UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Monteiro, Jefferson Willian Gouveia

Branch-cut-and-price para o Problema de Roteamento de Veículos Generalizado

Belo Horizonte
2019

Monteiro, Jefferson Willian Gouveia

Branch-cut-and-price para o Problema de Roteamento de Veículos Generalizado

Versão Final

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Geraldo Robson Mateus
Coorientador: Douglas Guimarães Macharet

Belo Horizonte
2019

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6ª Região nº 1510

Monteiro, Jefferson Willian Gouveia.

M775b Branch-cut-and-price para o problema de roteamento de veículos generalizado / Jefferson Willian Gouveia Monteiro — Belo Horizonte, 2019.
xxiv, 58 f. il.; 29 cm.

Dissertação (mestrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação
Orientador: Geraldo Robson Mateus.
Coorientador: Douglas Guimarães Macharet

1. Computação – Teses. 2. Problema de roteamento de veículos – Teses. 3. Algoritmo branch-cut -and-price – Teses.
. I. Orientador. II. Coorientador. III 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

Branch-cut-and-price para o Problema de Roteamento de Veículos Generalizado

JEFFERSON WILLIAN GOUVEIA MONTEIRO

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

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

PROF. DOUGLAS GUIMARÃES MACHARET - Coorientador
Departamento de Ciência da Computação - UFMG

PROF. MAYRON CÉSAR DE OLIVEIRA MOREIRA
Departamento de Ciência da Computação - UFLA

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

Belo Horizonte, 3 de Maio de 2019.

Dedico este trabalho aos meus pais, Gerson e Suely.

Agradecimentos

Ao meu orientador, professor Geraldo Robson Mateus, agradeço por todo ensinamento e confiança desde o tempo da graduação. Obrigado por ter feito parte da minha vida acadêmica, me dizendo sábias palavras sempre com bom humor e tranquilidade.

Ao professor Douglas Guimarães Macharet, agradeço a ativa participação durante o meu mestrado e as diversas discussões sobre robótica.

À minha família: Suely, Gerson e Clara, agradeço por sempre me darem forças para alcançar os meus objetivos. Vocês são a minha maior motivação. Amo vocês.

À minha namorada, Marília, obrigado por sempre estar ao meu lado especialmente durante os momentos mais difíceis. Você também faz parte desta vitória.

Aos amigos do LaPO, agradeço os bons momentos vivenciados e as diversas discussões sobre problemas de maratona de programação.

Agradeço àqueles que, direta ou indiretamente, contribuíram para o desenvolvimento deste trabalho.

Por fim, agradeço também à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), que financiou o meu mestrado.

"*Eu acredito*"
(Torcida Atleticana)

Resumo

Esta dissertação dedica-se ao estudo do Problema de Roteamento de Veículos Generalizado. Dado um conjunto de consumidores particionado em subconjuntos disjuntos, denominados *clusters*, o problema consiste em encontrar um conjunto de rotas, uma para cada veículo, tal que cada rota atenda às seguintes restrições: toda rota deve começar e terminar em um depósito específico; a demanda total dos consumidores atendidos em cada rota não deve exceder uma determinada capacidade; e cada *cluster* deve ter sua demanda atendida em exatamente um consumidor. O objetivo é encontrar um conjunto de rotas — que devem atender às restrições citadas — que minimize a soma dos custos de todas as rotas. Para a resolução do problema, propusemos uma solução exata utilizando um algoritmo *branch-cut-and-price* que explora a elementariedade parcial da rota durante o algoritmo de precificação. A metodologia foi avaliada utilizando instâncias da literatura para o problema, e os resultados foram comparados aos trabalhos já publicados. Os experimentos apresentaram resultados satisfatórios, de tal forma que conseguimos resolver instâncias que estavam em aberto e reduzimos significativamente o tempo de solução da maior parte delas.

Palavras-chave: Problema de Roteamento de Veículos Generalizado, Problema de Roteamento de Veículos, Branch-Cut-And-Price.

Abstract

This work addresses the Generalized Vehicle Routing Problem. Given a set of customers partitioned into disjoints subsets, defined as *clusters*, the problem aim to find a set of routes, one per vehicle, such that each route follows the given constraints: every route starts and ends at a specific depot; the total customers' demand satisfied by a route must not be higher than a predefined capacity; and each cluster must have its own demand satisfied in exactly one customer. The objective is to find a set of routes — such that all already defined constraints are satisfied — that minimizes the total routes cost. In order to solve this problem, we propose an exact solution with a *branch-cut-and-price* algorithm that explores partial elementarity of the route during the pricing algorithm. The methodology of our work is evaluated using instances from the literature and the results were compared with previous work. Our experiments showed satisfactory results, such that we were able to solve opened instances from the literature and we reduced significantly the solution time of most of them.

Keywords: Generalized Vehicle Routing Problem, Vehicle Routing Problem, Branch-Cut-And-Price.

Lista de Figuras

Lista de Tabelas

Capítulo 1

Lista de Símbolos

PRV Problema de Roteamento de Veículos

PRVG Problema de Roteamento de Veículos Generalizado

PCV Problema do Caixeiro Viajante

PCVG Problema do Caixeiro Viajante Generalizado

PCVA Problema do Caixeiro Viajante Assimétrico

PRAC Problema de Roteamento de Arcos Capacitados

PLR Problema de Localização-Roteamento

PIM Programação Inteira Mista

PM Problema Mestre

PMR Problema Mestre Restrito

SPPRC *Shortest Path Problem With Resource Constraint*

ESPPRC *Elementary Shortest Path Problem With Resource Constraint*

LNS *Large Neighborhood Search*

MTZ Miller-Tucker-Zemlin

*

Sumário

Capítulo 2

Introdução

O Problema de Roteamento de Veículos (PRV) é um dos problemas clássicos em otimização combinatória. Ele foi introduzido por [8] e intitulado *The Truck Dispatching Problem*. Nesse problema, deve-se determinar o melhor conjunto de rotas a partir de um depósito central, de tal forma que todas as demandas dos consumidores sejam atendidas sem exceder a capacidade de cada veículo que percorre essas rotas.

Passaram-se mais de 50 anos desde o primeiro trabalho publicado sobre o PRV, e, desde então, foram diversas contribuições científicas ([41]). Isso deve-se ao fato de o PRV ser atraente por sua aplicação prática em diversos problemas de logística, além de sua dificuldade considerável.

Por conta da sua grande relevância prática, são encontradas diversas variações do mesmo problema na literatura que se aproximam de várias aplicações. Algumas das mais relevantes são: (i) PRV com Restrição de Janela de Tempo ([28]), em que cada consumidor deve ser visitado dentro de um intervalo (janela) de tempo específico; (ii) PRV com coleta e entrega ([38]), em que, além de atender à demanda de cada consumidor, deve-se recolher as ofertas em cada fornecedor; (iii) PRV com múltiplos depósitos ([7]), em que rotas podem ser formadas a partir de diferentes depósitos em vez de ficarem restritas a um depósito central; (iv) e PRV estocástico ([22]), em que os dados do problema são estocásticos e seguem uma determinada distribuição de probabilidade.

Neste trabalho, foi considerado o Problema de Roteamento de Veículos Generalizado (PRVG). Essa variação considera consumidores agrupados em *clusters*, de tal forma que cada consumidor faz parte de exatamente um *cluster*, e a demanda de cada *cluster* deve ser atendida por um único consumidor que faça parte dele. Esse problema apresenta diversas aplicações práticas, como no roteamento de embarcações em transporte marítimo, na logística da área de saúde e na coleta de lixo público urbano. Essas e outras aplicações são discutidas em [1] e [4]. A Figura 2.1 ilustra um exemplo de solução para o PRVG.

O PRVG tem recebido pouca atenção na literatura, sendo [17] o primeiro trabalho a tratar o problema, e somente em [4] foi proposto o primeiro algoritmo exato que resolve o problema. É mais comum encontrar trabalhos relacionados a uma variação bem similar, o Problema do Caixeiro Viajante Generalizado (PCVG), inicialmente estudado por [23].

