analisadas, não houve diminuição no número de iterações do algoritmo de Geração de Colunas. A saber, em alguns casos, o número de iterações se manteve igual, em outros, houve um aumento de até 47% no número de iterações do algoritmo. Como consequência, para aquelas instâncias em que houve aumento no número de iterações, observou-se um aumento de 182 a 900% no tempo total gasto para resolver o PML. Além disso, o tempo gasto no método de estabilização por pontos interiores correspondeu entre 10 e 47% do tempo total gasto pelo algoritmo. Em conclusão, o método em questão não atingiu seu objetivo, pois foi observado um aumento no número de iterações do algoritmo de Geração de Colunas para a maior parte das instâncias consideradas.

2.5.2 Estabilização por Relaxação Lagrangeana

Em vez de utilizar os valores obtidos a partir do método Simplex para as variáveis duais ótimas do PMLR, o método de Relaxação Lagrangeana pode ser utilizado para obter variáveis duais que residam na face dual ótima, portanto ótimas, mas que possivelmente não sejam pontos extremos desta. Para fins de estabilização do algoritmo de Geração de Colunas, as variáveis duais ótimas do PMLR, obtidas a partir do método Simplex, são tomadas como Multiplicadores Lagrangeanos iniciais válidos e, então, perturbadas por meio do método de Relaxação Lagrangeana. A seguir, é descrito como esse método foi aplicado para fins de estabilização no contexto do PRVSMM.

As restrições (2.9), (2.11) e (2.12) do PMLR foram relaxadas e dualizadas de forma Lagrangeana, e os Multiplicadores Lagrangeanos $\{\xi_k^0 \in \mathbb{R}_+ : k \in K\}$, $\{\xi_i^1 \in \mathbb{R}_+ : i \in V \setminus \{1\}\}$ e $\{\xi_i^2 \in \mathbb{R}_+ : i \in V \setminus \{1\}\}$ foram associados respectivamente às restrições (2.9), (2.11) e (2.12). Assuma que $\xi = (\xi^0, \xi^1, \xi^2)$ denota o vetor de Multiplicadores Lagrangeanos de dimensão apropriada. O Problema de Relaxação Lagrangeana (PRL) é definido por:

$$z(\xi) = \min \quad (1 - \sum_{k \in \mathcal{K}} \xi_k^0)w + \sum_{i \in V}(\xi_i^1 - \xi_i^2)$$
$$+ \sum_{k \in \mathcal{K}} \sum_{p \in \tilde{\mathcal{P}}_k} (d^p \xi_k^0 - \sum_{i \in V \setminus \{1\}} \delta_i^p \xi_i^1 + \sum_{i \in V \setminus \{1\}} \beta_i^p \xi_i^2)\lambda_k^p, \tag{2.45}$$
$$\sum_{p \in \tilde{\mathcal{P}}_k} \lambda_k^p = 1, \forall k \in \mathcal{K}, \tag{2.46}$$
$$w \geq 0, \lambda_k^p \geq 0. \tag{2.47}$$

O PRL se decompõe em $K + 1$ subproblemas Lagrangeanos independentes: um subproblema para a variável w e um para cada veículo $k \in \mathcal{K}$. Deste modo, $z(\xi) = z_w(\xi) + \sum_{k \in \mathcal{K}} z_k(\xi) + \sum_{i \in V}(\xi_i^1 - \xi_i^2)$, no qual:

$$z_k(\xi) = \min \quad \sum_{p \in \tilde{\mathcal{P}}_k} (d^p \xi_k^0 - \sum_{i \in V \setminus \{1\}} \delta_i^p \xi_i^1 + \sum_{i \in V \setminus \{1\}} \beta_i^p \xi_i^2)\lambda_k^p, \tag{2.48}$$
$$\sum_{p \in \tilde{\mathcal{P}}_k} \lambda_k^p = 1, \tag{2.49}$$
$$\lambda_k^p \geq 0, \tag{2.50}$$

e

$$z_w(\xi) = \min \quad (1 - \sum_{k \in \mathcal{K}} \xi_k^0)w, \tag{2.51}$$
$$w \geq 0. \tag{2.52}$$

O subproblema associado a cada veículo k, determinado por (2.48)-(2.50), é resolvido por inspeção, já que escolhendo a coluna $p \in \tilde{\mathcal{P}}_k$ com menor custo associado resulta na solução ótima. Vale ressaltar que a inspeção é feita sobre $\tilde{\mathcal{P}}_k$, em vez de $\mathcal{P}_k$, portanto apenas um conjunto restrito de rotas é inspecionado. Por sua vez, o subproblema referente à variável w, determinado por (2.51) e (2.52), possui valor ótimo quando $w = 0$, já que a desigualdade (2.14) impõe $1 - \sum_{k \in \mathcal{K}} \xi_k^0 \geq 0$, uma vez que $\xi_k^0 = \bar{\gamma}_k^0, \forall k \in \mathcal{K}$.

Após a resolução dos subproblemas Lagrangeanos, os Multiplicadores Lagrangeanos são atualizados por meio do Método de Subgradiente Defletido (MSD) [Camerini et al., 1975], o qual é uma variação do Método de Subgradiente (MS) [Held et al., 1974]; uma revisão dos principais métodos existentes pode ser en-

contrada em [Guignard, 2003]. A diferença central do MSD para o MS reside na direção de busca utilizada, isto é, a direção de busca é uma combinação linear convexa entre o subgradiente da iteração corrente e os subgradientes das iterações anteriores.

Algorithm 2: Algoritmo de Estabilização por Relaxação Lagrangeana

Data: $\bar{\gamma}_k^0, \bar{\gamma}_k^1, \bar{\gamma}_k^2$

1 $\xi_k^{0(0)} = \bar{\gamma}_k^0, \forall k \in \mathcal{K}$

2 $\xi_i^{1(0)} = \bar{\gamma}_k^1, \forall i \in V \setminus \{1\}$

3 $\xi_i^{2(0)} = \bar{\gamma}_k^2, \forall i \in V \setminus \{1\}$

4 **for** $t \leftarrow 0$ **to** t_{max} **do**

5 resolva o subproblema (2.48)-(2.50) e seja $\bar{\lambda_k}^p$ sua solução ótima, $\forall k \in \mathcal{K}$

6 $s_{\xi_k^0}^{(t)} = d^p \bar{\lambda}_k^p, \forall k \in \mathcal{K}$

7 $s_{\xi_i^1}^{(t)} = 1 - \sum_{k \in \mathcal{K}} \delta_i^p \bar{\lambda}_k^p, \forall i \in V \setminus \{1\}$

8 $s_{\xi_i^2}^{(t)} = -1 + \sum_{k \in \mathcal{K}} \beta_i^p \bar{\lambda}_k^p, \forall i \in V \setminus \{1\}$

9 **if** $t = 0$ **then**

10 $b^{(t)} = s^{(t)}$

11 **else**

12 $b^{(t)} = (1 - \nu) s^{(t)} + \nu b^{(t-1)}$

13 $\rho^{(t)} = \epsilon (UB - z(\xi^{(t)}) / \|b^{(t)}\|^2$

14 $\xi_k^{0(t+1)} = \max\{\xi_k^{0(t)} + \rho^{(t)} b_{\xi_k^0}^{(t)}, 0\}, \forall k \in \mathcal{K}$

15 $\xi_i^{1(t+1)} = \max\{\xi_i^{1(t)} + \rho^{(t)} b_{\xi_i^1}^{(t)}, 0\}, \forall i \in V \setminus \{1\}$

16 $\xi_i^{2(t+1)} = \max\{\xi_i^{2(t)} + \rho^{(t)} b_{\xi_i^2}^{(t)}, 0\}, \forall i \in V \setminus \{1\}$

O Algoritmo 2 apresenta os principais passos do algoritmo de estabilização baseado no método de Relaxação Lagrangeana. Assuma que todos os valores (multiplicadores, subgradientes, entre outros) envolvidos no método sejam indexados por (t), em que t representa uma dada iteração do MSD. O algoritmo recebe como entrada os multiplicadores duais ótimos do PMLR para realizar a atribuição dos valores iniciais dos Multiplicadores Lagrangeanos nas Linhas 1–3. As Linhas 5–16 são repetidas por um número máximo t_{max} de iterações. Na Linha 5, o subproblema Lagrangeano associado é resolvido por inspeção para cada veículo k, sendo $\bar{\lambda}_k^p$ sua solução ótima. Nas Linhas 6–8, as componentes do vetor subgradiente $s^{(t)} = (s_{\xi^0}^{(t)}, s_{\xi^1}^{(t)}, s_{\xi^2}^{(t)})$ são computadas. Nas Linhas 9–12, o vetor direção de busca $b^{(t)}$ é calculado, sendo $\nu \in (0, 1]$ um escalar real. Na Linha 13, o tamanho do passo ρ é calculado, sendo $\epsilon \in (0, 2]$ um escalar real, UB um limite superior válido para o PRVSMM e $\|b^{(t)}\|^2$ é o quadrado da norma Euclidiana do vetor de direção de busca $b^{(t)}$. Por fim, nas Linhas 14–16, os Multiplicadores Lagrangeanos são atualizados.

Os valores das variáveis duais utilizados na resolução do problema de precificação foram aqueles obtidos por meio da combinação linear convexa dos Multiplicadores Lagrangeanos encontrados ao longo da execução do algoritmo. A saber, a cada dez iterações, os valores dos Multiplicadores Lagrangeanos foram armazenados para serem utilizados na etapa de combinação linear convexa. É importante notar que o valor da variável dual τ_k deve ser atualizado também a cada iteração, apesar de a desigualdade associada a esta variável não ter sido relaxada nem dualizada de forma Lagrangeana. Por meio da restrição (2.15), a variável dual τ_k em cada iteração do algoritmo foi determinada como:

$$\tau_k = \min_{p \in \tilde{\mathcal{P}}_k} (d^p \gamma_k^0 - \sum_{i \in V \setminus \{1\}} \delta_i^p \gamma_i^1 + \sum_{i \in V \setminus \{1\}} \beta_i^p \gamma_i^2), \forall k \in \mathcal{K}. \tag{2.53}$$

Por fim, foi realizada uma combinação linear convexa das variáveis τ_k, conforme explanado anteriormente.

Experimentos preliminares foram executados para avaliar a eficácia desta técnica de estabilização no algoritmo de Geração de Colunas. Observou-se uma redução de até 28% no número de iterações do algoritmo de Geração de Colunas para todas as instâncias analisadas exceto uma, para a qual houve um aumento de 29% do número de iterações. Se por um lado o método foi capaz de diminuir o número de iterações do algoritmo para a maioria das instâncias analisadas, por outro observou-se um aumento de até 211% no tempo gasto para resolver o PML. O acréscimo do tempo total gasto foi resultado do aumento do tempo dispendido no problema de precificação, o qual foi ocasionado pelo aumento de até 66% do número de estados analisados por aquele algoritmo de precificação baseado em Programação Dinâmica. Em conclusão, o método de estabilização apresentado se mostrou eficaz para a estabilização do algoritmo de Geração de Colunas em todas as instâncias exceto uma, porém sua utilização se mostrou desvantajosa por ter aumentado o tempo total do algoritmo de Geração de Colunas.

2.6 Heurísticas para o PRVSMM

Com base no algoritmo de Geração de Colunas discutido anteriormente, duas heurísticas são propostas para o PRVSMM. Uma delas é uma Heurística Lagrageana (HL), cuja solução é obtida a partir da resolução dos subproblemas Lagrangeanos. A outra, por sua vez, consiste na resolução de um Problema de Programação Inteira com base no conjunto de colunas existentes no PMLR ao término do algoritmo de Geração de

Colunas. As duas heurísticas propostas são discutidas a seguir.

2.6.1 Heurística Lagrangeana

No PRVSMM, o conjunto das K rotas obtidas pela resolução dos subproblemas Lagrangeanos pode não cobrir todos os vértices ou algum vértice pode ser visitado por mais de um veículo, resultando em uma solução inviável. Soluções como essas podem ser convertidas em soluções viáveis por meio de heurísticas simples. Tais heurísticas são denominadas Heurísticas Lagrangeanas e são projetadas especificamente para cada problema [Guignard, 2003].

Neste trabalho, é proposta uma HL com o objetivo de obter limites primais para o PRVSMM. A heurística proposta consiste de duas fases para viabilização de uma solução obtida na resolução dos subproblemas lagrangeanos. A primeira fase trata a existência de vértices visitados por mais de um veículo por meio de sua remoção da maior das rotas. A segunda fase trata a existência de vértices que não foram cobertos por meio da inserção de algum vértice que não tenha sido visitado e que cubra aquele vértice não coberto; a inserção desse vértice é realizada na menor das rotas. Nesta etapa, o vértice a ser inserido é aquele cuja inserção resulte no menor incremento em termos de distância e que cubra o maior número de vértices dentre aqueles que ainda não foram cobertos.

2.6.2 Heurística de Geração de Colunas

Com o objetivo de obter soluções viáveis de boa qualidade, é proposta uma heurística denominada Heurística de Geração de Colunas (HGC). Seja $\bar{\mathcal{P}}$ o conjunto de colunas disponíveis quando o PML for resolvido à otimalidade ou quando o algoritmo de Geração de Colunas for interrompido por atingir o limite de tempo imposto. A HGC consiste em resolver, por meio de um algoritmo *Branch-and-Bound*, o problema definido por (2.8)-(2.12) em que o conjunto $\mathcal{P}$ é substituído por $\bar{\mathcal{P}}$. A solução ótima para este problema é uma solução viável para o PRVSMM e pode ser calculada rapidamente por haver um pequeno número de colunas em $\bar{\mathcal{P}}$.

A HGC difere da CGH porque as rotas em $\bar{\mathcal{P}}$, geradas ao longo do algoritmo de Geração de Colunas, foram construídas com base nos valores das variáveis duais, enquanto que as rotas consideradas pela CGH foram construídas com base em dados primais. Ainda, parte das rotas consideradas na HGC foram geradas por abordagem heurística ou exata, enquanto que as rotas consideradas pela CGH foram geradas apenas por meio de heurísticas.

2.7 Experimentos Computacionais

Esta seção apresenta um estudo do impacto da densidade e do número de veículos no algoritmo de Geração de Colunas. A seguir, é apresentado o ambimente de teste utilizado, o que inclui recursos computacionais e a configuração dos algoritmos, e a metodologia utilizada. Logo em seguida, é feito um estudo da qualidade das soluções obtidas pelas heurísticas propostas à medida que a densidade e o número de veículos são variados. Por fim, é discutido o desempenho do algoritmo de Geração de Colunas nos testes realizados.

2.7.1 Ambiente Computacional e Metodologia

Nos experimentos, seis instâncias da TSPLIB, dentre aquelas utilizadas em [Valle et al., 2011], foram utilizadas. A saber, as seguintes instâncias foram consideradas: (i) eil51, composta por 51 vértices, (ii) st70, composta por 70 vértices, (iii) eil76, composta por 76 vértices, (iv) rat99, composta por 99 vértices, (v) kroA100, composta por 100 vértices e (vi) eil101, composta por 101 vértices. Para gerar as instâncias para o PRVSMM, aquela metodologia apresentada em [Valle et al., 2011] foi utilizada e é brevemente discutida a seguir.

Para cada par de vértices $i, j \in V$, d_{ij} foi definido como o menor número inteiro maior ou igual à distância euclidiana entre i e j. Por sua vez, o parâmetro R foi definido como o valor necessário para alcançar a densidade d que é definida por $\frac{\sum_{i \in V} |\omega(i) \setminus \{i\}|}{n}$. Para cada instância da TSPLIB considerada, foram geradas instâncias para o PRVSMM considerando todas as combinações possíveis entre $K \in \{2, 3, 4, 5, 6\}$ e $d \in \{0.75, 10\}$. Vale ressaltar que os valores dos parâmetros K e d foram escolhidos com base na aplicação do PRVSMM, já que redes de sensores são geralmente densas e podem possuir diversos soverdouros móveis. Deste modo, pretende-se avaliar o desempenho dos algoritmos propostos em cenários mais próximos da realidade.

A fim de avaliar a qualidade dos limites primais obtidos pelas heurísticas HL e HGC propostas neste trabalho, o BC proposto em Valle et al. [2011] foi considerado. O limite de 4 horas de execução, adotado em Valle et al. [2011], foi imposto ao algoritmo de Geração de Colunas e ao BC.

Nenhum dos métodos de estabilização discutidos foi utilizado para fins de estabilização, embora a HL tenha sido avaliada. A versão paralela para resolução do problema de precificação foi adotada, sendo que o algoritmo exato é chamado apenas se o algoritmo heurístico falhar. Todas as rotas com custo reduzido negativo encontradas pelo algoritmo heurístico foram inseridas no modelo. Por fim, o conjunto de

colunas iniciais do algoritmo de Geração de Colunas foi gerado por meio do GRASP proposto por Valle et al. [2011].

Os experimentos foram executados em uma máquina Intel Xeon E5645 64 bits com 12 processadores, cada qual com 2.40GHz, e 32Gb de memória RAM, rodando o sistema operacional Ubuntu 12.04. Os algoritmos propostos foram implementados em C++ e compilados com o compilador GCC 4.4.3. O ILOG CPLEX [cpl, 2012] versão 12.1 foi utilizado como resolvedor de Programação Linear.

2.7.2 Avaliação de Limites Primais

As Tabelas 2.1 e 2.2 apresentam os resultados computacionais para o BC e as heurísticas HGC e HL. A primeira e a segunda coluna apresentam respectivamente o nome da instância e o número de veículos considerado. As quatro colunas seguintes apresentam respectivamente o melhor limite inferior (lb), o melhor limite superior (ub), o *gap* de dualidade e o tempo de execução em segundos para o BC. Para aquelas instâncias em que o BC não foi possível atestar a otimalidade dentro do limite máximo de tempo, o símbolo "-" é apresentado na coluna referente ao tempo de execução. Por fim, as duas últimas colunas apresentam respectivamente o valor obtido pela HGC e o valor da melhor solução obtida pela HL ao longo da execução do algoritmo de Geração de Colunas.

A Tabela 2.1 apresenta os resultados computacionais quando a densidade das instâncias consideradas foi ajustada para 0.75 e o número de veículos variou entre 2 e 6. O BC foi capaz de atestar a otimalidade de 0.30% das instâncias consideradas, sendo que os *gaps* de dualidade tenderam a crescer à medida que o número de veículos considerados foi aumentado. A HL obteve limites primais de melhor qualidade em relação a HGC em 96.66% das instâncias e em 50% das instâncias em relação ao melhor limite primal obtido pelo BC.

A Tabela 2.2 apresenta os resultados computacionais quando a densidade das instâncias consideradas foi ajustada para 10 e o número de veículos variou entre 2 e 6. Não houve instância para a qual o BC tenha sido capaz de atestar a otimalidade. A HGC obteve limites primais de melhor qualidade em relação a HL em 73.33% das instâncias e em 90% das instâncias em relação ao melhor limite primal obtido pelo BC.

2.7.3 Desempenho do Algoritmo de Geração de Colunas

A Tabela 2.3 apresenta os resultados computacionais do algoritmo Geração de Colunas para as instâncias consideradas. As três primeiras colunas apresentam respectivamente

Instância		BC				GC	
K	Nome	lb	ub	gap(%)	t(s)	HGC	HL
2	eil51	225	225	0.0	1601.96	309	232
	st70	314.12	413	23.95	-	468	428
	eil76	268.57	279	3.75	-	385	295
	rat99	554.59	699	20.66	-	913	702
	kroA100	9086.04	13233	31.34	-	15395	12698
	eil101	325.54	361	9.83	-	448	411
3	eil51	154.59	163	5.18	-	251	210
	st70	215.25	360	40.21	-	421	357
	eil76	182.94	196	6.67	-	297	262
	rat99	386.75	616	37.22	-	718	636
	kroA100	6280.19	11520	45.48	-	14274	8400
	eil101	222.11	281	20.96	-	341	303
4	eil51	119.97	146	17.84	-	182	147
	st70	168.83	295	42.77	-	412	331
	eil76	141.03	200	29.49	-	260	242
	rat99	305.21	665	54.10	-	704	569
	kroA100	4899.18	13553	63.85	-	11666	10519
	eil101	170.77	308	44.56	-	321	296
5	eil51	100.53	135	25.54	-	150	130
	st70	141.91	281	49.50	-	325	250
	eil76	116.75	191	38.88	-	232	216
	rat99	257.43	620	58.48	-	613	598
	kroA100	4096.04	12895	68.24	-	11737	9659
	eil101	140.57	294	52.19	-	282	268
6	eil51	87.95	140	37.19	-	138	125
	st70	125.01	271	53.87	-	245	248
	eil76	101.11	201	49.70	-	247	194
	rat99	228.15	615	62.90	-	595	542
	kroA100	3576.51	11653	69.31	-	10788	9124
	eil101	120.62	230	47.56	-	265	234

Tabela 2.1: Resultados computacionais com densidade igual a 0.75.

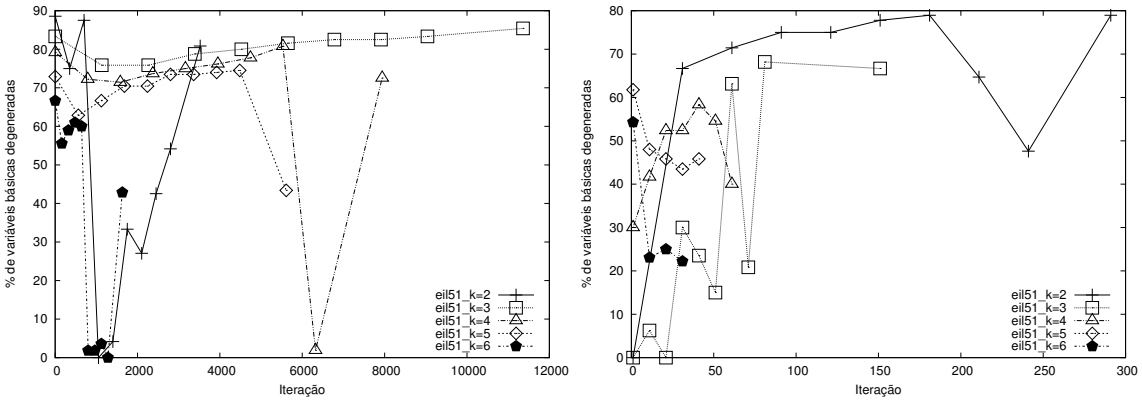
o nome, a densidade e o número de veículos da instância. A quarta coluna apresenta o limite de relaxação linear ou o símbolo "-" caso esse limite não tenha sido calculado dentro do limite de tempo imposto. A quinta coluna apresenta o tempo de execução em segundos ou o símbolo "-" para indicar que o algoritmo teve de ser interrompido por ter atingido o limite de tempo imposto. A sexta coluna apresenta a porcentagem de tempo total gasto para resolver os problemas de precificação. A sétima coluna apresenta o número total de iterações do algoritmo. A oitava coluna apresenta a porcentagem de iterações degeneradas, no sentido de que o valor da função objetivo se manteve inalterado em relação à iteração anterior. Por fim, a última coluna apresenta a porcentagem de rótulos dominados e podados no algoritmo de Programação Dinâmica. Uma discussão acerca dos dados apresentados nesta tabela é feita a seguir.

O algoritmo pôde calcular o limite de relaxação linear dentro do limite de tempo

Instância		BC				GC	
K	Nome	lb	ub	gap(%)	t(s)	HGC	HL
2	eil51	106.01	125	15.20	-	123	124
	st70	128.16	245	47.69	-	224	209
	eil76	104.12	161	35.34	-	151	151
	rat99	166.87	511	67.35	-	437	432
	kroA100	2492.09	9992	75.06	-	6059	6074
	eil101	95.09	188	49.42	-	145	145
3	eil51	80.44	100	19.57	-	104	111
	st70	89.76	237	62.13	-	200	199
	eil76	76.19	147	48.18	-	132	124
	rat99	120.90	534	77.36	-	390	418
	kroA100	1819.12	10521	82.71	-	4995	5545
	eil101	68.81	174	60.46	-	127	127
4	eil51	71.01	100	29.00	-	95	94
	st70	76.02	220	65.45	-	178	188
	eil76	63.57	161	60.52	-	121	120
	rat99	110.99	550	79.82	-	387	388
	kroA100	1600.01	15090	89.40	-	4899	5278
	eil101	57.41	173	66.82	-	102	105
5	eil51	76.01	90	15.55	-	91	91
	st70	70.12	226	68.98	-	179	177
	eil76	60.01	155	61.29	-	117	119
	rat99	108.16	613	82.36	-	387	394
	kroA100	1569.55	8393	81.30	-	4755	4741
	eil101	52.01	142	63.38	-	95	106
6	eil51	77.00	90	14.44	-	90	91
	st70	67.12	216	68.93	-	177	177
	eil76	59.71	138	56.73	-	111	113
	rat99	107.49	570	81.14	-	387	392
	kroA100	1581.04	8212	80.75	-	4503	4593
	eil101	50.92	164	68.95	-	95	104

Tabela 2.2: Resultados computacionais com densidade igual a 10.

imposto para 0 das 30 (0%) instâncias quando a densidade foi configurada para 0.75 e para 10 das 30 (33.33%) instâncias quando a densidade foi configurada para 10. Nota-se que o tempo gasto para calcular o limite de relaxação linear tendeu a decrescer à medida que o número de veículos foi aumentado. Observa-se que a porcentagem de tempo gasto na resolução dos problemas de precificação foi superior a 90% em 38 das 60 (63.33%) instâncias consideradas; para a instância eil101 com densidade igual a 0.75 e número de veículos igual a 2, a porcentagem de tempo gasto na resolução dos problemas de precificação foi igual a 10.16%, fato este que pode ser justificado pelo elevado número de colunas inseridas no modelo em decorrência do grande número de iterações do algoritmo. Nota-se que o número de iterações tendeu a decrescer à medida que o número de veículos foi aumentado. Percebe-se que, em 21 das 60 (35.00%) instâncias, a porcentagem de iterações degeneradas foi superior a 90%; em



(a) Densidade 0.75 (b) Densidade 10

Figura 2.2: Porcentagem de variáveis básicas degeneradas ao longo das iteraçãoes para a instância eil51.