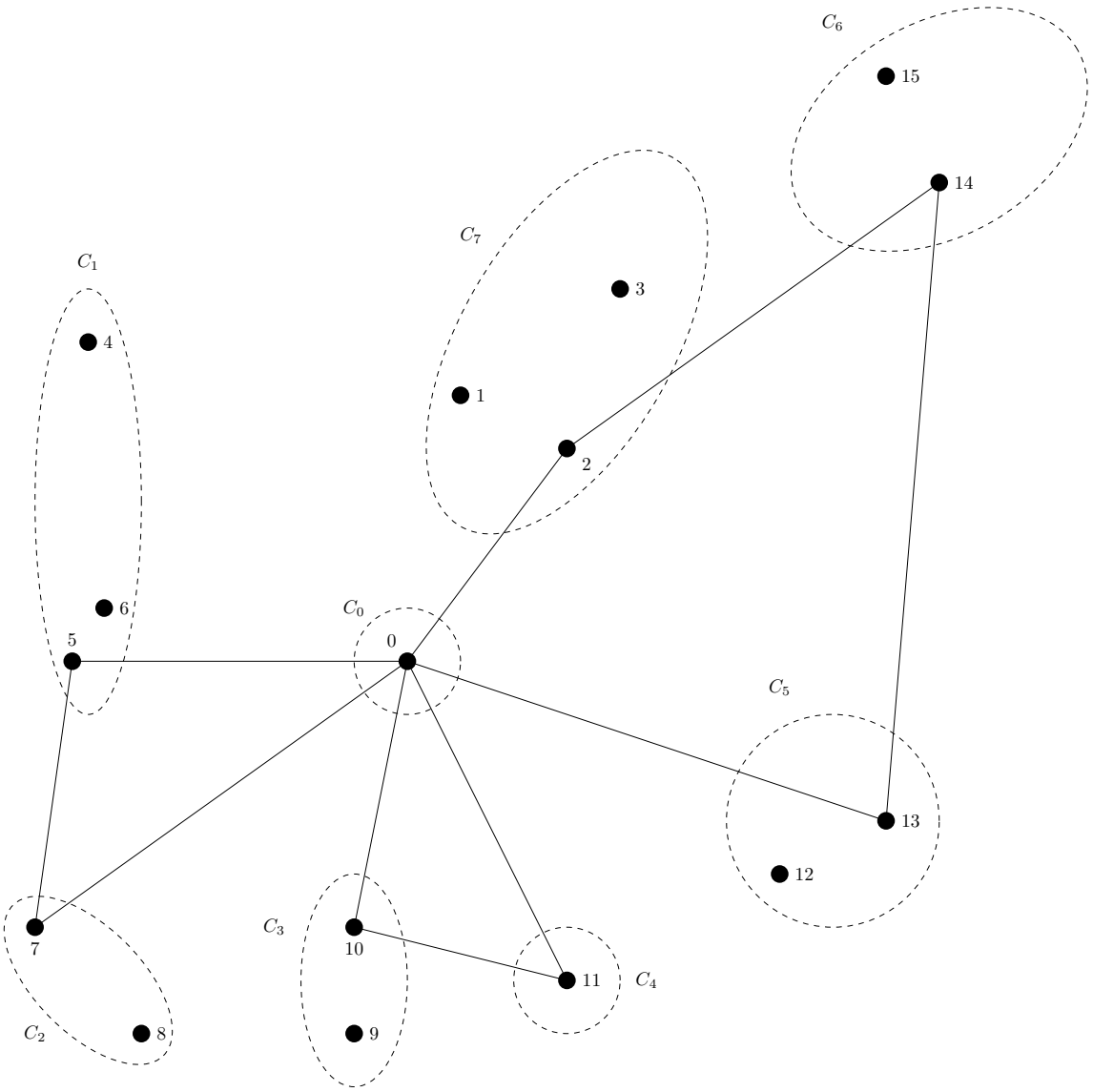


Figura 2.1: Representação visual de uma solução do PRVG. A solução é composta de sete *clusters* e três veículos. As rotas realizadas pelos veículos são: {10 e 11}, {5 e 7} e {2, 14 e 13}.

Os problemas se diferem no número de rotas utilizadas na solução: enquanto o primeiro aceita utilizar múltiplas rotas, o segundo é restrito a uma única rota. Por conta disso, o PCVG é também não capacitado, pois somente existe solução se com um único veículo é possível atender às demandas de todos os consumidores.

Recentemente, tivemos contribuições importantes para algoritmos de geração de colunas que resolvem a relaxação linear de algumas formulações para o PRV, como [35]; [2]; e [30]. No presente trabalho, propomos um algoritmo *branch-cut-and-price* que resolve o PRVG baseado em contribuições recentes da literatura. O algoritmo proposto neste trabalho não é o primeiro algoritmo *branch-cut-and-price* apresentado para o PRVG. Em [34] foi apresentado o primeiro, e único até então, algoritmo *branch-cut-and-price* que

resolve o PRVG. A principal diferença entre esse e o nosso algoritmo está no algoritmo de precificação do método de geração de colunas. Consideramos gerar rotas não elementares utilizando diferentes relaxações, enquanto [34] utilizaram somente rotas elementares no algoritmo de precificação. Os resultados experimentais, discutidos no Capítulo 5, mostram que a nossa proposta obteve bons resultados para as instâncias propostas por [4], resolvendo *duas instâncias* em aberto e superando os resultados obtidos em [4] e em [34] no tempo de solução.

Como ressaltado, tivemos motivação dentro da própria literatura combinatória para apresentar uma metodologia para PRVG, considerando tanto o pouco material publicado quanto os avanços recentes em algoritmos exatos para roteamento de veículos. No entanto, a motivação inicial partiu de um contexto dentro da literatura da robótica móvel. Nesse contexto, estudamos formas de modelar um *problema de roteamento de veículos não-holonômicos com regiões de interesse* como uma variação de um PRV e, com isso, utilizar os diversos algoritmos exatos disponíveis na literatura como uma alternativa às soluções, geralmente heurísticas, dentro da literatura de robótica móvel. Modelamos esse problema como um PRVG por meio de um processo de discretização do domínio da função que define o custo da trajetória dos veículos, discutido em mais detalhes no Capítulo 6. No entanto, essa modelagem implica utilizar uma função objetivo que consiste em minimizar o custo total de todas as rotas, porém percebemos que essa abordagem é pouco utilizada na prática dentro do contexto robótico, no qual a preferência é por uma função objetivo que minimize o custo da rota de maior custo. Apesar de isso não impedir a aplicação da nossa proposta, ela se torna menos motivadora, pois perdemos a eficiência de convergência de um método de geração de colunas que teríamos utilizando a função objetivo de um PRV clássico. Com isso, optamos por abordar como foco principal deste trabalho o desenvolvimento feito dentro do contexto do PRVG.

Este trabalho está organizado da seguinte forma: no Capítulo 3, é feita uma revisão dos algoritmos exatos utilizados para resolver PRVs na literatura nos últimos anos, além de um resumo bibliográfico de trabalhos relacionados ao PRVG. No Capítulo 4, apresentamos formulações para o PRVG e discutimos cortes válidos, algoritmos de geração de colunas e *branch-cut-and-price*. Além disso, discutimos diferentes tipos de relaxações da elementariedade da rota que é gerada pelo algoritmo de precificação. No Capítulo 5, apresentamos os experimentos computacionais realizados com os algoritmos propostos neste trabalho. Para tal, comparamos os limites inferiores obtidos pelos algoritmos da literatura com os nossos. Ademais, discutimos e mostramos os resultados do nosso algoritmo ao tentar resolver as instâncias propostas por [4].

Já no Capítulo 6, entramos em detalhes a respeito da motivação inicial deste trabalho, discutimos sobre o *problema de roteamento de veículos não-holonômicos com regiões de interesse* e suas aplicações e apresentamos um modelo para o problema baseado no PRVG através de um processo de discretização. No entanto, como explicitado anterior-

mente, a grande limitação do PRVG clássico para resolver essa aplicação está na função objetivo a ser considerada. Com isso, apenas os aspectos de modelagem são apresentados. Por fim, a conclusão e as considerações finais são apresentadas no Capítulo 7.

Capítulo 3

Revisão da Literatura

3.1 Uma breve revisão de algoritmos exatos

Muitos algoritmos exatos foram propostos para resolver o PRV e suas variantes nas últimas décadas ([41]). *Branch-and-bound* é comumente empregado para resolver diversos problemas de otimização combinatória, o PRV inclusive. Um algoritmo baseado nessa técnica utiliza uma árvore de enumeração para gerar todo o espaço de solução do problema que está sendo otimizado, tal que cada nó da árvore representa uma possível solução viável do problema. Começando pelo nó raiz, o problema pode ser dividido em duas ou mais partições da região viável de soluções para o problema. Esse procedimento é chamado de *branching* (ou ramificação). Esse passo geralmente é realizado fixando o valor de uma variável ou um conjunto de variáveis. Em cada nó é possível calcular um limite inferior (ou superior se o problema for de maximização) daquele nó ao resolver alguma relaxação do problema. Por exemplo, pode-se conseguir um limite inferior resolvendo a relaxação linear da formulação de uma Programação Inteira Mista (PIM). A relaxação linear é obtida relaxando a integralidade das variáveis inteiras. Esse limite pode ser utilizado para interromper a pesquisa em um determinado nó se não for possível encontrar mais nenhuma solução melhor do que alguma encontrada. A ideia fundamental do algoritmo *branch-and-bound* é de seguramente interromper o *branching* em um nó se o seu limite garante não existir soluções que sejam melhores que a melhor solução encontrada até o momento, ou se uma solução viável foi obtida. Esse passo é chamado de *pruning* (ou podação). Isso permite descartarmos um grande subconjunto de soluções.

A maioria dos problemas de combinatória NP-difíceis é modelada por uma formulação de uma PIM. Uma forma de melhorar os limites obtidos pela relaxação linear é por meio da adição de planos de cortes à formulação, e.g., restrições que são satisfeitas por todas as soluções inteiras viáveis, mas não são satisfeitas por um conjunto de soluções fracionárias da relaxação da formulação. Suponha que a relaxação linear seja resolvida. Logo, se a solução encontrada for não inteira e viável no problema relaxado, então existe alguma restrição que separa essa solução fracionária de uma solução inteira. Qualquer

restrição que faz essa separação é considerada uma *inequação violada*. Encontrar tais inequações é o objetivo dos algoritmos de separação. Sempre que pelo menos uma inequação violada for adicionada à formulação, haverá um *corte* da região convexa do espaço de solução do problema e, por conta disso, o problema deverá ser reotimizado. Esse processo pode ser repetido até que uma solução inteira seja encontrada ou nenhum corte do algoritmo de separação afete o espaço de solução do problema. Com a utilização de planos de cortes, o espaço de solução tende a ser menor, logo, tem-se um limite possivelmente melhor (ou mais forte) que o limite obtido pela relaxação linear sem a utilização de planos de cortes. Um algoritmo *branch-and-bound* que utiliza planos de cortes em cada nó da árvore de pesquisa é chamado de *branch-and-cut*.

Atualmente, os melhores resultados para PRVs e suas variantes têm sido utilizando o método de *geração de colunas*. Esse método é aplicado para resolver formulações lineares quando as variáveis não podem ser explicitamente enumeradas, ou quando a quantidade de variáveis é muito grande para ser enumerada na prática. O método foi introduzido por [14] e posteriormente generalizado para formulações lineares por [9]. A formulação linear utilizada no método de geração de colunas corresponde ao Problema Mestre (PM). O algoritmo começa utilizando somente um subconjunto de variáveis (ou *colunas*) para otimizar o problema — chamamos esse novo problema de Problema Mestre Restrito (PMR). Esse subconjunto de colunas deve ser o suficiente para formar uma base viável para o problema. Em seguida, o algoritmo tenta gerar novas colunas candidatas a melhorar a solução atual do PMR. Gerar tais colunas corresponde a resolver um problema auxiliar denominado de *problema de precificação*. Se pelo menos uma coluna for gerada pelo problema de precificação, então o PMR é reotimizado. Esse processo iterativo continua até que não exista coluna candidata a entrar na base da formulação PMR. Quando não houver nenhuma coluna candidata, então o valor objetivo da solução ótima do PMR é o mesmo da solução ótima do PM. Um algoritmo *branch-and-bound* que utiliza o método de geração de colunas em cada nó que for otimizado da árvore de pesquisa é chamado de *branch-and-price*.

Por fim, quando se utiliza o método de geração de colunas e planos de cortes durante um algoritmo *branch-and-bound*, temos o algoritmo *branch-cut-and-price*. Cada um dos algoritmos discutidos, em sua essência, é um algoritmo *branch-and-bound* tal que, em cada caso, aplica-se uma estratégia diferente: relaxação linear simples, planos de cortes, geração de colunas etc.

3.2 Trabalhos relacionados ao PRVG

O PRVG pode ser considerado um Problema de Localização-Roteamento (PLR), uma vez que ele representa um problema de localização (escolha do consumidor em cada *cluster*) e de roteamento (reduzir o custo da rota pelos consumidores selecionados). Para o PLR, existem diversos estudos ([24]), diferentemente do PRVG, explicitamente introduzido na literatura há pouco menos de duas décadas. A seguir, comentamos brevemente os principais trabalhos sobre o PRVG em ordem cronológica.

1. **[17]** foram os primeiros autores a estudar o PRVG, além de um caso particular do PLR. Os autores propuseram uma transformação do PRVG para o Problema de Roteamento de Arcos Capacitados (PRAC) e utilizaram um exemplo numérico para ilustrar a transformação. Apesar de nenhum algoritmo exato ter sido apresentado, esse trabalho foi muito importante para que o PRVG ganhasse um pouco de destaque na literatura.

2. **[20]** apresentaram uma formulação de tamanho polinomial para o PRVG em um trabalho não publicado. Ela é baseada em uma formulação de atribuição utilizando as conhecidas restrições de Miller-Tucker-Zemlin (MTZ) ([31]) para o Problema do Caixeiro Viajante (PCV), porém, adaptadas para o PRV ([21]). Os autores também demonstraram como é possível resolver outros problemas conhecidos a partir da formulação do PRVG, tais como o PCVG e o clássico PRV. Por fim, foi utilizado um *solver*[1] com a formulação proposta para encontrar a solução ótima da única instância apresentada por [17].

3. **[1]** discutiram diversas aplicações para o PRVG e como o problema pode ser utilizado sendo um *framework* para modelar diversos outros problemas de roteamento de veículos.

4. **[4]** propuseram o primeiro algoritmo *branch-and-cut* que resolve o PRVG. Os autores utilizaram os cortes conhecidos de capacidade para o PRV ([27]) e os cortes gerados automaticamente pelo CPLEX (e.g., cortes de Gomory, cortes disjuntivos). Além desses cortes, os autores apresentaram um novo conjunto de cortes para o PRVG denominados *same-vertex inequalities*. Esse conjunto de cortes proíbe a formação de caminhos disjuntos dentro de cada *cluster*. Essa formulação foi provada ser mais forte do que a formulação de tamanho polinomial, ou seja, melhorando os limites inferiores para o PRVG.

[1] pacote que resolve formulações lineares, geralmente utilizando o algoritmo *branch-and-bound*.

Para demonstrar a eficiência do algoritmo *branch-and-cut*, os autores propuseram o primeiro, e único até então, conjunto de instâncias para o problema. Para criá-lo, foi utilizado um método similar ao realizado por [13] ao propor instâncias para o PCVG a partir de instâncias do PCV. Dessa vez, foram utilizadas como base as instâncias do PRV da CVRP-library, disponíveis em http://vrp.atd-lab.inf.puc-rio.br/ [2]. Dois subconjuntos de instâncias foram criados. O primeiro (subconjunto médio) representa o subconjunto cujas instâncias têm número de vértices variando entre 16 e 101, e o segundo (subconjunto grande) com quantidade de vértices variando entre 101 e 262. Utilizando o algoritmo *branch-and-cut*, foi possível resolver 139 das 148 instâncias do subconjunto médio, e 3 das 10 instâncias do subconjunto grande.

Por fim, também foi proposta uma heurística baseada em *Large Neighborhood Search* (LNS) para calcular um limite superior inicial do algoritmo *branch-and-cut*. Essa heurística é uma adaptação de outra, proposta por [40], e similar à utilizada em [37]. Ela consiste em modificar uma solução inicial ao, repetidamente, remover um conjunto de vértices (destruir uma solução) e reinserir o mesmo conjunto, ou um equivalente, formando uma nova solução (reconstruir a solução). A heurística demonstrou ser eficiente e de qualidade, e os autores ressaltaram que o limite superior encontrado pela heurística foi a solução ótima de 136 instâncias.

5. **[34]** apresentaram o primeiro algoritmo *branch-cut-and-price* que resolve o PRVG. Os autores aplicaram as metodologias propostas por [35] e [2] para alcançar a elementariedade da rota no problema de precificação. Além disso, eles propuseram um algoritmo dinâmico para realizar a expansão de *labels* por *cluster*, em vez de vértices, no problema de precificação.

   Para realizar os experimentos, foram utilizados os mesmos conjuntos de instâncias propostos por [4]. Com o *branch-cut-and-price*, foi possível melhorar significativamente os limites inferiores quando comparado ao *branch-and-cut* proposto por [4]. No entanto, devido ao alto custo computacional do algoritmo de precificação, o tempo total para resolver as instâncias não foi necessariamente melhor.

   Os autores compararam o algoritmo proposto com o *branch-and-cut* implementado em [4] utilizando somente o subconjunto médio de instâncias e concluíram que não houve dominância clara entre os dois algoritmos. No entanto, com o *branch-cut-and-price* foi possível resolver algumas instâncias que não foram resolvidas pelo algoritmo *branch-and-cut*. Os resultados demonstraram que, com o algoritmo *branch-cut-and-price*, foi possível resolver 141 das 148 instâncias do subconjunto médio de instâncias. Os autores não apresentaram resultados da metodologia utilizando o conjunto grande de instâncias propostas em [4].

[2] último acesso em 2017.

Com isso, do conjunto de instâncias propostas por [4], têm-se oito instâncias (uma instância do subconjunto de tamanho médio e sete instâncias do subconjunto de tamanho grande) que não foram resolvidas pelo algoritmo *branch-and-cut* proposto por [4] nem pelo algoritmo *branch-cut-and-price* proposto por [34]. No trabalho proposto neste texto, tem-se como proposta apresentar uma metodologia que, utilizando o mesmo conjunto de instâncias proposto por [4], consiga resolver todas as instâncias resolvidas em [4] e [34], além de também resolver algumas instâncias que não tiveram solução ótima apresentada em nenhum dos trabalhos.

Capítulo 4

Metodologia

Neste Capítulo, é discutida a metodologia desenvolvida durante o trabalho para resolver o PRVG. Na Seção 4.2, discutimos duas formulações inteiras e lineares para o PRVG. A primeira, introduzida por [4], é baseada em fluxo em arcos e tem tamanho polinomial de restrições e variáveis. A segunda é uma variação da formulação proposta por [3], baseada em particionamento de conjuntos, com tamanho polinomial de restrições e tamanho exponencial de variáveis. Na Seção 4.3, discutimos como inequações válidas para o PRV podem ser aplicadas para o PRVG em ambas as formulações. Na Seção 4.4, explicamos como geração de colunas é aplicada à formulação de particionamento de conjuntos para resolver a sua relaxação linear sem explicitamente enumerar todas as variáveis. Na Seção 4.5, abordamos a estratégia de solução do algoritmo de precificação do método de geração de colunas que explora a relaxação da elementariedade das rotas geradas. Por fim, o algoritmo *branch-cut-and-price* é discutido na Seção 4.6, na qual é detalhado como cada nó da árvore de pesquisa é resolvido, além das decisões relacionadas ao *branch-and-bound*, tais como inicialização, ramificação e escolha do nó.

4.1 Definição

A definição formal do PRVG é dada da seguinte forma. Seja $G = (V, E)$ um grafo não direcionado, tal que $V = \{0, \ldots, n\}$ é o conjunto de vértices e $E = \{\{i, j\} \mid i, j \in V\}$ é o conjunto de arestas. Então, os vértices do grafo representam os consumidores, enquanto as arestas correspondem a um caminho bidirecional entre eles. Entre os vértices, um deles é utilizado para representar o depósito. Em algumas formulações, o caminho é unidirecional entre os consumidores, nesse caso, em vez do conjunto E de arestas, é utilizado um conjunto A de arcos. Neste trabalho, o caminho entre os consumidores é bidirecional e representado por uma aresta $e = \{i, j\}$ ou por dois arcos (i, j) e (j, i). Toda vez que um veículo percorre uma aresta (ou um arco), é calculado um custo de locomoção w_e (ou w_{ij} para arcos). O conjunto V de vértices é particionado em subconjuntos (denominados de

clusters) não vazios e disjuntos $\{C_0, C_1, \ldots, C_m\}$, em que $C_0 = \{0\}$ é um *cluster* com um único vértice que representa o depósito. Cada *cluster* C_u tem uma demanda associada d_u, tal que $d_u > 0$ para $u \in \{1, \ldots, m\}$ e $d_0 = 0$. Existem K veículos idênticos com capacidade Q. Toda vez que um veículo visita um vértice $i \in V$, é consumido $d_{\alpha(i)}$ da sua capacidade, em que $\alpha(i) \in M = \{0, \ldots, m\}$ corresponde ao índice do *cluster* do qual o vértice i pertence, portanto, $i \in C_{\alpha(i)}$ para qualquer $i \in V$.

O PRVG consiste em encontrar um conjunto de rotas, uma para cada veículo, tal que cada rota atenda às seguintes restrições: toda rota deve começar e terminar no vértice depósito, a demanda total de cada rota não deve exceder a capacidade Q de cada veículo, e cada *cluster* deve ser visitado exatamente uma vez em somente um vértice do *cluster*. O objetivo é encontrar um conjunto de rotas que atenda às restrições citadas e que minimize a soma dos custos de todas as rotas. O custo de cada rota é definido pela soma dos custos das arestas que compõem aquela rota.

4.2 Formulações

Nesta Seção, apresentamos duas formulações para o PRVG: uma baseada em fluxo em arcos, e outra baseada em particionamento de conjuntos. A segunda é utilizada no método de geração de colunas para que possamos resolver a sua relaxação linear sem explicitamente enumerar todas as suas variáveis. Como notação adicional para as formulações, tem-se: para qualquer subconjunto $S \subset V$, $\delta(S) = \{(i,j) \in A \mid i \in S, j \notin S \text{ ou } i \notin S, j \in S\}$, $\delta^+(S) = \{(i,j) \in A \mid i \in S, j \notin S\}$, e $\delta^-(S) = \{(i,j) \in A \mid i \notin S, j \in S\}$. Por questões de simplicidade, quando $S = \{i\}$, escrevemos $\delta(i)$ em vez de $\delta(\{i\})$. Para o caso de grafos não direcionados, a notação se mantém, exceto que os pares de arcos (i,j) e (j,i) são substituídos pela aresta $\{i,j\}$.