18 das 30 (60.00%) instâncias com densidade igual a 0.75, a porcentagem de iterações degeneradas foi igual a 100%. Por fim, a porcentagem de caminhos dominados ou podados foi inferior a 83.18%. A seguir, é apresentada uma análise da porcentagem de variáveis básicas degeneradas ao longo do algoritmo para os cenários e casos analisados.

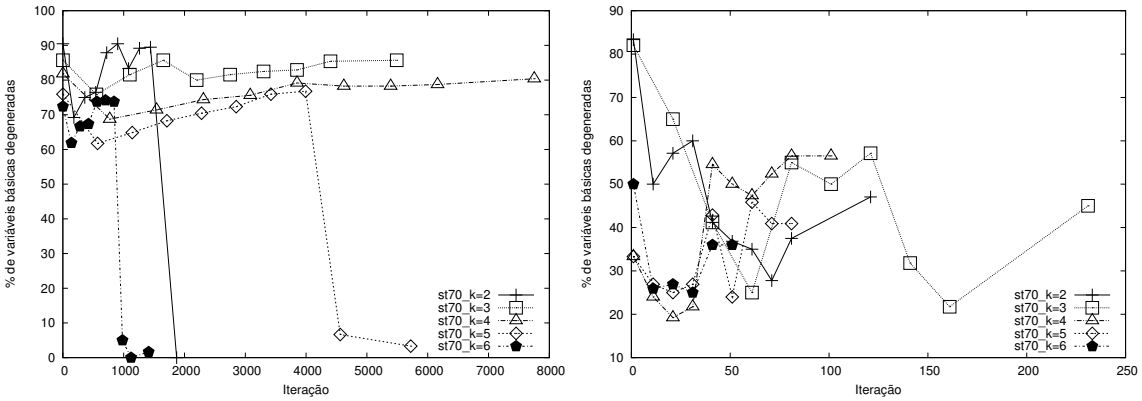
As Figuras 2.2-2.7 apresentam a porcentagem de variáveis básicas degeneradas ao longo das iterações do algoritmo de Geração de Colunas. A cada dez iterações, a porcentagem de variáveis básicas degeneradas foi calculada. Por conta do grande número de iterações em algumas instâncias e um pequeno número em outras, as figuras apresentam a porcentagem de degeneração de no máximo dez pontos amostrais. A saber, quando possível, apresenta-se a porcentagem referente à primeira e à última iteração juntamente com aquelas referentes a outras oito iterações intermediárias. Vale ressaltar que há casos em que apenas um ou dois pontos são apresentados devido ao pequeno número de iterações do algoritmo para a instância considerada.

A Figura 2.2 apresenta a porcentagem de variáveis básicas degeneradas ao longo das iterações do algoritmo para a instância eil51 em relação às densidades consideradas. Nos dois cenários analisados, verifica-se um comportamento irregular ao longo das iterações para as instâncias consideradas.

A Figura 2.3 apresenta a porcentagem de variáveis básicas degeneradas ao longo das iterações do algoritmo para a instância st70 em relação às densidades consideradas. No cenário em que a densidade foi configurada para 0.75, observa-se um comportamento regular para algumas instâncias consideradas. Por sua vez, no cenário em que a densidade foi configurada para 10, observa-se comportamento irregular ao longo das iterações.

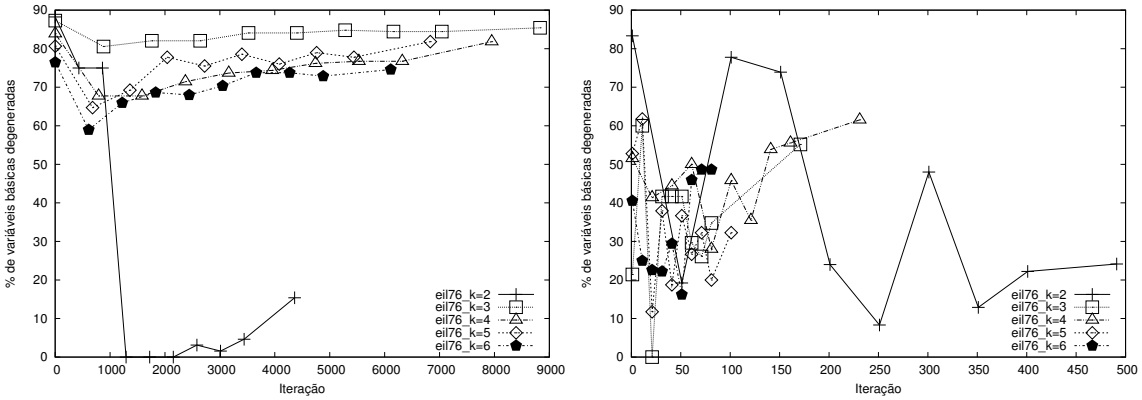
Instância	Densidade	K	lb	t(s)	%tp	#iter	%iter	%dom
eil51	0.75	2	-	-	92.18	3521	70.41	68.08
		3	-	-	20.89	11367	100.00	3.81
		4	-	-	46.62	7942	84.85	59.39
		5	-	-	66.96	5619	87.20	24.70
		6	-	-	96.65	1630	52.64	42.12
	10	2	102.50	582.84	96.75	292	90.75	48.65
		3	74.33	913.30	98.56	155	58.06	40.13
		4	64.75	79.64	94.31	70	75.71	41.04
		5	59.60	58.05	95.75	47	85.11	42.42
		6	54.67	435.12	99.54	37	43.24	37.88
st70	0.75	2	-	-	95.85	1875	97.17	10.83
		3	-	-	72.66	5499	100.00	11.91
		4	-	-	29.95	7750	100.00	8.29
		5	-	-	49.80	5725	79.91	8.99
		6	-	-	94.72	1412	63.95	19.89
	10	2	-	-	99.93	120	65.00	69.09
		3	-	-	99.78	232	68.97	55.59
		4	114.75	3012.91	99.47	103	67.96	46.77
		5	98.00	2158.59	99.37	86	62.79	50.80
		6	101.17	666.84	99.03	58	56.90	42.70
eil76	0.75	2	-	-	77.29	4364	36.89	38.97
		3	-	-	26.02	8838	100.00	5.22
		4	-	-	21.61	7958	100.00	5.09
		5	-	-	25.92	6833	100.00	4.11
		6	-	-	27.07	6119	100.00	2.78
	10	2	-	-	99.46	495	54.34	56.03
		3	-	-	99.79	179	58.66	47.56
		4	-	-	99.66	239	76.15	52.53
		5	-	-	99.86	102	43.14	46.67
		6	-	-	99.91	84	36.90	45.77
rat99	0.75	2	-	-	61.76	4745	63.46	42.67
		3	-	-	86.70	2623	100.00	8.88
		4	-	-	67.01	4106	7.18	25.53
		5	-	-	98.73	373	100.00	3.87
		6	-	-	94.97	947	100.00	5.79
	10	2	-	-	99.94	49	44.90	83.18
		3	-	-	99.88	69	33.33	85.24
		4	-	-	99.54	231	39.83	53.78
		5	-	-	99.70	200	34.00	52.96
		6	-	-	99.93	41	78.05	43.39
kroA100	0.75	2	-	-	69.20	3947	83.79	25.11
		3	-	-	70.99	4366	23.29	55.38
		4	-	-	24.24	6735	100.00	7.05
		5	-	-	27.25	5784	100.00	5.63
		6	-	-	36.41	4822	100.00	7.62
	10	2	-	-	99.95	42	69.05	68.66
		3	-	-	99.98	25	60.00	55.27
		4	-	-	99.91	45	84.44	42.35
		5	2971.00	5845.96	99.56	123	73.98	33.60
		6	2621.50	5450.57	99.66	68	61.76	37.82
eil101	0.75	2	-	-	10.16	6725	100.00	1.32
		3	-	-	36.57	6798	100.00	0.44
		4	-	-	23.89	6734	100.00	0.67
		5	-	-	21.38	6052	100.00	3.59
		6	-	-	25.42	5290	99.98	4.01
	10	2	-	-	99.38	402	65.17	68.86
		3	-	-	99.89	54	42.59	72.61
		4	-	-	99.79	77	46.75	54.64
		5	-	-	99.87	63	36.51	50.15
		6	-	-	99.87	89	67.42	52.61

Tabela 2.3: Resultados computacionais para as instâncias PRVSMM.



(a) Densidade 0.75 (b) Densidade 10

Figura 2.3: Porcentagem de variáveis básicas degeneradas ao longo das iteraçãoes para a instância st70.

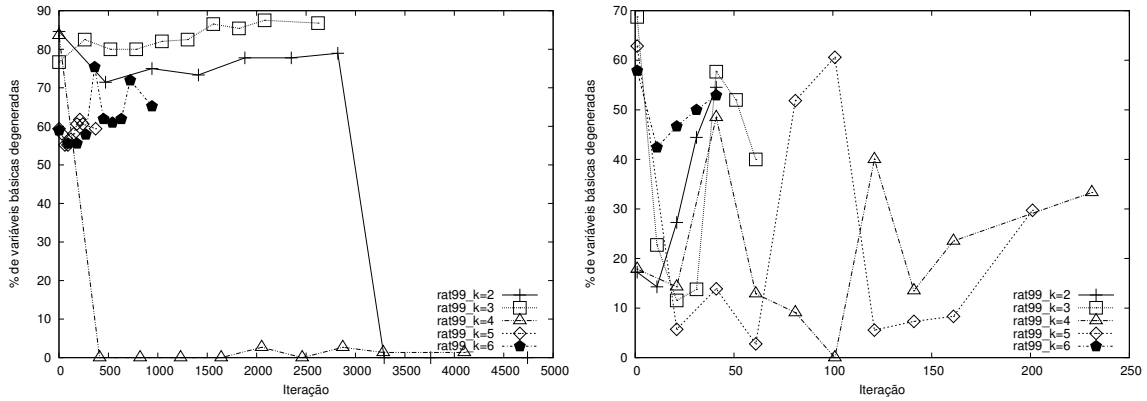


(a) Densidade 0.75 (b) Densidade 10

Figura 2.4: Porcentagem de variáveis básicas degeneradas ao longo das iteraçãoes para a instância eil76.

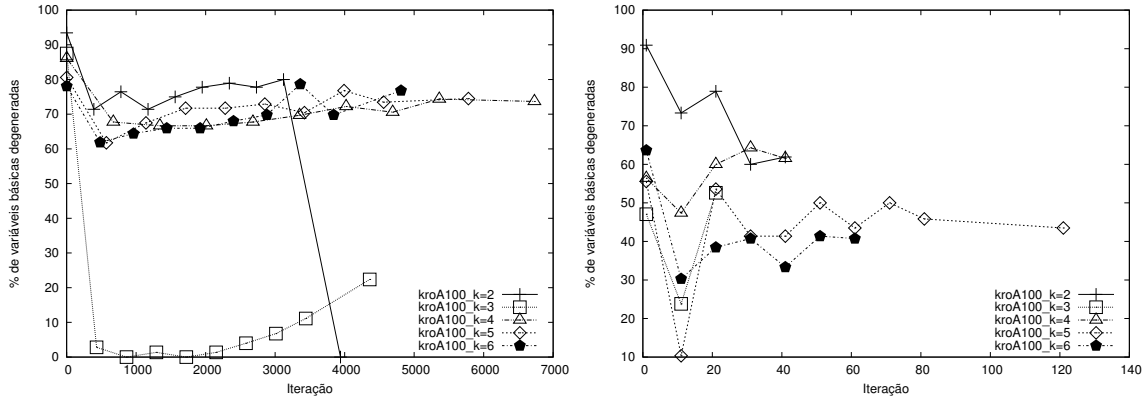
A Figura 2.4 apresenta a porcentagem de variáveis básicas degeneradas ao longo das iterações do algoritmo para a instância eil76 em relação às densidades consideradas. Quando a densidade foi configurada para 0.75, observa-se que a porcentagem de variáveis básicas degeneradas se manteve acima de 50% para todos os casos exceto um, em que houve uma significativa queda nas últimas iterações. No caso em que a densidade foi configurada para 10, observa-se um comportamento irregular ao longo das iterações.