4.2.1 Formulação baseada em fluxo em arcos

A seguinte formulação foi apresentada por [4] e corresponde a uma derivação da formulação de *single-commodity*[1] para o PRV proposta por [16]. A formulação é baseada em um grafo direcionado e utiliza uma variável binária x_{ij} para cada arco $(i,j) \in A$, tal que se o arco (i,j) é atravessado por algum veículo, então $x_{ij} = 1$ (ou 0, caso contrário).

[1] do inglês, mercadoria única

Uma variável contínua $f_{ij} \geq 0$ para todo arco $(i,j) \in A$ indica o fluxo de i para j, tal que o fluxo corresponde à quantidade de mercadoria carregada por um veículo. Como notação adicional, tem-se: $x(A') = \sum_{(i,j)\in A'} x_{ij}$ e $f(A') = \sum_{(i,j)\in A'} f_{ij}$, tal que $A' \subseteq A$. A formulação $(\mathcal{F}_1)$ é definida da seguinte forma:

$$(\mathcal{F}_1) \quad \text{minimize} \quad \sum_{(i,j)\in A} w_{ij}x_{ij} \tag{4.1}$$

$$\text{sujeito a} \quad x(\delta^+(C_u)) = 1 \quad \forall u \in M \setminus \{0\}, \tag{4.2}$$

$$x(\delta^-(C_u)) = 1 \quad \forall u \in M \setminus \{0\}, \tag{4.3}$$

$$x(\delta^+(C_0)) = K, \tag{4.4}$$

$$x(\delta^+(i)) = x(\delta^-(i)) \quad \forall i \in V, \tag{4.5}$$

$$f(\delta^+(i)) - f(\delta^-(i)) = \frac{1}{2}d_{\alpha(i)}(x(\delta^+(i)) + x(\delta^-(i))) \quad \forall i \in V \setminus C_0, \tag{4.6}$$

$$d_{\alpha(i)} \leq f_{ij} \leq (Q - d_{\alpha(j)})x_{ij} \quad \forall (i,j) \in A. \tag{4.7}$$

Na formulação acima, (4.2) e (4.3) correspondem às restrições de associação para que cada *cluster* seja visitado uma única vez. A restrição (4.4) garante que exatamente K veículos saiam do *cluster* depósito. As restrições (4.5) garantem a continuidade de cada rota. Por sua vez, as restrições (4.6) impõem que o fluxo da mercadoria em cada vértice seja o necessário para atender à demanda de cada consumidor, e (4.7) impõem limites para a quantidade de mercadoria transportada. Por fim, a função objetivo (4.1) minimiza o custo total dos arcos atravessados pelos veículos.

Em [4], essa formulação foi utilizada para resolver o PRVG por meio de um algoritmo *branch-and-bound*, e também por um algoritmo *branch-and-cut* ao adicionar planos de cortes. A metodologia utilizada e os resultados obtidos podem ser consultados no próprio artigo dos autores; não estendemos a discussão da metodologia neste trabalho, pois utilizamos a formulação proposta pelos autores somente para um referencial teórico.

4.2.2 Formulação baseada em particionamento de conjuntos

Na formulação baseada em particionamento de conjuntos, cada variável de decisão representa uma rota viável para o PRVG, e o conjunto de todas essas variáveis representa

o conjunto de todas as rotas viáveis. Dessa forma, tem-se um número exponencial de variáveis. O fato de o modelo ter somente variáveis que representam rotas viáveis faz com que essa formulação detenha, normalmente, limites de relaxação linear melhores que formulações cuja rota é determinada por variáveis de decisão em arcos (ou arestas).

Seja $\mathcal{R}$ o conjunto de rotas que começam no vértice depósito, visitam uma sequência de vértices com uma demanda total menor ou igual a Q e, por fim, retornam ao vértice depósito. Uma variável binária λ_r indica se a rota $r \in \mathcal{R}$ pertence ou não à solução. Uma formulação ($\mathcal{F}_2$) para o PRVG baseada em particionamento de conjuntos pode ser definida da seguinte forma:

$$(\mathcal{F}_2) \quad \text{minimize} \quad \sum_{r \in \mathcal{R}} \lambda_r c_r \tag{4.8}$$

$$\text{sujeito a} \quad \sum_{r \in \mathcal{R}} \lambda_r = K, \tag{4.9}$$

$$\sum_{r \in \mathcal{R}} a_u^r \lambda_r = 1 \qquad \forall u \in M \setminus \{0\}, \tag{4.10}$$

$$\lambda_r \in \{0, 1\} \qquad \forall r \in \mathcal{R}. \tag{4.11}$$

Nessa formulação, a função objetivo (4.8) minimiza o custo total das rotas selecionadas na solução. A restrição (4.9) impõe uma solução com K rotas, uma para cada veículo. As restrições (4.10) garantem que haja somente uma única visita em cada *cluster* e que seja feita por algum veículo. Por fim, as restrições (4.11) forçam valores binários para as variáveis que definem se uma rota pertence à solução.

Neste trabalho, relaxamos o conjunto de rotas viáveis $\mathcal{R}$ para que também inclua rotas que violem a condição de que somente um consumidor dentro de cada *cluster* possa ser visitado. As rotas que violam essa condição são chamadas de *rotas não elementares*, e isso permite a formação de ciclos dentro da rota. Essa mudança é válida para o problema, pois as restrições da formulação $\mathcal{F}_2$ não permitem que essas rotas adicionais façam parte de uma solução viável. No entanto, elas podem fazer parte da solução viável da relaxação linear da formulação. Isso implica um enfraquecimento do limite inferior da relaxação linear da formulação. Em contrapartida, essa relaxação permite reduzir significativamente o tempo computacional do algoritmo de precificação do método de geração de colunas. Essa relaxação foi fundamental para que pudéssemos resolver instâncias abertas da literatura, tal como é discutido no Capítulo 5.

O custo c_r associado à rota r é dado pela soma dos pesos de todas as arestas atravessadas ao longo da rota. Para cada rota $r \in \mathcal{R}$ e um *cluster* $u \in M \setminus \{0\}$, o coeficiente a_u^r indica quantas vezes o *cluster* foi visitado durante a rota. Para rotas não elementares (uma rota que visita mais de uma vez um mesmo *cluster*), os coeficientes a_u^r podem assumir valores inteiros não negativos; caso contrário, para rotas elementares, os coeficientes

assumem valores binários. É importante ressaltar que a formulação naturalmente descarta qualquer solução com rotas não elementares, uma vez que a restrição (4.10) proíbe $\lambda_r = 1$ para qualquer rota r com algum coeficiente $a_u^r > 1$. Portanto, as restrições da definição do PRVG são atendidas mesmo incluindo rotas não elementares no conjunto $\mathcal{R}$.

4.3 Inequações Válidas

As formulações apresentadas ($\mathcal{F}_1$ e $\mathcal{F}_2$) podem ser reforçadas por meio da adição de inequações válidas (ou cortes). Essas inequações removem um conjunto de soluções fracionárias e mantêm as soluções inteiras do problema. Como demonstrado por [4], qualquer inequação baseada em arcos para o PRV pode ser transformada para o PRVG, fazendo com que a variável que representa o valor fracionário de cada arco no PRV seja um somatório das variáveis que representam o valor fracionário dos arcos entre um par de *clusters* no PRVG.

Dessa forma, é possível adicionar diversos cortes do PRV já existentes na literatura ao PRVG. O pacote CVRPSEP ([26]) contém separações heurísticas eficientes para as seguintes famílias de cortes: *rounded capacities*, *framed capacities*, *strengthened combs*, *multistars* e *extended hypertours*. Enquanto todas as famílias de cortes apresentam um papel importante em algoritmos *branch-and-cut* ([27]), somente os cortes de *rounded capacity* e, por um fator significativamente menor, de *strengthened combs* podem fortalecer de forma significativa a formulação $\mathcal{F}_2$, como observado por [15]. Como ressaltado por [33], isso acontece devido ao fato de a maioria dos cortes das outras famílias já estar definidos de forma implícita nas restrições (4.9) e (4.10).

Portanto, neste trabalho, foram utilizados somente os cortes de *rounded capacity*. Dessa forma, também é possível comparar os resultados obtidos neste trabalho com os resultados de [4] e [34], nos quais, de mesmo modo, somente os cortes de *rounded capacity* foram adicionados. Em [4], igualmente foram adicionados os cortes de *same-vertex inequalities*, mas estes são utilizados somente para garantir que só um vértice seja visitado em cada *cluster*. A formulação $\mathcal{F}_2$ já garante, por definição, que toda rota $r \in \mathcal{R}$ visite um único vértice em cada *cluster*, do contrário, a rota não seria válida.

Os cortes de *rounded capacity* para o PRVG impõem um número mínimo de veículos que devem visitar um subconjunto de *clusters*. Um limite inferior para o número de veículos que devem visitar um subconjunto de *clusters* $M' \subseteq M \setminus \{0\}$ é dado por $k(M') = \lceil q(M')/Q \rceil$, em que $q(M') = \sum_{u \in M'} d_u$ é a demanda total do subconjunto de *clusters*. Dessa forma, as seguintes inequações são válidas para a formulação $\mathcal{F}_1$:

$$x(\delta^+(S)) + x(\delta^-(S)) \geq 2k(M'), \quad S = \bigcup_{u \in M'} C_u, M' \subseteq M \setminus \{0\}. \tag{4.12}$$

De forma semelhante, a mesma inequação (4.12) pode ser adaptada para a formulação $\mathcal{F}_2$ da seguinte maneira:

$$\sum_{e \in \delta(S)} \sum_{r \in \mathcal{R}} b_e^r \lambda_r \geq 2k(M'), \quad S = \bigcup_{u \in M'} C_u, M' \subseteq M \setminus \{0\} \tag{4.13}$$

cujo coeficiente b_e^r indica se a aresta e faz parte da formação da rota r.

Uma vantagem durante o método de geração de colunas, discutido na Seção 4.4, é de que as inequações (4.13) são consideradas robustas, ou seja, o efeito das variáveis duais dessas inequações na formulação $\mathcal{F}_2$ pode ser capturado no grafo de custos reduzidos que é construído durante o problema de precificação.