A Figura 2.5 apresenta a porcentagem de variáveis básicas degeneradas ao longo das iterações do algoritmo para a instância rat99 em relação às densidades consideradas. No cenário em que a densidade foi configurada para 0.75, observa-se um comportamento



(a) Densidade 0.75 (b) Densidade 10

Figura 2.5: Porcentagem de variáveis básicas degeneradas ao longo das iterações para a instância rat99.

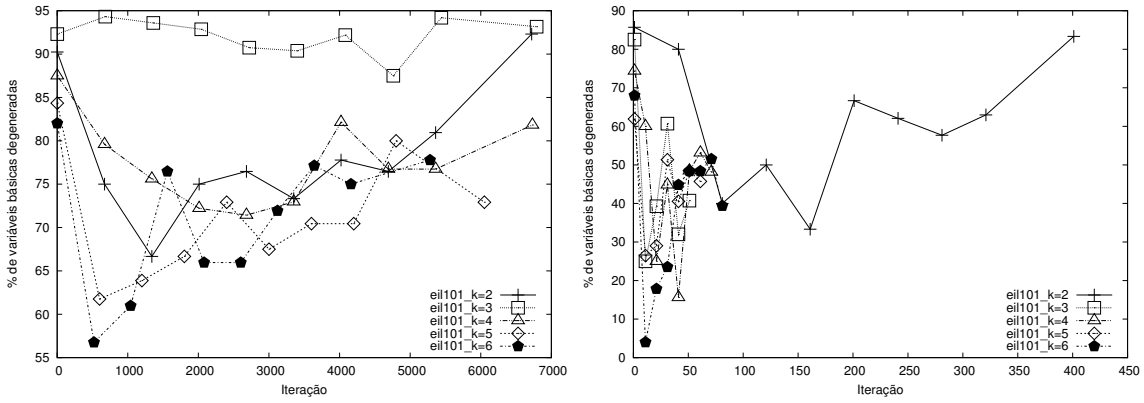


(a) Densidade 0.75 (b) Densidade 10

Figura 2.6: Porcentagem de variáveis básicas degeneradas ao longo das iterações para a instância kroA100.

regular na maioria das instâncias. Vale ressaltar que a porcentagem de iterações degeneradas se manteve abaixo de 10% na maioria das iterações quando o número de veículos foi configurado para seis. No caso em que a densidade foi configurada para 10, observa-se um comportamento irregular ao longo das iterações.

A Figura 2.6 apresenta a porcentagem de variáveis básicas degeneradas ao longo das iterações do algoritmo para a instância kroA100 em relação às densidades consideradas. Quando a densidade foi configurada para 0.75, observa-se que a porcentagem de variáveis básicas degeneradas variou entre 0% e 90%. Além disso, observa-se um comportamento regular ao longo das iterações. Para o cenário em que a densidade considerada foi igual a 10, observa-se um comportamento irregular ao longo das iterações.



(a) Densidade 0.75 (b) Densidade 10

Figura 2.7: Porcentagem de variáveis básicas degeneradas ao longo das iterações para a instância eil101.

A Figura 2.7 apresenta a porcentagem de variáveis básicas degeneradas ao longo das iterações do algoritmo para a instância eil101 em relação às densidades consideradas. Observa-se que a porcentagem de variáveis básicas degeneradas variou entre 55% e 95% para todos os casos quando a densidade foi configurada para 0.75. Além disso, nota-se um comportamento irregular ao longo das iterações. No caso em que a densidade foi configurada para 10, observa-se um comportamento irregular ao longo das iterações .

Com base nos gráficos apresentados acerca da porcentagem de variáveis básicas ao longo das iterações, percebe-se que a reformulação proposta para as instâncias e cenários avaliados é bastante degenerada. Na maioria das iterações dos cenários e casos avaliados, a porcentagem de variáveis degeneradas se manteve acima de 50% quando a densidade foi configurada para 0.75 e um comportamento irregular foi observado ao longo da iterações quando a densidade foi configurada para 10.

2.8 Conclusões

Neste capítulo, foi estudado o PRVSMM. Na literatura, são encontrados trabalhos que propõem métodos heurísticos e exatos para este problema. Nesses trabalhos, a otimalidade foi atestada para instâncias de tamanho relativamente pequeno, havendo instâncias com *gap* de dualidade acima de 40%. Deste modo, fazem-se necessários métodos para tornar este problema melhor resolvido na prática.

Este trabalho propõe uma reformulação por recobrimento de conjuntos, um algoritmo de Geração de Colunas, uma Heurística de Geração de Colunas e uma Heurística

Lagrangeana para o PRVSMM. Para resolução do problema de precificação, foram implementados um algoritmo exato e um heurístico. A saber, um algoritmo baseado em Programação Dinâmica foi implementado, sendo que regras de dominância e de poda foram propostas a fim de diminuir o tempo gasto na resolução dos subproblemas. Na tentativa de estabilizar o algoritmo, duas técnicas de estabilização foram investigadas. Em experimentos preliminares, o uso dessas técnicas não se mostrou vantajoso e, portanto, nenhuma das técnicas foi utilizada para fins de estabilização. Estudos computacionais foram realizados para comparar a qualidade dos limites primais fornecidos pelos métodos heurísticos propostos com aqueles fornecidos pelos métodos encontrados na literatura à medida que a densidade e o número de veículos considerados foram variados. Ainda, o desempenho do algoritmo de Geração de Colunas foi avaliado empiricamente, em especial o efeito da degeneração foi avaliado nas instâncias consideradas.

De um modo geral, os métodos propostos neste trabalho não se mostraram competitivos em relação àqueles encontrados na literatura para as instâncias consideradas. O algoritmo de Geração de Colunas proposto foi capaz de calcular o limite de relaxação linear de apenas 16.66% das instâncias consideradas dentro de quatro horas de execução, tornando inviável a implementação de um algoritmo *Branch-and-Price*. Com base no estudo feito acerca do desempenho deste algoritmo, indubitavelmente, o gargalo reside na resolução do problema de precificação, que consumiu mais de 90% do tempo total na maioria das instâncias consideradas mesmo com a paralelização de sua resolução e com o uso do algoritmo heurístico. Associado a isso, com base nos experimentos realizados, o efeito de degeneração degrada o desempenho do algoritmo, já que uma parte considerável das iterações foram degeneradas, no sentido do valor função objetivo não ter decrescido entre iterações. Os métodos heurísticos propostos consumiram quatro horas na maioria dos casos avaliados, já que sua execução está condicionada à finalização do algoritmo de Geração de Colunas, finalização esta que ocorre por limite de tempo ou por ter resolvido o PML. Notou-se que a HL obteve soluções melhores em relação a HGC quando a densidade considerada foi igual a 0.75, sendo que este cenário se inverteu quando a densidade considerada foi configurada para 10.

Diferentemente do que é observado em outras variações do PRV presentes na literatura, com base no estudo realizado neste trabalho, o algoritmo de Geração de Colunas proposto não obteve bons resultados para o PRVSMM. A fim de mudar esse cenário, fazem-se necessárias técnicas para resolver o problema de precificação de modo mais eficiente. Por exemplo, podem ser encontradas novas regras de dominância e estratégias de poda, como também pode-se encontrar uma relaxação válida para o problema de precificação que possa ser resolvida por algum algoritmo pseudo-polinomial.

Capítulo 3

Algoritmo *Branch-and-Price* para o Problema de Cobertura Multi-Veículo

Neste capítulo, o Problema de Cobertura Multi-Veículo é estudado. Um algoritmo *Branch-and-Price* e uma heurística são apresentados para o problema. Resultados dos experimentos computacionais realizados são apresentados e discutidos ao final do capítulo.

3.1 Definição do Problema

O PCMV é definido sobre um grafo $G = (V \cup W, E)$ simples, completo, finito e não-direcionado com conjunto $V \cup W$ de vértices e com conjunto E de arestas. O vértice $1 \in T$ é considerado o depósito de onde partem até m veículos idênticos. O conjunto V representa o conjunto de vértices que podem ser visitados pelos veículos. Há também um subconjunto $T \subseteq V$ de vértices terminais que devem obrigatoriamente ser visitados pelo conjunto de rotas. O conjunto W, por sua vez, representa o conjunto de vértices que não podem ser visitados, mas que devem ser cobertos, no sentido de estar a uma distância pré-definida c de algum vértice visitado. Por fim, uma matriz de distância $D = (d_{ij})$ que satisfaz à propriedade de desigualdade triangular é definida sobre E, de modo que $d_{ij} = d_{ji}$ e $d_{ii} = 0$.

No PCMV, deve-se encontrar um conjunto de no máximo m rotas (uma para cada veículo) de modo a minimizar a soma do comprimento de cada uma das rotas, satisfazendo as seguintes restrições: (i) cada vértice $v \in V$ pode ser visitado por no

máximo uma única rota, (ii) cada vértice $v \in T$ deve ser visitado por exatamente uma rota, (iii) cada vértice $v \in W$ deve ser coberto por algum vértice visitado, (iv) o número de vértices em uma rota não pode ultrapassar o valor de um parâmetro p e (v) o comprimento de uma rota não pode ultrapassar o valor de um parâmetro q.

Apesar de haver alguma semelhança entre o PCMV e o PRVSMM, este não é um caso especial daquele (nem vice-versa). O PCMV difere do PRVSMM por quatro aspectos: (i) o problema não possui natureza min-max, (ii) há restrição que determine quais vértices devem ser obrigatoriamente cobertos e quais visitados, (iii) há restrição no comprimento de uma rota e (iv) há restrição acerca do número máximo de vértices em uma rota.

O PCMV é $\mathcal{NP}$-Difícil, já que o Problema do Caixeiro Viajante é um de seus casos particulares quando $m = 1, T = V, W = \emptyset$ e não são impostas as restrições de capacidade ($p = |V|$ e $q = \infty$).

3.2 Revisão Bibliográfica

O PCMV foi proposto em Hachicha et al. [2000]. Naquele trabalho, uma formulação de Programação Inteira com um número exponencial de restrições do tipo GSEC e três heurísticas foram propostas. Vale ressaltar que, apesar de os autores terem proposto uma formulação, nenhum algoritmo foi implementado para resolver de forma exata o PCMV. As heurísticas propostas foram avaliadas em instâncias em que a posição dos vértices foi aleatoriamente gerada e em uma instância que representa uma aplicação real. Vale ressaltar que os métodos propostos não fornecem nenhuma garantia de aproximação da solução ótima.

Um tentativa para propor um algoritmo exato para o PCMV é encontrada em Jozefowiez [2012]. Naquele trabalho, os autores afirmaram ter proposto uma Reformulação por Recobrimento de Conjuntos e um algoritmo de Geração de Colunas para o PCMV. Contudo, a reformulação proposta não está de acordo com a definição original do problema, já que em uma solução encontrada pelo algoritmo proposto: (i) cada vértice em T pode não ser visitado exatamente uma única vez, e (ii) uma rota pode não satisfazer as restrições acerca do número máximo de vértices visitados e de comprimento máximo. Ainda, os autores nem sequer consideraram o conjunto T na definição do problema. Os autores modelaram o problema de precificação como um *Ring Star Problem* [Labbé et al., 2004] e como um PCMERR. Para a primeira modelagem, os autores implementaram um BC, enquanto que, para a segunda, o algoritmo apresentado em [Feillet et al., 2004] foi implementado. Em termos de experimentos

computacionais, os autores apresentaram resultados computacionais para seis instâncias. Contudo, nem como as instâncias foram geradas nem qual abordagem foi utilizada para resolver o problema de precificação foram indicados. Deste modo, com base na revisão bibliográfica realizada, não há trabalho que tenha proposto um algoritmo exato para resolver o PCMV, originalmente definido em Hachicha et al. [2000].

3.3 Reformulação por Recobrimento de Conjuntos

A formulação proposta neste trabalho é uma reformulação do PCMV por Recobrimento de Conjuntos. Antes de apresentar a reformulação, fazem-se necessárias algumas definições. Sejam um grafo $G = (V \cup W, E)$ e R o conjunto de todas as rotas viáveis sobre G para o PCMV. Para cada rota $r \in R$, os seguintes parâmetros são definidos: (i) d^r representa o comprimento da rota, (ii) η^r representa o número de vértices visitados pela rota, (iii) δ_i^r assume valor 1 se r cobrir o vértice $i \in W$ ou valor 0 caso contrário, (iv) β_i^r assume valor 1 se r visitar o vértice $i \in V$ ou valor 0 caso contrário, e (v) θ_{ij}^r assume valor 1 se r utilizar o arco (i, j) ou valor 0 caso contrário. Por fim, seja $\lambda_r \in \mathbb{B}$ uma variável que assume valor 1 se a rota $r \in R$ for utilizada ou valor 0 caso contrário.