4.4 Geração de Colunas

O conjunto de variáveis (ou colunas) da formulação $\mathcal{F}_2$ apresenta tamanho exponencial em relação à quantidade de vértices ou de *clusters*. Portanto, até mesmo para resolver a relaxação linear da formulação, não é prático gerar todo o conjunto $\mathcal{R}$. Com isso, faz-se necessário resolver a relaxação linear de $\mathcal{F}_2$ por um outro meio.

Utilizamos o método de *geração de colunas*, que consiste em resolver um problema linear contendo somente um subconjunto de variáveis da formulação e, de forma iterativa, adiciona a esse subconjunto variáveis candidatas a melhorar a solução. As variáveis candidatas são quaisquer que tenham custo reduzido negativo. Nesse método, o problema linear é o PM, e o problema linear restrito é o PMR. Além disso, o problema responsável por adicionar novas colunas ao PMR é denominado de *problema de precificação*. Em geral, esse método possibilita encontrar a solução ótima de uma formulação linear enumerando explicitamente uma quantidade bem menor de variáveis se comparado ao conjunto original de variáveis da formulação.

No contexto do PRVG, o PM do método de geração de colunas corresponde à relaxação linear da formulação $\mathcal{F}_2$, que se faz ao relaxar a restrição (4.11) para $\lambda_r \geq 0$. Note que, apesar de não definirmos de forma explícita um limite superior para as variáveis na relaxação linear, em qualquer solução viável nenhuma variável pode assumir valor maior que 1 devido à restrição (4.10). Portanto, as variáveis que definem as rotas na relaxação linear têm limite superior implícito menor ou igual a 1. O PMR é definido da mesma forma que o PM, no entanto, utilizando um subconjunto reduzido de rotas $\mathcal{R}' \subset \mathcal{R}$. Dada

uma solução do PMR, o conjunto de rotas candidatas a serem adicionadas à formulação do PMR é definido como:

$$\mathcal{R}'' = \{r \mid r \in \mathcal{R} \setminus \mathcal{R}', \bar{c}_r < 0\}, \tag{4.14}$$

tal que $\bar{c}_r$ representa o valor do custo reduzido da rota c_r. Esse custo reduzido é calculado utilizando os valores das variáveis duais da formulação $\mathcal{F}_2$ e dos cortes adicionados, da seguinte forma:

$$\bar{c}_r = c_r - \pi_0 - \sum_{u \in M \setminus \{0\}} a_u^r \pi_u - \sum_{S \subseteq V} \sum_{e \in \delta(S)} b_e^r \mu_S, \tag{4.15}$$

em que π_0 e π_u são as variáveis duais das restrições (4.9) e (4.10), respectivamente. E, ainda, μ_S corresponde às variáveis duais dos cortes de *rounded capacity* (4.13) que foram adicionados à formulação. Portanto, qualquer subconjunto não vazio de $\mathcal{R}''$ é suficiente para que o PMR seja reotimizado e haja mais uma iteração do método de geração de colunas. Caso o conjunto $\mathcal{R}''$ seja vazio, então a solução ótima do PMR é equivalente à solução ótima do PM.

Como alternativa, é possível construir um *grafo de custos reduzidos* e, então, resolver um problema de otimização sobre esse grafo para encontrar rotas de custo reduzido negativo. Dessa forma, não é necessário calcular explicitamente o custo reduzido de cada rota (4.15). O custo reduzido $\bar{c}_r$ pode ser obtido simplesmente somando os custos reduzidos de cada arco atravessado para formar a rota r. O custo reduzido $\bar{w}_{ij}$ de cada arco do grafo de custos reduzidos pode ser definido da seguinte forma:

$$\bar{w}_{ij} = w_{\{i,j\}} - \left(\frac{\pi_{\alpha(i)} + \pi_{\alpha(j)}}{2}\right) - \sum_{S \subseteq V \mid \{i,j\} \in \delta(S)} \mu_S, \quad \forall (i,j) \mid \{i,j\} \in E. \tag{4.16}$$

O problema de otimização utilizado para encontrar colunas candidatas para serem incluídas no PMR é chamado de *problema de precificação*. No caso do PRVG, esse problema pode ser definido como: *dado um grafo cujos arcos têm pesos (positivos ou negativos), encontre uma ou mais rotas de custo negativo de tal forma que a demanda total de cada rota não exceda a capacidade do veículo*. Ainda é possível deixar o problema mais restrito, impondo a elementariedade da rota. De qualquer modo, o problema continua fazendo parte da classe de problemas NP-Difícil.

4.5 Algoritmo de Precificação

O problema de precificação em questão é conhecido como *Elementary Shortest Path Problem With Resource Constraint* (ESPPRC)[2]. Esse é um problema clássico da literatura e aparece como problema de precificação em diversos algoritmos de geração de colunas para variações do PRV. O ESPPRC é um problema de caminho mínimo em um grafo no qual os consumidores possuem uma quantidade de recursos que são consumidos durante uma visita. Uma restrição garante que o total de recursos consumidos não ultrapasse determinada quantidade. Para o caso em que o caminho encontrado não precisa ser elementar, em que um nó intermediário pode ser visitado mais de uma vez, o problema é conhecido como *Shortest Path Problem With Resource Constraint* (SPPRC)[3].

Alguns trabalhos exploram a relação entre a elementariedade das rotas e a eficiência do algoritmo de precificação ([11, 35]). Um subproblema de precificação que encontra rotas elementares provê limites inferiores mais fortes, no entanto, o custo computacional necessário para encontrar essas rotas é maior. No caso de rotas não elementares, ainda é possível obter limites inferiores melhores utilizando uma relaxação que proíbe a formação de determinados ciclos. Por exemplo, o algoritmo dinâmico desenvolvido por [6] impede a formação de 2-ciclos (ciclos de comprimento menor ou igual a 2). Esse algoritmo tem complexidade $\mathcal{O}(n^2Q)$ e as rotas geradas por esse algoritmo são denominadas **q**-*routes*. [18] generalizaram a ideia da eliminação de 2-ciclos para uma eliminação de k-ciclos, no entanto, apesar de o algoritmo ter uma contribuição teórica interessante, ele não é muito prático, pois a complexidade cresce fatorialmente à medida que se deseja eliminar ciclos maiores. Recentemente, [2] propuseram uma nova relaxação utilizando **ng**-*routes* para alcançar elementariedade parcial. Com essa nova relaxação, foi possível melhorar diversos resultados do estado da arte de PRVs ([36]).

Para o PRVG, exploramos a utilização de **q**-*routes* e **ng**-*routes* no algoritmo de precificação para encontrar rotas não necessariamente elementares. Os trabalhos que introduziram o conceito dessas rotas utilizaram o PRV como uma aplicação. Neste trabalho, implementamos a variação dessas rotas para considerar o agrupamento de consumidores em *clusters*.

[2] do inglês, problema do caminho mínimo elementar com restrição de recursos.
[3] do inglês, problema do caminho mínimo com restrição de recursos.

4.5.1 A relaxação q-*route* aplicada ao PRVG

O algoritmo proposto por [6] para calcular caminhos de custo mínimo sem a formação de 2-ciclos consiste em utilizar duas matrizes bidimensionais de programação dinâmica. As duas matrizes têm as mesmas dimensões, e cada entrada (j, p) das matrizes corresponde a um caminho sem a formação de 2-ciclos de menor custo possível entre o vértice depósito e um vértice $j \in V \setminus C_0$, tal que o consumo total de recursos ao longo do caminho seja $p \in \{0, \ldots, Q\}$. Além disso, uma restrição é imposta às duas matrizes: dada uma mesma entrada (j, p) em ambas as matrizes, os caminhos correspondentes em ambas as matrizes devem ter os penúltimos vértices visitados distintos. Para simplificar, considera-se uma matriz sendo a *principal*, e a outra como a *auxiliar*. Além disso, para uma mesma entrada (j, p), a matriz principal sempre terá um caminho de custo menor ou igual se comparada com a matriz auxiliar. Com isso, sempre é possível evitar a formação de 2-ciclos, pois pelo menos uma das matrizes permite uma expansão para um vértice de tal forma que não ocorrerá uma formação de 2-ciclos. Essa ideia é simples e efetiva, porque como o custo computacional adicional é o de computar uma segunda matriz, então a complexidade assintótica do algoritmo é a mesma com ou sem a proibição de 2-ciclos.

A aplicação do algoritmo ao PRVG é bem semelhante à original. A única diferença está relacionada na detecção de 2-ciclos que, no caso do PRVG, é feita verificando se a formação do ciclo ocorre no *cluster*, e não nos vértices. A seguir, é apresentado o algoritmo de programação dinâmica para calcular a menor rota sem a formação de 2-ciclos para o PRVG.

Sejam H e H' duas matrizes de programação dinâmica de dimensões $n \times (Q + 1)$, em que cada entrada $H(j, p)$ e $H'(j, p)$ corresponde ao caminho de menor custo que começa no vértice depósito e termina no vértice j sem a formação de 2-ciclos e com uma demanda total p consumida. Para cada entrada (j, p) das matrizes, os *clusters* dos vértices predecessores imediatos ao vértice j nos caminhos formados por $H(j, p)$ e $H'(j, p)$ devem ser distintos. Com isso, podemos convencionar $H(j, p) \leq H'(j, p)$ para todo $j \in V$ e $p \in \{0, \ldots, Q\}$.

As matrizes H e H' são atualizadas utilizando as seguintes relações de recorrência:

$$H(j,p) = \begin{cases} \infty, & \text{se } p < d_{\alpha(j)} \\ \bar{w}_{0j}, & \text{se } p = d_{\alpha(j)} \\ \min\limits_{i \in V \setminus C_{\alpha(j)}} \{h(i, j, p - d_{\alpha(j)}) + \bar{w}_{ij}\}, & \text{se } p > d_{\alpha(j)} \end{cases} \tag{4.17}$$

$$H'(j,p) = \begin{cases} \infty, & \text{se } p \leq d_{\alpha(j)} \\ \min\limits_{i \in V \setminus C_{\alpha(j)} | \alpha(i) \neq \alpha(\tau(j,p))} \{h(i,j,p-d_{\alpha(j)}) + \bar{w}_{ij}\}, & \text{se } p > d_{\alpha(j)} \end{cases} \tag{4.18}$$

em que $\tau(j,p)$ corresponde ao vértice predecessor imediato ao vértice j no caminho calculado em $H(j,p)$. Além disso, $h(i,j,p)$ indica o valor mínimo para a expansão dos caminhos $H(j,p)$ ou $H'(j,p)$ a partir do vértice i sem permitir a formação de 2-ciclos, sendo calculado da seguinte forma:

$$h(i,j,p) = \begin{cases} H(i,p), & \text{se } \alpha(\tau(i,p)) \neq \alpha(j) \\ H'(i,p), & \text{caso contrário .} \end{cases} \tag{4.19}$$

Com os valores da matriz H, é possível calcular o menor custo reduzido de uma rota sem a formação de 2-ciclos da seguinte forma:

$$\min_{i \in V \setminus C_0, 0 \leq p \leq Q} \{H(i,p) + \bar{w}_{i0}\}. \tag{4.20}$$

Portanto, a complexidade assintótica para calcular a rota de menor custo sem a formação de 2-ciclos é a mesma de computar todos os valores das matrizes H e H', que é $\mathcal{O}(n^2Q)$.

É importante ressaltar que os vértices visitados ao longo do caminho podem ser recuperados se utilizarmos uma matriz adicional que armazena o vértice predecessor e que é atualizada toda vez que for calculada alguma entrada na matriz H. Dessa forma, é possível retornar à rota de menor custo reduzido.

Se o valor do custo reduzido da rota calculada em (4.20) for não negativo, então não existe nenhuma rota candidata a ser adicionada ao PMR. Caso contrário, existe pelo menos uma rota candidata. Mais rotas também podem ser retornadas calculando diferentes entradas da matriz H caso elas também tenham custo reduzido negativo.

Esse algoritmo tem a vantagem de apresentar uma complexidade pseudopolinomial no pior caso, no entanto, garantir a eliminação somente de ciclos de tamanho dois não representa um ganho significativo no limite inferior da relaxação linear do problema, pois ciclos maiores ainda são frequentes.

4.5.2 A relaxação ng-*route* aplicada ao PRVG

Recentemente, [2] apresentaram uma ideia simples, porém efetiva, para alcançar elementariedade parcial da rota. Basicamente, em vez de "memorizar" todos os vértices

já visitados para evitar a formação de ciclos (como acontece para rotas elementares), somente um subconjunto limitado de vértices é "memorizado". Esse subconjunto é utilizado para evitar a formação de ciclos, no entanto, sem garantir que nenhum ciclo seja formado. A vantagem desse algoritmo é o fato de que ciclos que são mais comuns durante a precificação de rotas não elementares contêm vértices definidos em uma vizinhança próxima, portanto, utilizando uma "memorização" baseada em proximidade, é possível evitar prováveis formações de ciclos. Rotas formadas por meio dessa definição são chamadas de **ng**-*routes*.

Para cada vértice i é definido um conjunto $NG(i)$ de vértices, tal que os vértices desse conjunto estão relacionados de alguma forma ao vértice i (incluindo o próprio vértice i). Uma possível relação pode ser a de vizinhança, e.g., $NG(i)$ contém os vértices mais próximos de i, incluindo i. Esses conjuntos são denominados **ng**-*set* e eles contêm os vértices que i consegue "memorizar". Com isso, durante a construção de um caminho P, ao tempo em que o caminho alcança o vértice i, tem-se um conjunto $\Pi(P)$ de vértices "memorizados", de tal forma que só será possível expandir o caminho até i se $i \notin \Pi(P)$. E, ao expandir o caminho para i, o conjunto de vértices "lembrados" pelo caminho P é atualizado para "esquecer" os vértices que não podem ser "lembrados" por i. Obviamente, todo vértice que estiver no conjunto $\Pi(P)$ é um candidato para formação de ciclos. Logo, o tamanho do conjunto $NG(i)$ é um fator importante para determinar o quão próximo uma **ng**-*route* está de alcançar a elementariedade, pois quanto maior o conjunto, menor será a probabilidade de formação de ciclos.

Ao aplicar a relaxação **ng**-*route* ao PRVG, não é suficiente verificar a formação de ciclos somente de vértices, pois uma rota que visita somente vértices distintos pode visitar um mesmo *cluster* mais de uma vez. Por conta disso, os **ng**-*sets* são construídos utilizando a mesma ideia proposta por [2], porém considerando *clusters* como formação de ciclos. Nesse caso, temos para cada *cluster* $u \in M \setminus \{0\}$ um conjunto $NG(u)$ que contém os *clusters* "memorizáveis" por u. Além disso, todo **ng**-*set* possui uma cardinalidade limitada $|NG(u)| \leq \Delta$, tal que Δ é um parâmetro definido *a priori*. Seja $P = (\iota_0, \iota_1, \ldots, \iota_s)$ um caminho que começa no vértice depósito $\iota_0 = 0$, e qualquer vértice ι_y tal que $y \in \{1, \ldots, s\}$ é visitado logo após o vértice ι_{y-1}. Por fim, o conjunto $\Pi(P) \subseteq M \setminus \{0\}$ de *clusters* "memorizáveis" para uma expansão imediata do caminho P é definido da seguinte forma:

$$\Pi(P) = \left\{ \alpha(\iota_y) : \alpha(\iota_y) \in \bigcap_{x=y+1}^{s} NG(\alpha(\iota_x)), y = 1, \ldots, s-1 \right\} \cup \{\alpha(\iota_s)\}. \tag{4.21}$$

Assumindo que $P_{(y)}$ é um caminho parcial de P, tal que $P_{(y)}$ contém os primeiros $y+1$ vértices de P. Então, é dito que o caminho P é **ng**-*feasible* se, e somente se, para qualquer caminho parcial $P_{(y)}$ em que $y \in \{1, \ldots, s\}$ temos $\alpha(\iota_y) \notin \Pi(P_{(y-1)})$. Dessa forma, dado um caminho $P_{(y)}$ que é **ng**-*feasible*, então é possível calcular o conjunto $\Pi(P_{(y+1)})$ da expansão imediata do caminho $P_{(y)}$ para o vértice ι_{y+1} como:

$$\Pi(P_{(y+1)}) = (\Pi(P_{(y)}) \cap NG(\alpha(\iota_{y+1}))) \cup \{\alpha(\iota_{y+1})\}. \tag{4.22}$$

Os **ng**-*sets* são construídos utilizando a proximidade entre os *clusters*. Ou seja, o **ng**-*set* de um *cluster* específico é determinado pelos *clusters* mais próximos a ele. Nesse caso, podemos definir a proximidade entre dois *clusters* de algumas formas diferentes, e.g., podemos considerar a distância entre *clusters* como a menor distância entre dois vértices ou como a média das distâncias entre cada par de vértices entre os *clusters*. Neste trabalho, foi considerada a média das distâncias entre os pares de vértices, pois assim levamos em consideração a distância relativa entre os vértices dentro de um mesmo *cluster*. A proximidade entre os *clusters* u e v é definida da seguinte forma:

$$prox(u, v) = \sum_{e \in \delta(C_u) \cap \delta(C_v)} \frac{w_e}{|\delta(C_u) \cap \delta(C_v)|}. \tag{4.23}$$

Como observado, a qualidade da solução está diretamente relacionada com a definição do conjunto $NG(u)$. Neste trabalho, como o tamanho de cada conjunto é limitado por um parâmetro único Δ, quanto maior definirmos o valor de Δ menor será a probabilidade de formar ciclos. Com isso, fazendo $\Delta = m$, garantimos que a rota seja elementar.

Ao utilizar **ng**-*routes* no algoritmo de precificação da geração de colunas para o PRVG, o objetivo é encontrar as rotas **ng**-*feasible* de menor custo sem que seja necessário enumerar todas as possíveis rotas **ng**-*feasible*. Para isso, utilizamos um algoritmo de enumeração de *labels* que consiste em encontrar um caminho de menor custo a partir da enumeração de soluções utilizando duas operações: expansão e dominância.

4.5.3 Algoritmo de enumeração de *labels*

O algoritmo de enumeração de *labels* é um algoritmo de programação dinâmica cuja ideia principal é enumerar todas as soluções, no entanto, evitando enumerar um subconjunto de soluções que podem ser garantidamente descartadas como candidatas à solução ótima.

Cada *label* corresponde a uma solução viável, e duas principais operações podem ser realizadas na *label*: **expansão** e **dominância**. A expansão é utilizada para enumerar outras soluções a partir de soluções já existentes, e a dominância é utilizada para reduzir o espaço de soluções enumeradas ao remover *labels* que são dominadas por outras *labels*.

No contexto de **ng**-*routes*, cada *label* corresponde a um caminho **ng**-*feasible* do vértice depósito até um vértice $i \in V \setminus C_0$. Uma rota pode ser extraída adicionando uma aresta que conecta o último vértice do caminho que a *label* representa ao vértice depósito.

Além disso, cada *label* carrega o custo reduzido do caminho, a demanda total e o conjunto $\Pi(P)$ de vértices memorizados para evitar a formação de ciclos.

Cada *label* é definida por uma tupla $L_1 = \langle \zeta_1, i, \xi_1, \Pi_1 \rangle$, e é associado a ela um caminho $P = (\iota_0, \iota_1, \ldots, \iota_s)$, tal que $\iota_0 = 0$. Na tupla $\zeta_1 = \sum_{y=1}^{s} \bar{w}_{\iota_{y-1}\iota_y}$ é o custo reduzido total do caminho; $i = \iota_s$ representa o último vértice visitado no caminho; $\xi_1 = \sum_{y=1}^{s} d_{\alpha(\iota_y)}$ é a demanda total consumida durante o caminho; e Π_1 é o conjunto de *clusters* proibidos para expansão imediata, como definido em (4.21). Além disso, as operações de expansão e dominância são definidas da seguinte forma:

Expansão. Dada uma *label* $L_1 = \langle \zeta_1, i, \xi_1, \Pi_1 \rangle$. Se $\alpha(j) \notin \Pi_1$ e $d_{\alpha(j)} + \xi_1 \leq Q$, então é possível expandir L_1 para uma nova *label* L_2 por meio da adição do arco (i, j), gerando a seguinte *label*:

$$L_2 = \left\langle \zeta_1 + \bar{w}_{ij}, j, \xi_1 + d_{\alpha(j)}, (\Pi_1 \cap NG(\alpha(j))) \cup \{\alpha(j)\} \right\rangle. \tag{4.24}$$

Dominância. Dadas duas *labels* $L_1 = \langle \zeta_1, i, \xi_1, \Pi_1 \rangle$ e $L_2 = \langle \zeta_2, i, \xi_2, \Pi_2 \rangle$, que possuem o mesmo último vértice visitado, diz-se que L_1 domina L_2 se, e somente se,

$$\begin{aligned} \zeta_1 &\leq \zeta_2 \\ \xi_1 &\leq \xi_2 \\ \Pi_1 &\subseteq \Pi_2, \end{aligned} \tag{4.25}$$

e pelo menos uma das condições é estrita. A propriedade de dominância diz que, se L_1 domina L_2, então para qualquer sequência de expansões até um vértice j partindo de L_2 também é possível expandir até j a partir de L_1, tal que as condições (4.25) se mantenham satisfeitas. Portanto, considerando que se trata de um problema de minimização, se L_1 domina L_2, então L_2 não trará nenhum benefício para encontrar a solução ótima e, com isso, pode ser descartada.