O PM é definido por:

$$\min \quad \sum_{r \in R} d^r \lambda_r, \tag{3.1}$$

$$-\sum_{r \in R} \lambda_r \geq -m, \tag{3.2}$$

$$-\sum_{r \in R} \beta_i^r \lambda_r \geq -1, \forall i \in V \setminus T, \tag{3.3}$$

$$\sum_{r \in R} \beta_i^r \lambda_r = 1, \forall i \in T \setminus \{1\}, \tag{3.4}$$

$$\sum_{r \in R} \delta_i^r \lambda_r \geq 1, \forall i \in W, \tag{3.5}$$

$$\lambda_r \in \{0, 1\}, \forall r \in R. \tag{3.6}$$

A função objetivo (3.1) minimiza a soma das distâncias. A restrição (3.2) garante que no máximo m rotas sejam selecionadas. As restrições (3.3) asseguram que cada vértice $i \in V \setminus T$ seja visitado no máximo uma vez, enquanto as restrições (3.4) forçam que cada vértice $i \in T \setminus \{1\}$ seja visitado exatamente uma vez. As restrições (3.5) garantem que cada vértice $i \in W$ seja coberto. Por fim, a restrição (3.6) define a integralidade das variáveis de decisão.

Considera-se como PML o problema definido por (3.1)-(3.5), em que as restrições (3.6) são substituídas por $\lambda_r \geq 0, \forall r \in R$. Associando-se variáveis duais τ, γ_i^1, γ_i^2, γ_i^3 respectivamente às restrições (3.2)-(3.5) do PML, obtém-se o Problema Dual do Mestre Linear:

$$\max \quad -m\tau - \sum_{i \in V \setminus T} \gamma_i^1 + \sum_{i \in T \setminus \{1\}} \gamma_i^2 + \sum_{i \in W} \gamma_i^3, \tag{3.7}$$

$$-\tau - \sum_{i \in V \setminus T} \beta_i^r \gamma_i^1 + \sum_{i \in T \setminus \{1\}} \beta_i^r \gamma_i^2 + \sum_{i \in W} \delta_i^r \gamma_i^3 \leq d^r, \forall r \in \tilde{R}, \tag{3.8}$$

$$\tau \geq 0, \gamma_i^1 \geq 0, \gamma_i^3 \geq 0. \tag{3.9}$$

Por conta do número exponencial de variáveis no PML, faz-se necessário um procedimento de Geração de Colunas. Para este fim, formula-se um PMLR em que R é substituído por conjunto $\tilde{R} \subset R$ ($\tilde{R} \ll R$) de rotas no PML.

Suponha que $\bar{\lambda}$ seja a solução ótima para o PMLR e $\bar{\tau}$, $\bar{\gamma}_i^1$, $\bar{\gamma}_i^2$, $\bar{\gamma}_i^3$ sejam as respectivas variáveis duais ótimas. Se não houver rota $r \in R \setminus \tilde{R}$ que viole a restrição dual

$$d^r + \bar{\tau} + \sum_{i \in V \setminus T} \beta_i^r \bar{\gamma}_i^1 - \sum_{i \in T \setminus \{1\}} \beta_i^r \bar{\gamma}_i^2 - \sum_{i \in W} \delta_i^r \bar{\gamma}_i^3 \geq 0, \tag{3.10}$$

então $\bar{\lambda}$ resolve o PML. Caso contrário, existe pelo menos uma rota $\hat{r} \in R \setminus \tilde{R}$ que deve ser inserida no PMLR, o qual será resolvido novamente. Com o objetivo de encontrar uma rota que viole a restrição (3.10) ou de obter um certificado de otimalidade para o PML, deve-se resolver um problema de precificação que é definido por:

$$r^* = \arg\min_{r \in R} \quad \phi(r), \tag{3.11}$$

em que $\phi(r) = d^r + \bar{\tau} + \sum_{i \in V \setminus T} \beta_i^r \bar{\gamma}_i^1 - \sum_{i \in T \setminus \{1\}} \beta_i^r \bar{\gamma}_i^2 - \sum_{i \in W} \delta_i^r \bar{\gamma}_i^3$ representa o custo reduzido de uma rota r.

O problema de precificação em questão corresponde ao PCMERR, já discutido anteriormente, de modo a atender às restrições de capacidade: $\eta^r \leq p$ e $d^r \leq q$. Os algoritmos implementados para a resolução do PCMERR no contexto do PCMV serão discutidos na próxima seção.

3.4 Resolvendo o Problema de Precificação

Foram implementados um algoritmo exato e um algoritmo heurístico para resolver o problema de precificação. Inicialmente, o algoritmo heurístico é executado. Caso este não consiga encontrar alguma rota com custo reduzido negativo, o algoritmo exato é executado para ou encontrar uma rota com essa propriedade ou atestar a otimalidade. Esses algoritmos são discutidos a seguir.

3.4.1 Algoritmo Exato

O algoritmo exato implementado é aquele apresentado em [Feillet et al., 2004], já discutido anteriormente. No algoritmo implementado, um rótulo l_r é representado por $\langle d^r, \eta^r, \phi(r), \upsilon(r), \varphi(r), \pi(r)\rangle$, onde (a) para cada $i \in V \setminus T$, $\upsilon^i(r)$ assume valor 1 se i for visitado por r ou valor 0 caso contrário, (b) para cada $i \in T \setminus \{1\}$, $\varphi^i(r)$ assume valor 1 se i for visitado por r ou valor 0 caso contrário, e (c) para cada $i \in W$, $\pi^i(r)$ assume valor 1 se i for coberto por r ou valor 0 caso contrário.

Com o objetivo de limitar a extensão de caminhos parciais que conduzam a rotas sub-ótimas, uma regra de dominância e uma regra de poda foram desenvolvidas. Antes de apresentar as regras, fazem-se necessárias as seguintes definições: (a) $\Gamma_1^- = \{j \in V \setminus T : \bar{\gamma}_j^1 < 0\}$ representa o conjunto de vértices que possuem custo dual negativo, (b) $\Gamma_2^+ = \{j \in T \setminus \{1\} : \bar{\gamma}_j^2 > 0\}$ representa o conjunto de vértices que possuem custo dual positivo, (c) $\Gamma_3^+ = \{j \in W : \bar{\gamma}_j^3 > 0\}$ representa o conjunto de vértices que possuem custo dual positivo e (d) $\omega(i) = \{j \in W : d_{ij} \leq c\}, \forall i \in V$, representa o conjunto dos vértices cobertos pelo vértice i.

Proposição 4. *Sejam* $l_r = \langle d^r, \eta^r, \phi(r), \upsilon(r), \varphi(r), \pi(r)\rangle$ *e* $l_{\tilde{r}} = \langle d^{\tilde{r}}, \eta^{\tilde{r}}, \phi(\tilde{r}), \upsilon(\tilde{r}), \varphi(\tilde{r}), \pi(\tilde{r})\rangle$ *dois rótulos associados respectivamente aos caminhos* r *e* $\tilde{r}$ *que terminam no vértice* i*. Se (a)* $d^r \leq d^{\tilde{r}}$*, (b)* $\eta^r \leq \eta^{\tilde{r}}$*, (c)* $\upsilon^j(r) \leq \upsilon^j(\tilde{r}), \forall j \in \Gamma_1^-$*, (d)* $\varphi^j(r) \leq \varphi^j(\tilde{r}), \forall j \in \Gamma_2^+$*, (e)* $\pi^j(r) \leq \pi^j(\tilde{r})$*,* $\forall j \in \Gamma_3^+$ *e (f)* $\phi(r) \leq \phi(\tilde{r})$*, então* l_r *domina* $l_{\tilde{r}}$*.*

Demonstração. Assuma que $\hat{\omega}(u, q) = \{a \in \omega(u) \setminus \{1\} | \pi^a(q) = 0\}$ denota o conjunto de todos os vértices cobertos pelo vértice u que não são cobertos por uma rota q. Além disso, assuma que $\hat{\omega}^+(u, q) = \{a \in \omega(u) \cap \Gamma_3^+ \setminus \{1\} | \pi^a(q) = 0\}$ denota o conjunto de todos os vértices com custo dual positivo cobertos pelo vértice u que não são cobertos por uma rota q. Seja um vértice $j \in V$, uma vez que $\pi^u(r) \leq \pi^u(\tilde{r})$, $\forall u \in \Gamma_3^+$, tem-se $\hat{\omega}^+(j, \tilde{r}) \subseteq \hat{\omega}^+(j, r)$ e, como resultado, $\sum_{u \in \hat{\omega}(j,r)} \bar{\gamma}_u^3 \geq \sum_{u \in \hat{\omega}(j,\tilde{r})} \bar{\gamma}_u^3$. As possíveis extensões válidas são analisadas a seguir.

Caso 1: Considere $j \in \Gamma_1^-$, $\upsilon^j(\tilde{r}) = 0$ ou $j \notin \Gamma_1^-$, $\upsilon^j(\tilde{r}) = 0$. A extensão dos rótulos l_r e $l_{\tilde{r}}$ ao vértice j em termos de custo resulta em $\phi(r) + d_{ij} + \bar{\gamma}_j^1 - \sum_{u \in \hat{\omega}(j,r)} \bar{\gamma}_u^3 \leq \phi(\tilde{r}) + d_{ij} + \bar{\gamma}_j^1 - \sum_{u \in \hat{\omega}(j,\tilde{r})} \bar{\gamma}_u^3$, uma vez que $\sum_{u \in \hat{\omega}(j,r)} \bar{\gamma}_u^3 \geq \sum_{u \in \hat{\omega}(j,\tilde{r})} \bar{\gamma}_u^3$. Portanto, o rótulo $l_{\tilde{r}}$ pode ser eliminado, já que sua extensão resulta em um caminho com custo sempre maior ou igual àquele resultante da extensão de l_r.

Caso 2: Considere $j \in \Gamma_2^+$, $\varphi^j(\tilde{r}) = 0$ ou $j \notin \Gamma_2^+$, $\varphi^j(\tilde{r}) = 0$. A extensão dos rótulos l_r e $l_{\tilde{r}}$ ao vértice j em termos de custo resulta em $\phi(r) + d_{ij} - \bar{\gamma}_j^2 - \sum_{u \in \hat{\omega}(j,r)} \bar{\gamma}_u^3 \leq \phi(\tilde{r}) + d_{ij} - \bar{\gamma}_j^2 - \sum_{u \in \hat{\omega}(j,\tilde{r})} \bar{\gamma}_u^3$, uma vez que $\sum_{u \in \hat{\omega}(j,r)} \bar{\gamma}_u^3 \geq \sum_{u \in \hat{\omega}(j,\tilde{r})} \bar{\gamma}_u^3$. Portanto, o rótulo $l_{\tilde{r}}$ pode ser eliminado, já que sua extensão resulta em um caminho com custo sempre maior ou igual àquele resultante da extensão de l_r.

Caso 3: Considere $j \in V \setminus T$, $j \notin \Gamma_1^-$, $\upsilon^j(r) = 1$. A extensão de $l_{\tilde{r}}$ ao vértice j resulta em $\phi(\tilde{r}) + d_{ij} + \bar{\gamma}_j^1 - \sum_{u \in \hat{\omega}(j,\tilde{r})} \bar{\gamma}_u^3 \geq \phi(r)$, já que $\upsilon^j(r) = 1$ e $\sum_{u \in \hat{\omega}(j,\tilde{r})} \bar{\gamma}_u^3 \leq 0$ porque $\pi^u(r) \leq \pi^u(\tilde{r})$, $\forall u \in \Gamma_3^+$. Portanto, o rótulo $l_{\tilde{r}}$ pode ser eliminado, já que sua extensão resulta em um caminho com custo sempre maior ou igual ao rótulo l_r.