O algoritmo de enumeração de *labels* implementado neste trabalho é semelhante ao proposto por [33]. Nessa implementação, tem-se, para cada vértice i e uma demanda p, um *bucket* $F(i, p)$ que contém todas as *labels* nas quais o caminho termina no vértice i e a demanda total consumida é p. O algoritmo de enumeração de *labels* itera em todos os *buckets* e aplica rotinas de expansão e dominância. Como ressaltado por [33], é mais eficiente, em termos de tempo de execução, aplicar regras de dominância somente em *labels* dentro do mesmo *bucket*. Dessa forma, é possível que deixemos de remover algumas *labels* que são dominadas, no entanto, esse não é o caso comum e o custo de verificar a dominância em outros *buckets* não compensa a redução de *labels* dominadas.

A performance geral do algoritmo de enumeração de *labels* (Algoritmo 1) está diretamente relacionada à eficiência da propriedade de dominância, pois quanto maior for a quantidade de *labels* dominadas menor será o número de expansões necessárias. Para as regras de dominância (4.25), quanto menor for a cardinalidade Δ dos **ng**-*sets*, maiores

Algoritmo 1: Algoritmo de enumeração de *Labels*

início
  $F(i,p) \leftarrow \emptyset, \forall i \in V, p \in \{0,\ldots,Q\}$
  $F(0,0) \leftarrow \{(0,0,0,\emptyset)\}$
  para cada $p \in \{1,\ldots,Q\}$ **faça**
    para cada $j \in V \setminus C_0$ **faça**
      se $p < d_{\alpha(j)}$ **então**
        continue
      fim
      para cada $i \in V \setminus C_{\alpha(j)}$ **faça**
        se $\{i,j\} \in E$ **então**
          para cada $L_1 = (\zeta_1, i, p - d_{\alpha(j)}, \Pi_1) \in F(i, p - d_{\alpha(j)})$ **faça**
            se $\alpha(j) \notin \Pi_1$ **então**
              $L_2 \leftarrow (\zeta_1 + \bar{w}_{ij}, j, p, (\Pi_1 \cap NG(j)) \cup \{\alpha(j)\})$
              se L_2 **não dominado qualquer em** $F(j,p)$ **então**
                $D \leftarrow \emptyset$
                para cada $L \in F(j,p)$ **faça**
                  se L_2 **domina** L **então**
                    $D \leftarrow D \cup \{L\}$
                  fim
                fim
                $F(j,p) \leftarrow (F(j,p) \setminus D) \cup \{L_2\}$
              fim
            fim
          fim
        fim
      fim
    fim
  fim
fim

serão as *labels* dominadas, e o contrário acontece para Δ maior. Ou seja, reduzindo o valor de Δ, deixamos o algoritmo mais eficiente, no entanto, geramos rotas com maior probabilidade de terem ciclos e, com isso, reduzimos o limite da relaxação linear da formulação do problema. No Capítulo 5, exploramos essa flexibilidade do algoritmo para encontrar um *tradeoff* entre a eficiência do algoritmo de precificação e a qualidade do limite inferior.

Ao final da execução do algoritmo de expansão de *labels* (Algoritmo 1), tem-se todos os *buckets* com todas as *labels* expandidas e que não foram descartadas pela propriedade de dominância. Com isso, para encontrar a rota de menor custo reduzido, basta verificar para cada *label* $L_1 = \langle \zeta_1, i, \xi_1, \Pi_1 \rangle$ em todos os *buckets* o custo reduzido total da rota

$$\zeta_1 + \bar{w}_{i0}, \tag{4.26}$$

e, então, selecionar a que tem menor custo reduzido total de rota. Se esse valor for

negativo, então temos uma rota candidata a fazer parte da base do PMR. Para recuperar a rota, basta armazenar em cada *label* qual foi a *label* que originou a sua expansão durante o algoritmo de expansão de *labels*. Dessa forma, é possível reconstruir o caminho a partir de qualquer *label*.

Além disso, da mesma forma como foi mencionado para **q**-*routes*, também é possível retornar mais rotas com custo reduzido negativo. Basta verificar para cada *label* dentro dos *buckets* se o custo da rota calculado (4.26) é negativo.

4.5.4 Heurística de Precificação

O algoritmo de expansão de *labels* discutido garante encontrar a rota de menor custo reduzido, se alguma existir. Na geração de colunas, não é necessário encontrar a coluna com o menor custo reduzido, pois qualquer coluna que tenha custo reduzido negativo é suficiente para reotimizar o PMR. Com isso, é possível utilizar heurísticas no problema de precificação para retornar colunas para o PMR em um tempo menor, especialmente durante as primeiras iterações, quando a quantidade de colunas com custo reduzido negativo é maior.

Uma simples, porém efetiva heurística consiste em remover arcos com um alto custo do grafo de custos reduzidos. São realizadas duas etapas de modificação no grafo de custos reduzidos: a primeira consiste em remover 75% dos arcos de maior custo, enquanto na segunda são removidos 50% dos arcos. Após a remoção dos arcos, é então utilizado o algoritmo de expansão de *labels* discutido em §4.5.3. Se durante a primeira etapa pelo menos uma coluna de custo reduzido negativo for encontrada, então o problema de precificação é interrompido. Caso isso não aconteça, a segunda etapa é iniciada, e, da mesma forma, se alguma coluna de custo reduzido negativo for encontrada, o problema é interrompido. Por fim, caso nenhuma das duas etapas retorne alguma coluna de custo reduzido negativo, então o grafo original de custos reduzidos é utilizado no problema de precificação.

4.6 Algoritmo branch-cut-and-price

Um algoritmo *branch-cut-and-price* é um algoritmo *branch-and-bound* em que cada nó da árvore de subproblemas é resolvido utilizando o método de geração de colunas, além

disso são adicionados cortes por meio de um algoritmo de separação para melhorar a qualidade do limite inferior obtido. Nesta Seção, são discutidos a inicialização do algoritmo, como é realizada a ramificação de cada nó e como cada nó é resolvido.

O limite inferior do problema é calculado considerando o menor valor solução de um nó que ainda não foi ramificado, e o limite superior é sempre a menor solução inteira encontrada. O algoritmo termina quando a diferença entre os limites é menor que um *gap* definido *a priori*. Como todas as instâncias utilizadas contêm somente valores inteiros, então o algoritmo termina quando a diferença entre os limites for menor que 1.

4.6.1 Inicialização

A inicialização do algoritmo é a fase responsável por gerar o conjunto inicial de rotas $\mathcal{R}'$ e também por calcular um limite superior para o problema. Após a inicialização, é gerado o nó raiz da árvore de subproblemas utilizando o conjunto inicial de rotas, e a partir de então uma série de subproblemas são resolvidos e particionados em problemas menores.

A fase de inicialização é composta por duas etapas. Na primeira, calculamos um limite superior para o PRVG por meio de uma heurística. Esse valor é utilizado para fazer o cálculo do *gap* caso o algoritmo termine sem encontrar nenhuma solução viável, além de ser utilizado para descartar nós da árvore de subproblemas cujo valor da solução seja maior que o limite superior inicial. Na segunda etapa, é gerado um conjunto de rotas que forma uma base viável para o problema. Esse conjunto é utilizado para inicializar o conjunto $\mathcal{R}'$ de rotas do PMR no método de geração de colunas.

A implementação da heurística que calcula um limite superior para o PRVG é a mesma da discutida em [4]. Os autores utilizaram uma heurística baseada em LNS que consiste em melhorar uma solução inicial ao destruir e reconstruir soluções repetidas vezes. Essa heurística é uma variação similar do que foi proposto em [37].

O conjunto inicial de rotas do PMR é definido por rotas fictícias. Essas rotas têm um custo elevado e são utilizadas somente para inicializar o método de geração de colunas satisfazendo as restrições (4.9)-(4.11). Em alguns casos, são utilizadas heurísticas mais complexas para gerar um conjunto de colunas de baixo custo para tentar reduzir a quantidade de iterações do método de geração de colunas. No entanto, utilizando tanto rotas fictícias quanto rotas calculadas por meio de alguma heurística, não foi possível observar qualquer diferença expressiva na performance do método de geração de colunas, pois a maioria das colunas da base da solução ótima foram geradas pelo algoritmo de precificação.

Para definir um conjunto de rotas fictícias e viáveis, é necessário particionar o conjunto de *clusters*, tal que a soma total das demandas dos *clusters* em cada partição não exceda a capacidade Q de cada veículo. Para encontrar tais partições, resolvemos um problema de empacotamento de caixas. Cada caixa representa uma rota, portanto temos K caixas, e a capacidade máxima de cada caixa representa a capacidade Q de cada veículo. Os itens que devem ser empacotados são representados pelos *clusters* de tal forma que o peso de cada item é o valor da demanda d_u de cada *cluster*. Note que não é necessário saber qual o vértice de cada *cluster* que foi visitado, pois, como se trata de uma rota fictícia, o custo de cada rota não depende de quais vértices foram visitados.

4.6.2 Escolha de um nó para resolver

Dado um conjunto de nós da árvore de subproblemas que ainda não foram resolvidos, deve-se escolher um nó para que seja então resolvido. Adotamos a estratégia de *best-node first*[4], ou seja, o nó que é o melhor candidato a minimizar o valor da função objetivo é resolvido primeiro. Esse nó corresponde àquele que tem o menor valor de limite inferior.

Existem outras estratégias de solução, como *depth first*[5], dedicado a resolver o nó mais profundo da árvore primeiro. Esse caso é geralmente utilizado para encontrar uma solução viável mais rapidamente. No entanto, como já obtemos um limite superior de qualidade utilizando a heurística de LNS, obtivemos resultados melhores escolhendo sempre o nó com menor limite inferior, já que conseguimos reduzir o *gap* mais rapidamente.