Caso 4: Considere $j \in T \setminus \{1\}$, $j \notin \Gamma_2^+$, $\varphi^j(r) = 1$. A extensão de $l_{\tilde{r}}$ ao vértice j resulta em $\phi(\tilde{r}) + d_{ij} - \bar{\gamma}_j^2 - \sum_{u \in \hat{\omega}(j,\tilde{r})} \bar{\gamma}_u^3 \geq \phi(r)$, já que $\varphi^j(r) = 1$ e $\sum_{u \in \hat{\omega}(j,\tilde{r})} \bar{\gamma}_u^3 \leq 0$ porque $\pi^u(r) \leq \pi^u(\tilde{r})$, $\forall u \in \Gamma_3^+$. Portanto, o rótulo $l_{\tilde{r}}$ pode ser eliminado, já que sua extensão resulta em um caminho com custo sempre maior ou igual ao rótulo l_r. □

Proposição 5. *Sejam l_r um rótulo associado a um caminho r que termina no vértice i e um vértice $j \in V \setminus \{1\}$ para o qual a extensão de l_r seja válida. Define-se $\omega(j)^+ = \{v \in \omega(j) : \bar{\gamma}_v^3 > 0\}$ como o conjunto de vértices cobertos por j que possuem custo dual positivo. Se $j \in V \setminus T$, $\bar{\gamma}_j^1 \geq 0$ ou $j \in T \setminus \{1\}, \bar{\gamma}_j^2 \leq 0$, e $\pi^v(r) = 1$, $\forall v \in \omega(j)^+$, então a extensão de l_r para j pode ser podada.*

Demonstração. Suponha que $\tilde{r}$ seja uma rota tal que r seja seu subcaminho e as arestas $\{i,j\}$ e $\{j,l\}$ sejam utilizadas, sendo l o vértice visitado imediatamente após o vértice j. Seja $\hat{r}$ a rota obtida a partir de $\tilde{r}$ pela remoção do vértice j, de modo a remover as arestas $\{i,j\}$ e $\{j,l\}$, e reconectar o caminho por meio da aresta $\{i,l\}$. Com base na propriedade da desigualdade triangular e no fato de $\pi^v(r) = 1$, $\forall v \in \omega(j)^+$, temos que:

Se $j \in V \setminus T$ e $\gamma_j^1 \geq 0$: $\phi(\hat{r}) = \phi(\tilde{r}) - d_{ij} - d_{jl} + d_{il} - \bar{\gamma}_j^1 \leq \phi(\tilde{r})$.

Se $j \in T \setminus \{1\}$ e $\gamma_j^2 \leq 0$: $\phi(\hat{r}) = \phi(\tilde{r}) - d_{ij} - d_{jl} + d_{il} + \bar{\gamma}_j^2 \leq \phi(\tilde{r})$.

Portanto, a extensão de l_r para j pode ser podada por conduzir a um caminho com custo maior ou igual àquele em que o vértice j não é visitado. □

O algoritmo implementado tem sua execução interrompida quando a primeira rota com custo reduzido negativo for encontrada. Experimentos computacionais pre-

liminares indicaram que o uso dessa estratégia resulta na diminuição do tempo total gasto com a resolução dos problemas de precificação.

3.4.2 Algoritmo Heurístico

O algoritmo heurístico implementando foi aquele discutido na Seção 2.4.2. A única diferença do algoritmo aqui discutido para aquele é que a vizinhança 2-OPT, já discutida anteriormente, foi considerada na fase de busca local, totalizando, então, cinco vizinhanças na fase de busca local.

3.5 Heurística de Geração de Colunas

A fim de obter soluções viáveis (limites superiores) de boa qualidade, é proposta uma HGC para o PCMV. Seja $\bar{R}$ o conjunto de colunas disponíveis quando o PML for resolvido à otimalidade ou quando o algoritmo de Geração de Colunas for interrompido por atingir o limite de tempo imposto. A ideia básica dessa heurística é resolver por meio de um algoritmo *Branch-and-Bound* o problema definido por (3.1)-(3.6), em que o conjunto R é substituído por $\bar{R}$. A solução ótima para este problema é uma solução viável para o PCMV e pode ser calculada rapidamente por haver um pequeno número de colunas em $\bar{R}$.

3.6 Algoritmo *Branch-and-Price*

O algoritmo BP implementado inicialmente resolve a relaxação linear do problema definido por (3.1)-(3.6) por meio do algoritmo de Geração de Colunas proposto. Se a solução em mãos para a relaxação linear for inteira, então o problema original foi resolvido e o BP é encerrado. Caso contrário, deve-se recorrer a alguma estratégia de ramificação (do inglês, *branching*), que é um dos aspectos mais importantes em um BP. Duas regras de ramificação foram implementadas e são discutidas a seguir.

A primeira regra consiste em realizar ramificação em um vértice $i \in V \setminus T$ tal que $0 < y_i < 1$, em que $y_i := \sum_{r \in \tilde{R}} \beta_i^r \lambda_r$. Em seguida, dois subproblemas são criados: um problema em que o vértice não pode ser visitado ($y_i = 0$) e um outro em que o vértice é visitado uma única vez ($y_i = 1$). Essa regra de ramificação é implementada pela modificação da desigualdade (3.3) associada ao vértice i e, no caso de $y_i = 0$, deve-se tanto remover de $\tilde{R}$ todas as rotas que visitam i como remover o vértice i do grafo de entrada do algoritmo de precificação.

A segunda regra consiste em realizar ramificação no arco (i, j), que é apenas uma orientação da aresta $\{i, j\} \in E$, tal que $0 < x_{ij} < 1$, em que $x_{ij} := \sum_{r \in \tilde{R}} \theta_{ij}^r \lambda_r$. Em seguida, dois subproblemas são criados: um problema em que o arco não pode pertencer à solução ($x_{ij} = 0$) e um outro em que o arco é forçado a fazer parte da solução ($x_{ij} = 1$). No caso em que o arco não pode fazer parte da solução, deve-se remover de $\tilde{R}$ todas as rotas que utilizam este arco e remover este arco do grafo de entrada do algoritmo de precificação. No caso em que deve-se forçar o arco a fazer parte da solução, arcos (i, k) com $k \neq j$ ou (k, j) com $k \neq i$ devem ser removidos do grafo de entrada do algoritmo de precificação e rotas que utilizem esses arcos devem ser removidas de $\tilde{R}$.

É importante notar que as duas regras de ramificação implementadas não invalidam a regra de dominância nem a regra de poda propostas. Ainda, em ambas as regras de ramificação, a variável mais fracionária foi aquela escolhida. Por fim, após a aplicação da regra de ramificação, a heurística HGC é executada para obter uma solução primal no nó considerado.

A escolha do nó a ser considerado, dentre aqueles ativos na árvore de enumeração, foi feita com base no nó com o menor limite inferior (estratégia de busca em largura). Em experimentos preliminares, o BP obteve um desempenho melhor com esta estratégia em relação àquela de busca em profundidade.

3.7 Experimentos Computacionais

Nesta seção, são apresentados os resultados dos experimentos computacionais realizados para avaliar os algoritmos propostos. Ainda, o ambiente computacional utilizado e a metodologia adotada para gerar as instâncias são apresentados.

3.7.1 Ambiente Computacional e Metodologia

Nos experimentos realizados, as instâncias consideradas foram geradas de acordo com a metodologia proposta em Hachicha et al. [2000]. Essa metodologia propõe a geração de $|V| + |W|$ pontos em uma região quadrada, definida por $[0, 100] \times [0, 100]$, sendo que o depósito deve ser gerado em uma região quadrada, definida por $[25, 75] \times [25, 75]$. Os pontos devem ser gerados com base em uma distribuição contínua uniforme de números aleatórios. O conjunto V é composto pelos $|V|$ primeiros pontos gerados, dos quais os $|T|$ primeiros pontos são atribuídos ao conjunto T; os demais são atribuídos ao conjunto W. Por fim, o valor de c foi definido como $c = \max\{\max_{h \in V \setminus T} \min_{l \in W}\{d_{lh}\}, \max_{l \in W}\{d_{lh(l)}\}\}$, em que h(l) é o segundo vértice em

$V \setminus T$ mais próximo do vértice l. Deste modo, cada vértice em $V \setminus T$ cobre ao menos um vértice em W e cada vértice de W é coberto por no mínimo dois vértices de $V \setminus T$. Apesar de os autores terem proposto que d_{ij} é definido como a distância euclidiana entre os vértices em questão, neste trabalho, d_{ij} foi definido como o menor número inteiro maior ou igual à distância euclidiana entre i e j. Foram geradas instâncias com as seguintes configurações $|V| = 50, 100, 200$, $|W| = 50, 100, 200$, $|T| = 1, \lceil |V|/4 \rceil + 1, \lceil |V|/2 \rceil + 1$, $p = 4, \infty$ e $q = 200, \infty$. Por fim, o parâmetro m foi configurado para $\lceil |V|/p \rceil$ quando $p = 4$ ou para ∞ quando $p = \infty$.

Para gerar o conjunto de colunas iniciais, foi implementado um algoritmo baseado na metaheurística GRASP. A fase construtiva utiliza-se do algoritmo CIA para construir um conjunto de no máximo m rotas, de modo que todos os vértices em T sejam visitados e todos aqueles em W sejam cobertos. Vale ressaltar que a restrição de capacidade associada a q foi relaxada na fase construtiva; tenta-se viabilizar a solução na fase de busca local do algoritmo. A fase de buca local consiste na aplicação das vizinhanças apresentadas na Seção 2.4.2 em conjunto com a vizinhança 2-OPT, a qual é essencial para tentar satisfazer a restrição de capacidade associada a q. Caso a solução construída seja inviável, esta é desconsiderada. Vale ressaltar que esta heurística foi implementada apenas para obter uma solução básica inicial para o algoritmo de Geração de Colunas, portanto nenhuma análise é feita acerca da qualidade das soluções providas nem de seu desempenho.

Os experimentos foram executados em uma máquina Intel Xeon E5645 64 bits com 2.40GHz de processador e com 32Gb de memória RAM, rodando o sistema operacional Ubuntu 12.04. Os algoritmos propostos foram implementados em C++ e compilados com o compilador GCC 4.4.3. O ILOG CPLEX versão 12.1 foi utilizado apenas como resolvedor de Programação Linear. Por fim, um limite de 4 horas de execução foi imposto ao BP.

3.7.2 Resultados Computacionais

As Tabelas 3.1 e 3.2 apresentam os resultados computacionais para as instâncias e configurações consideradas. As três primeiras colunas apresentam a quantidade de vértices nos conjuntos T, V e W, respectivamente. As quatro colunas seguintes apresentam respectivamente o limite de relaxação linear (lb), o tempo gasto em segundos, o número de iterações do algoritmo de Geração de Colunas e o limite superior obtido pela HGC na raiz da árvore de enumeração. Por fim, as cinco últimas colunas apresentam respectivamente o melhor limite inferior (blb), o melhor limite superior (bub), o *gap* de dualidade, o tempo gasto em segundos e, por fim, o número de nós analisados pelo BP.

A Tabela 3.1 apresenta os resultados computacionais obtidos quando $p = 4$ e $q = 200$. Dentre as 15 instâncias consideradas, o BP foi capaz de atestar a otimalidade de 5 delas. Para 5 das instâncias, o algoritmo nem sequer foi capaz de calcular o limite de relaxação linear dentro do limite de tempo imposto. Para aquelas em que foi possível calcular o limite de relaxação linear, os *gaps* de dualidade obtidos foram inferiores a 3.7%. Além disso, nota-se que o limite primal obtido pela HGC na raiz foi igual ao melhor limite superior encontrado pelo BP em todas as instâncias exceto uma, em que o melhor limite superior fornecido pelo BP foi menor por uma unidade do que aquele encontrado na raiz.

			raiz				BP				
|T|	|V|	|W|	lb	t(s)	#it	CGH	blb	bub	gap(%)	t(s)	#nodes
1	50	50	328.0	1	18	328	328	328	0	1	1
1	50	100	310.0	2	19	310	310	310	0	2	1
1	100	100	486.0	72	54	501	493	501	1.6	-	321
1	100	200	597.7	3023	108	604	604	604	0	11263	5
1	200	200	-	-	200	723	-	723	-	-	0
13	50	50	567.8	11	38	571	571	571	0	708	221
13	50	100	535.0	6	28	544	543	543	0	4998	2230
26	100	100	1019.8	2194	89	1064	1024.5	1064	3.7	-	29
26	100	200	-	-	124	1289	-	1289	-	-	0
51	200	200	-	-	153	1726	-	1726	-	-	0
26	50	50	875.5	66	63	905	882	905	2.5	-	1035
26	50	100	814.2	12	42	846	821.4	846	2.9	-	2545
51	100	100	1455.5	3109	117	1485	1455.5	1485	2.0	-	5
51	100	200	-	-	103	1818	-	1818	-	-	0
101	200	200	-	-	165	2766	-	2766	-	-	0

Tabela 3.1: Resultados computacionais com $p = 4$, $q = 200$.

A Tabela 3.2 apresenta os resultados computacionais obtidos quando $p = \infty$ e $q = \infty$. O BP foi capaz de atestar a otimalidade de 4 dentre as 15 instâncias consideradas. Para as demais instâncias, o algoritmo não foi capaz de calcular o limite de relaxação linear dentro do limite de tempo imposto.

A Tabela 3.3 apresenta os resultados computacionais para a raiz do BP para as instâncias consideradas. As duas primeiras colunas apresentam respectivamente os valores dos parâmetros $|V|$ e $|W|$. As três colunas seguintes apresentam os valores dos parâmetros p, q e $|T|$, respectivamente. A quinta coluna apresenta o tempo de execução em segundos ou o símbolo "-" para indicar que o algoritmo teve de ser interrompido por ter atingido o limite de tempo imposto. A sexta coluna apresenta a porcentagem do tempo total gasto para resolver os problemas de precificação. A sétima coluna apresenta o número total de iterações do algoritmo. A oitava coluna apresenta a porcentagem de iterações degeneradas, no sentido de que o valor da função objetivo

			raiz				BP				
|T|	|V|	|W|	lb	t(s)	#it	CGH	blb	bub	gap(%)	t(s)	#nodes
1	50	50	272.0	1	23	272	272	272	0	1	1
1	50	100	271.0	4	46	271	271	271	0	4	1
1	100	100	-	-	554	354	-	354	-	-	0
1	100	200	-	-	2093	393	-	393	-	-	0
1	200	200	-	-	3215	457	-	457	-	-	0
13	50	50	349.0	270	257	349	349	349	0	270	1
13	50	100	363.0	6773	920	363	363	363	0	6773	1
26	100	100	-	-	10216	507	-	507	-	-	0
26	100	200	-	-	10077	528	-	528	-	-	0
51	200	200	-	-	25418	653	-	653	-	-	0
26	50	50	-	-	824	475	-	475	-	-	0
26	50	100	-	-	2364	456	-	456	-	-	0
51	100	100	-	-	9021	562	-	562	-	-	0
51	100	200	-	-	8031	689	-	689	-	-	0
101	200	200	-	-	17556	889	-	889	-	-	0

Tabela 3.2: Resultados computacionais com $p = \infty$ e $q = \infty$.

se manteve inalterado em relação à iteração anterior. Por fim, a última coluna apresenta a porcentagem de rótulos dominados e podados pelo algoritmo de Programação Dinâmica. Uma discussão acerca dos dados apresentados nesta tabela é feita a seguir.

O algoritmo foi capaz de calcular o limite de relaxação linear de 14 das 30 (46.66%) instâncias consideradas. Observa-se que a porcentagem de tempo gasto resolvendo o problema de precificação foi superior a 90% em todas as instâncias exceto duas. Nota-se um aumento considerável do número de iterações e, consequentemente, no tempo despendido quando p e q tornaram-se ilimitados. A porcentagem de iterações degeneradas variou entre 29.03 e 99.99%. Por fim, nota-se que a porcentagem de rótulos dominados ou podados pelas regras de dominância e de poda foi inferior a 7.53%. A seguir, é apresentada uma análise da porcentagem de variáveis básicas degeneradas ao longo do algoritmo para os cenários e casos analisados.

As Figuras 3.1-3.3 apresentam a porcentagem de variáveis básicas degeneradas ao longo das iterações do algoritmo de Geração de Colunas na raiz da árvore de enumeração. A cada cinco iterações, a porcentagem de variáveis básicas degeneradas foi calculada. Por conta do grande número de iterações em algumas instâncias e um pequeno número em outras, as figuras apresentam a porcentagem de degeneração de no máximo dez iterações. A saber, quando possível, apresenta-se a porcentagem referente à primeira e à última iteração juntamente outras oito iterações intermediárias.

A Figura 3.1 apresenta a porcentagem de variáveis básicas degeneradas ao longo das iterações do algoritmo para os cenários considerados quando $|V| = 50$ e $|W| = 50$. Observa-se um comportamento irregular quando $p = 4$ e $q = 200$ e uma elevada

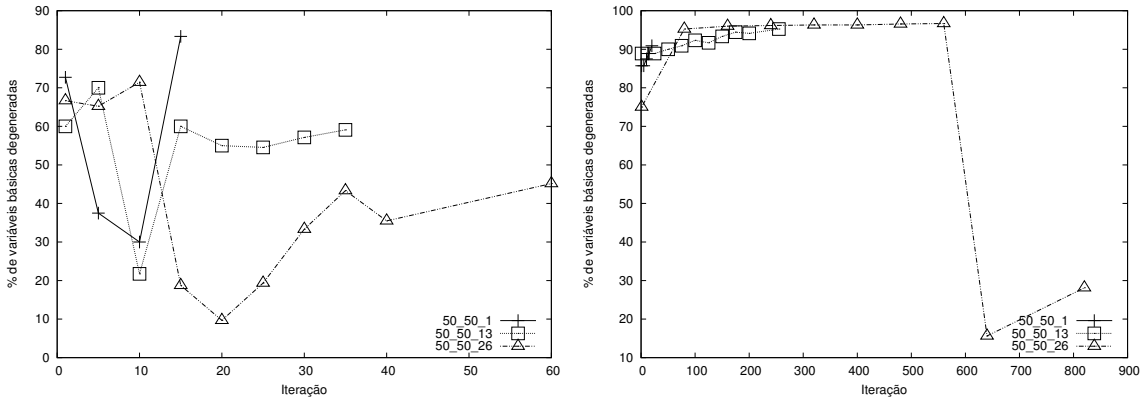
|V|	|W|	p	q	|T|	t(s)	%tp	#iter	%iter	%dom
50	50	4	200	1	1.04	96.15	18	55.56	6.09
				13	10.80	99.81	38	60.53	3.88
				26	66.38	99.92	63	49.21	5.08
		∞	∞	1	1.18	98.31	23	95.65	3.28
				13	270.24	99.93	257	99.61	2.62
				26	-	99.99	824	83.98	3.17
50	100	4	200	1	1.78	99.44	19	63.16	5.79
				13	5.62	99.82	28	53.57	3.47
				26	11.93	99.66	42	42.86	3.88
		∞	∞	1	3.60	98.89	46	86.96	2.46
				13	6773.36	99.95	920	96.41	2.03
				26	-	99.89	2364	99.96	1.65
100	100	4	200	1	72.46	99.92	54	62.96	4.37
				26	2194.43	99.99	89	32.58	3.31
				51	3109.23	99.99	117	56.41	2.77
		∞	∞	1	-	99.97	554	99.28	2.75
				26	-	95.74	10216	97.19	2.88
				51	-	96.56	9021	99.99	0.87
100	200	4	200	1	3023.20	99.99	108	67.59	7.53
				26	-	99.99	124	29.03	5.08
				51	-	99.99	103	44.66	2.91
		∞	∞	1	-	99.40	2093	97.99	2.64
				26	-	94.52	10077	99.99	2.78
				51	-	94.08	8031	91.05	2.29
200	200	4	200	1	-	99.99	200	45.50	5.45
				51	-	99.99	153	32.03	3.32
				101	-	99.99	165	36.36	2.11
		∞	∞	1	-	95.08	3215	98.79	2.13
				51	-	56.98	25418	99.99	1.53
				101	-	63.98	17556	99.99	1.01

Tabela 3.3: Resultados computacionais para as instâncias PCMV.

porcentagem de degeneração quando $p = \infty$ e $q = \infty$ ao longo das iterações. Vale ressaltar que a porcentagem de degeneração se manteve acima de 70% em quase todas as iterações quando $p = \infty$ e $q = \infty$.

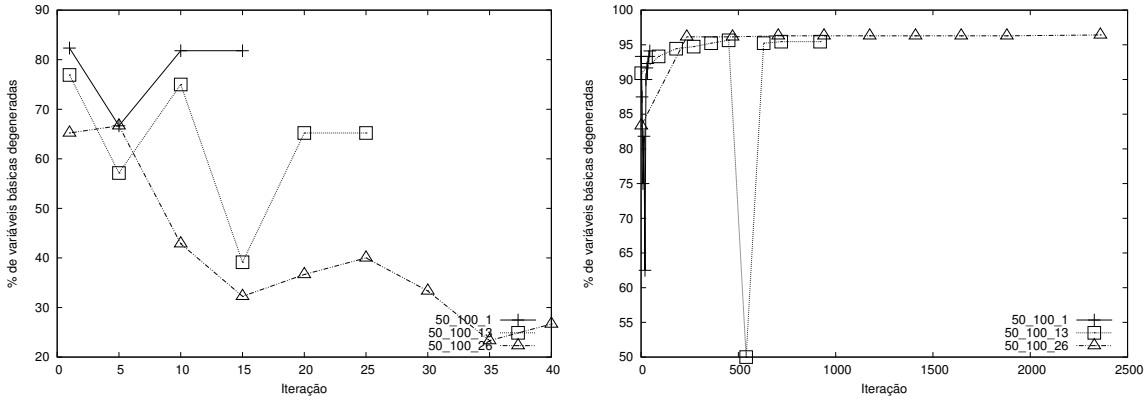
A Figura 3.2 apresenta a porcentagem de variáveis básicas degeneradas ao longo das iterações do algoritmo para os cenários considerados quando $|V| = 50$ e $|W| = 100$. No cenário em que $p = 4$ e $q = 200$, observa-se um comportamento irregular em todos os casos avaliados e a porcentagem de degeneração se manteve abaixo de 90% na maioria das iterações. Por sua vez, quando $p = \infty$ e $q = \infty$, percebe-se uma elevada porcentagem de degeneração para todos os casos avaliados na maioria das iterações.

A Figura 3.3 apresenta a porcentagem de variáveis básicas degeneradas ao longo das iterações do algoritmo para os cenários considerados quando $|V| = 100$ e $|W| = 100$. No cenário em que $p = 4$ e $q = 200$, observa-se um comportamento irregular em todos os casos avaliados e a porcentagem de degeneração se manteve abaixo de 80% em



(a) p=4, q=200 (b) p=∞, q=∞

Figura 3.1: Porcentagem de variáveis básicas degeneradas ao longo das iterações quando $|V| = 50$ e $|W| = 50$.



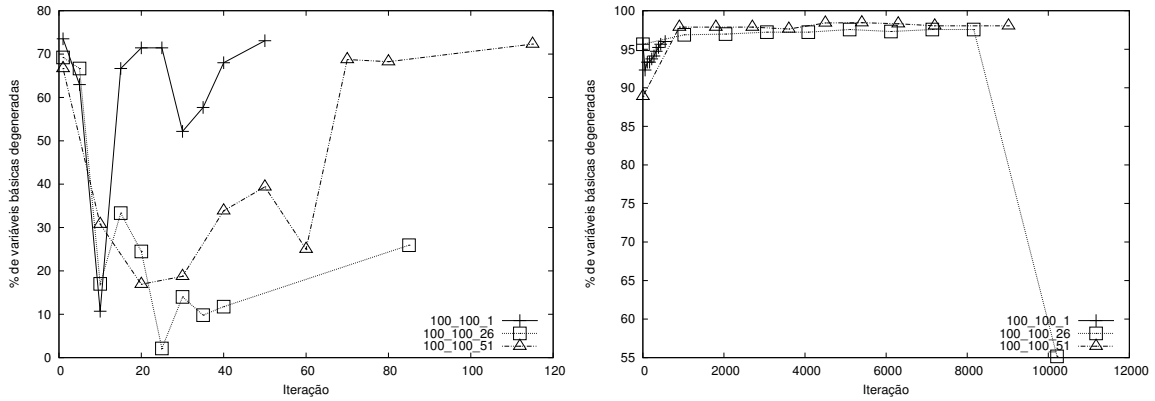
(a) p=4, q=200 (b) p=∞, q=∞

Figura 3.2: Porcentagem de variáveis básicas degeneradas ao longo das iterações quando $|V| = 50$ e $|W| = 100$.

todas as iterações. Quando $p = \infty$ e $q = \infty$, percebe-se uma elevada porcentagem de degeneração e um comportamento regular ao longo das iterações.

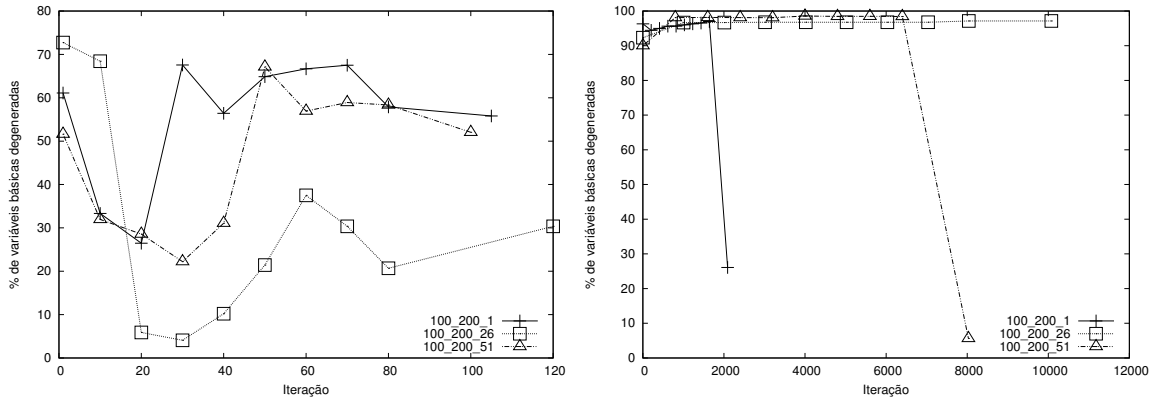
A Figura 3.4 apresenta a porcentagem de variáveis básicas degeneradas ao longo das iterações do algoritmo para os cenários considerados quando $|V| = 100$ e $|W| = 200$. No cenário em que $p = 4$ e $q = 200$, observa-se um comportamento irregular em todos os casos avaliados e a porcentagem de degeneração se manteve abaixo de 80% na maioria das iterações. Quando $p = \infty$ e $q = \infty$, percebe-se uma elevada porcentagem de degeneração e um comportamento regular ao longo das iterações.

A Figura 3.5 apresenta a porcentagem de variáveis básicas degeneradas ao longo das iterações do algoritmo para os cenários considerados quando $|V| = 200$ e $|W| = 200$.



(a) p=4, q=200 (b) p=∞, q=∞

Figura 3.3: Porcentagem de variáveis básicas degeneradas ao longo das iterações quando $|V| = 100$ e $|W| = 100$.



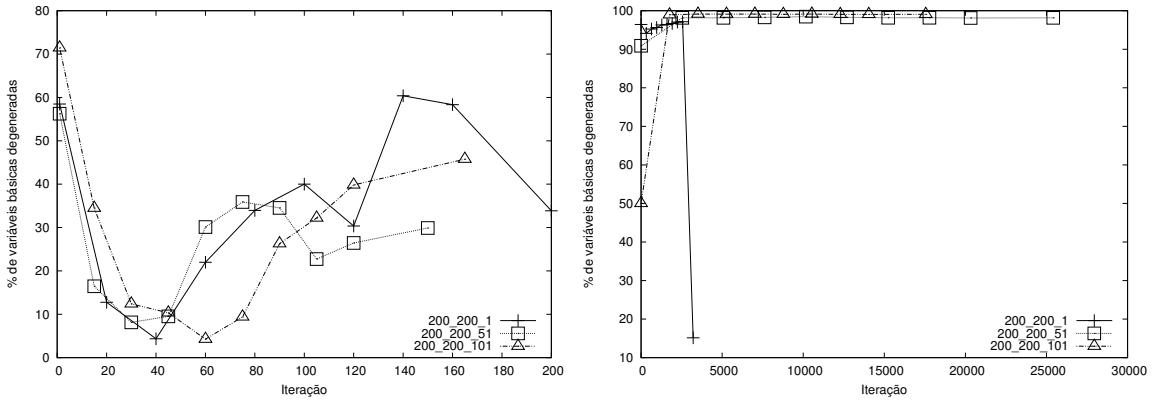
(a) p=4, q=200 (b) p=∞, q=∞

Figura 3.4: Porcentagem de variáveis básicas degeneradas ao longo das iterações quando $|V| = 100$ e $|W| = 200$.

No cenário em que $p = 4$ e $q = 200$, observa-se um comportamento irregular em todos os casos avaliados e a porcentagem de degeneração se manteve abaixo de 70% em todas as iterações. Quando $p = \infty$ e $q = \infty$, percebe-se uma elevada porcentagem de degeneração e um comportamento regular ao longo das iterações.

3.8 Conclusões

Neste capítulo, foi estudado o PCMV. Na literatura, é encontrado um único trabalho que propõe três heurísticas para este problema. Ainda, é encontrado um trabalho que tenta propor um algoritmo exato, mas, como já discutido, esse trabalho diverge da



(a) p=4, q=200 (b) p=∞, q=∞

Figura 3.5: Porcentagem de variáveis básicas degeneradas ao longo das iterações quando $|V| = 200$ e $|W| = 200$.

definição original do PCMV. Portanto, com base na revisão bibliográfica realizada, é desconhecida a existência de algum algoritmo exato para o PCMV. A fim de suprir essa carência, neste trabalho, foi proposto um algoritmo exato para o PCMV.

Este trabalho propõe uma reformulação por recobrimento de conjuntos, um algoritmo de Geração de Colunas, um BP e uma HGC para o PCMV. Para resolução do problema de precificação, foram implementados um algoritmo exato e um heurístico. A saber, um algoritmo de Programação Dinâmica foi implementado, sendo que uma regra de dominância e uma regra de poda foram propostas a fim de acelerar a resolução do problema de precificação. Experimentos computacionais com trinta instâncias foram conduzidos para avaliar o desempenho do BP. Ainda, foi realizado um estudo do desempenho do algoritmo de Geração de Colunas na raiz da árvore de enumeração.

O BP foi capaz de atestar a otimalidade de 9 das 30 (30%) instâncias consideradas dentro de quatro horas de execução. Para outras 5, este algoritmo foi capaz de calcular o limite de relaxação linear, de modo que os *gaps* de dualidade encontrados foram menores ou iguais a 3.7%. Por fim, para 16 das 30 instâncias, o algoritmo não foi capaz de calcular o limite de relaxação linear. Com base no estudo feito acerca do desempenho do algoritmo de Geração de Colunas na raiz da árvore de enumeração, indubitavelmente, o gargalo do algoritmo reside na resolução do problema de precificação, que consumiu mais de 90% em todas as instâncias consideradas exceto duas. Associado a isso, a degenerescência também contribuiu para a degradação do desempenho do algoritmo. Por fim, menos de 8% dos rótulos analisados no algoritmo de Programação de Dinâmica foram dominados ou podados. Em termos de limites primais, nota-se que a HGC obteve soluções muito próximas da solução ótima, quando esta foi encontrada.

Com base no estudo realizado neste trabalho, observa-se que o BP provê limites de relaxação linear de boa qualidade, embora um tempo considerável seja despendido para calculá-los na maioria das instâncias. Para contornar esse cenário, fazem-se necessárias técnicas para tornar o problema de precificação melhor resolvido na prática. Por exemplo, podem ser encontradas novas regras de dominância e de poda, como também pode-se encontrar uma relaxação válida para o problema de precificação que possa ser computada por algum algoritmo pseudo-polinomial.

Capítulo 4

Considerações Finais

Neste trabalho, duas variações do Problema de Roteamento de Veículos foram estudadas. Os problemas aqui considerados combinam duas características (seletividade e cobertura) não muito comuns na literatura. No primeiro problema, denominado Problema de Roteamento de Veículos Seletivo Min-Max, busca-se um conjunto de rotas em que o comprimento da maior rota seja minimizado, de modo que todos os vértices sejam cobertos. No segundo, intitulado Problema de Cobertura Multi-Veículo, busca-se um conjunto de no máximo m rotas em que a soma da distância de cada rota seja minimizada, de modo a cobrir e a visitar todos os vértices especificados.

Para o primeiro problema, um algoritmo de Geração de Colunas e duas heurísticas foram propostas. Os métodos propostos não se mostraram eficientes para as instâncias consideradas, de modo que não se justificou a implementação de um algoritmo *Branch-and-Price*. Para o segundo, um algoritmo *Branch-and-Price* e uma heurística foram propostas, por meio dos quais algumas instâncias foram resolvidas à otimalidade.

Estabelecendo-se uma comparação entre os métodos propostos para os dois problemas, percebe-se que, apesar de as técnicas empregadas para resolver os dois problemas terem o mesmo fundamento, o algoritmo de Geração de Colunas para o segundo obteve um desempenho melhor em relação àquele proposto para o primeiro. Esse fato pode ser justificado pela inexistência da natureza min-max no segundo problema. Além disso, nota-se que, em geral, as porcentagens de iterações degeneradas no segundo problema foram menores em relação àquelas obtidas no primeiro problema, sugerindo que o segundo problema sofreu menos do efeito negativo da degenerescência.

As principais contribuições deste trabalho são o estudo do impacto da variação da densidade e do número de veículos nos algoritmos propostos para o PRVSMM, e a proposição do primeiro algoritmo exato para o PCMV.

Referências Bibliográficas

(2012). ILOG CPLEX. http://www.ilog.com/products/cplex/, Visitado em: 17 de setembro de 2012.

(2012). OpenMP. http://www.openmp.org. Visitado em: 17 de setembro de 2012.

(2012). TSPLIB. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/. Visitado em: 17 de setembro de 2012.

Akyildiz, I.; Su, W.; Sankarasubramaniam, Y. & Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422.

Baldacci, R.; Dell'Amico, M. & González, J. S. (2007). The capacitated m-ring-star problem. *Operations Research*, 55(6):1147–1162.

Bard, J. F.; Huang, L.; Dror, M. & Jaillet, P. (1998). A branch and cut algorithm for the VRP with satellite facilities. *IIE Transactions*, 30:821–834.

Barnhart, C.; Johnson, E. L.; Nemhauser, G. L.; Savelsbergh, M. W. P. & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329.

Benavent, E.; Corberán, A.; Plana, I. & Sanchis, J. M. (2009). Min-max k-vehicles windy rural postman problem. *Networks*, 54(4):216–226.

Boland, N.; Dethridge, J. & Dumitrescu, I. (2006). Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34(1):58–68.

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

Camerini, P. M.; Fratta, L. & Maffioli, F. (1975). On improving relaxation methods by modified gradient techniques. Em *Nondifferentiable Optimization*, volume 3 of *Mathematical Programming Studies*, pp. 26–34. Springer Berlin Heidelberg.

Croes, A. (1958). A method for solving traveling salesman problems. *Operations Research*, 5:791–812.

Dantzig, G.; Fulkerson, R. & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410.

Dantzig, G. B. & Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.

Dror, M. (1994). Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42:977–978.

Edmonds, J. (1965). Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467.

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

Feo, T. A. & Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

Fischetti, M. & Lodi, A. (2003). Local branching. *Mathematical Programming*, 98:23–47.

Glaab, H. (2002). A new variant of a vehicle routing problem: Lower and upper bounds. *European Journal of Operational Research*, 139(3):557–577.

Goetschalckx, M. & Jacobs-Blecha, C. (1989). The vehicle routing problem with backhauls. *European Journal of Operational Research*, 42(1):39–51.

Guignard, M. (2003). The lagrangian relaxation. *Top*, 11(2):151–228.

Hachicha, M.; Hodgson, M. J.; Laporte, G. & Semet, F. (2000). Heuristics for the multi-vehicle covering tour problem. *Computers & Operations Research*, 27(1):29–42.

Held, M.; Wolfe, P. & Crowder, H. P. (1974). Validation of subgradient optimization. *Mathematical Programming*, 6:62–88.

Jozefowiez, N. (2012). Column generation for the multiple vehicle covering tour problem. Em *Book of Extended Abstracts of 5th International Workshop on Freight Transportation and Logistics*, pp. 534–537.

Julstrom, B. A. (1999). Coding TSP tours as permutations via an insertion heuristic. Em *Proceedings of the 1999 ACM symposium on Applied computing*, SAC '99, pp. 297–301. ACM.

Labbé, M.; Laporte, G.; Martín, I. R.; José, J. & González, S. (2004). The ring star problem: polyhedral analysis and exact algorithm. *Networks*, 43:177–189.

Laporte, G.; Desrochers, M. & Nobert, Y. (1984). Two exact algorithms for the distance-constrained vehicle routing problem. *Networks*, 14(1):161–172.

Laporte, G.; Louveaux, F. V. & van Hamme, L. (2002). An integer l-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50(3):415–423.

Lourenço, H. R.; Martin, O. & Stützle, T. (2002). Iterated local search. Em Glover, F. & Kochenberger, G., editores, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pp. 321–353. Kluwer Academic Publishers, Norwell, MA.

Lübbecke, M. & Desrosiers, J. (2004). Selected topics in column generation. *Operations Research*, 53:1007–1023.

Mladenovic, N. & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Pacheco, P. S. (2011). *An Introduction to Parallel Programming*. Elsevier.

Righini, G. & Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273.

Rousseau, L.-M.; Gendreau, M. & Feillet, D. (2006). Interior point stabilization for column generation. *Operations Research Letters*, 35(5):660–668.

Savelsbergh, M. W. P. & Sol, M. (1995). The general pickup and delivery problem. *Transportation Science*, 29:17–29.

Valle, C. A.; da Cunha, A. S.; Aioffi, W. M. & Mateus, G. R. (2008). Algorithms for improving the quality of service in wireless sensor networks with multiple mobile sinks. Em *Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pp. 239–243. ACM.

Valle, C. A.; da Cunha, A. S.; Mateus, G. R. & Martinez, L. C. (2009). Exact algorithms for a selective vehicle routing problem where the longest route is minimized. Em *LAGOS'09 - V Latin-American Algorithms, Graphs and Optimization Symposium*, volume 35, pp. 133–138. Electronic Notes in Discrete Mathematics.

Valle, C. A.; Martinez, L. C.; da Cunha, A. S. & Mateus, G. R. (2011). Heuristic and exact algorithms for a min-max selective vehicle routing problem. *Computers & Operations Research*, 38:1054–1065.

Yakici, E. & Karasakal, O. (2012). A min–max vehicle routing problem with split delivery and heterogeneous demand. *Optimization Letters*, pp. 1–15. 10.1007/s11590-012-0571-8.