4.6.3 Ramificação

Dado um subproblema que foi resolvido e ainda não foi ramificado, se a sua solução não satisfizer a integralidade das rotas (4.11), então é necessário realizar a ramificação do nó em dois ou mais nós.

Seja $\hat{\lambda}_r$ o valor solução da variável λ_r para qualquer $r \in \mathcal{R}'$, então é possível criar um *grafo de solução* no qual os vértices correspondem aos *clusters*, e as arestas indicam o fluxo fracionário da solução entre os *clusters*. Os valores das arestas desse grafo são

[4] do inglês, *melhor nó primeiro*
[5] do inglês, *profundidade primeiro*

calculados da seguinte forma:

$$\hat{y}_{\{v,v\}} = \sum_{e \in \delta(C_u) \cap \delta(C_v)} \sum_{r \in \mathcal{R}'} \hat{\lambda}_r b_e^r \qquad u, v \in M. \tag{4.27}$$

Desse modo, se para qualquer $u, v \in M$ tem-se $\hat{y}_{\{v,v\}} \in \{0, 1\}$, então a solução é inteira e não é necessário ramificação. Caso contrário, é preciso realizar a ramificação do nó em dois ou mais nós filhos. A estratégia de ramificação adotada consiste em selecionar a aresta mais fracionária e, então, criar dois subproblemas: um em que a aresta selecionada é imposta na solução, e outro no qual ela é proibida. A aresta mais fracionária é definida como:

$$\arg\min_{\{u,v\}|u,v \in M} \{|\hat{y}_{\{v,v\}} - 0.5|\}. \tag{4.28}$$

Após selecionada em qual aresta se deve realizar o *branching*, temos de garantir que as rotas do conjunto $\mathcal{R}'$ e que novas colunas geradas pelo algoritmo de precificação satisfaçam as novas condições de *branching*. Para o primeiro caso, basta verificar para cada rota $r \in \mathcal{R}'$ se ela viola ou não a nova regra de *branching* adicionada. Caso ela viole, então simplesmente aumentamos o custo da rota para um valor muito elevado, pois, dessa forma, ao reotimizar o PMR, elas serão naturalmente descartadas da solução. Para o segundo caso, utilizaremos o grafo de custos reduzidos do algoritmo de precificação para evitar a formação de rotas que violem a nova regra de *branching* e também as regras de *branching* dos nós antecessores. Para isso, aplicaremos os seguintes passos de modificação no grafo de custos reduzidos:

1. Para toda aresta $\{u, v\}$ que for proibida na solução, basta remover todos os arcos (i, j) e (j, i) para todo $i \in C_u$ e $j \in C_v$ do grafo de custos reduzidos.

2. Para qualquer vértice l que tiver duas arestas adjacentes impostas na solução $\{l, u\}$ e $\{l, v\}$, atualizamos os valores dos arcos que conectam os vértices $i \in C_u$ e $j \in C_v$ no grafo de custos reduzidos da seguinte forma:

$$\bar{w}_{ij} = \min_{k \in C_l}(\bar{w}_{ik} + \bar{w}_{kj}). \tag{4.29}$$

$$\bar{w}_{ji} = \min_{k \in C_l}(\bar{w}_{jk} + \bar{w}_{ki}). \tag{4.30}$$

E, por fim, removemos todos os vértices $i \in C_u$ do grafo de custos reduzidos e impomos a aresta $\{u, v\}$ na solução do problema. Essa transformação pode ser aplicada enquanto houver arestas impostas e adjacentes uma a outra. No fim, restará nenhuma ou somente arestas impostas que não são adjacentes entre si. E, então, o passo seguinte pode ser executado. A Figura 4.1 ilustra esse tipo de transformação.

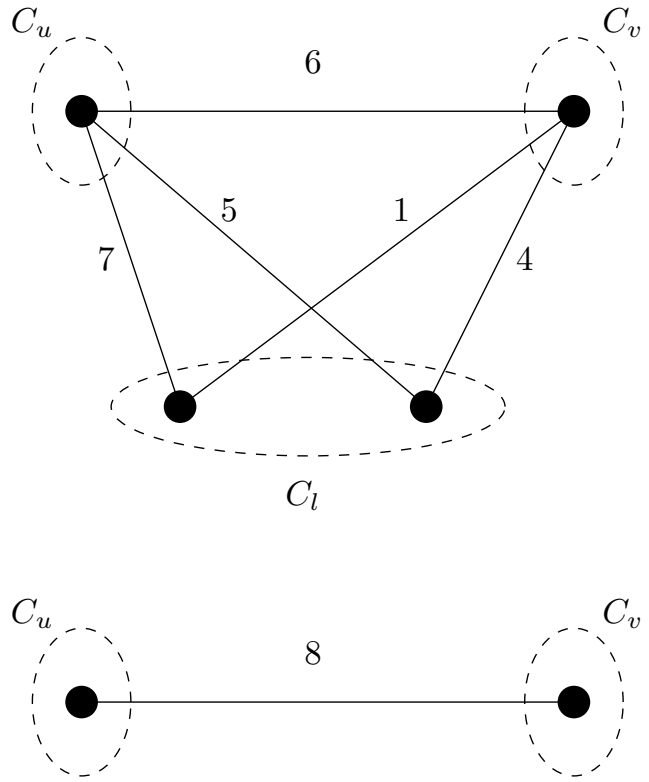


Figura 4.1: Transformação do grafo de custos reduzidos quando as arestas $\{u, l\}$ e $\{v, l\}$ são impostas na solução. Quando atualizados os custos (4.29 e 4.30), então o *cluster* l é removido do grafo e a aresta $\{u, v\}$ passa a ser imposta na solução.

3. Para toda aresta $\{u, v\}$ que for imposta na solução e para qualquer vértice l tal que $u \neq l$ e $v \neq l$, atualizamos os valores dos arcos do grafo de custos reduzidos da seguinte forma:

$$\bar{w}_{kj} = \min_{i \in C_u}(\bar{w}_{ki} + \bar{w}_{ij}) \qquad \forall k \in C_l, \forall j \in C_v \tag{4.31}$$

$$\bar{w}_{ki} = \min_{j \in C_v}(\bar{w}_{kj} + \bar{w}_{ji}) \qquad \forall k \in C_l, \forall i : w \in C_u \tag{4.32}$$

Dessa forma, com os custos atualizados, fazemos com que sempre que um vértice $i \in C_u$ for visitado, então algum vértice $j \in C_v$ é visitado logo em seguida, e vice-versa. A Figura 4.2 ilustra esse tipo de transformação.

Note que, ao realizar a ramificação, toda informação carregada pelo pai é repassada aos filhos. Isso inclui: o conjunto de rotas $\mathcal{R}'$, as arestas que sofreram ramificações etc.

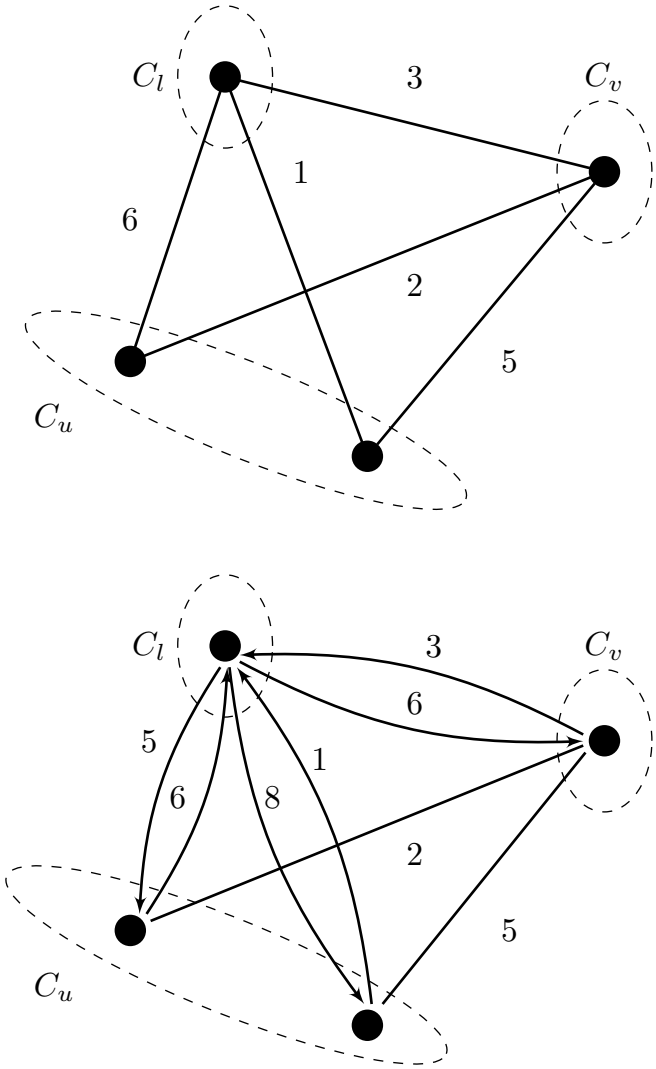


Figura 4.2: Transformação do grafo de custos reduzidos quando a aresta $\{u, v\}$ é imposta na solução. Quando atualizados os custos do grafo reduzido (4.31 e 4.32), então o custo de visitar algum vértice em C_v é equivalente a ter visitado C_u antes, e vice-versa.

4.6.4 Solução do nó

No momento em que um determinado subproblema tiver de ser resolvido, têm-se as seguintes informações: o conjunto de rotas adicionadas ao PMR, o conjunto de cortes gerados para as inequações de *rounded capacity* e uma lista de arestas que foram impostas ou proibidas na solução para esse subproblema.

A solução do subproblema constitui-se de três passos: resolver a formulação linear do PMR, adicionar colunas (ou rotas) candidatas ao PMR por meio de um algoritmo de precificação e adicionar cortes de *rounded capacity* resolvendo um algoritmo de separação. Esses passos são repetidos enquanto pelo menos uma coluna ou um corte for adicionado ao

PMR. Quando nenhum corte ou coluna for adicionado, então o subproblema foi resolvido.

A formulação linear do PMR é resolvida por meio de um *solver*[6]. O algoritmo de separação utilizado para encontrar cortes de *rounded capacity* é o mesmo utilizado na biblioteca de separação disponibilizada por [26]. Os algoritmos de precificação foram discutidos na Seção 4.5 e, no Capítulo 5, analisamos o desempenho do algoritmo *branch-cut-and-price* utilizando os diferentes algoritmos de precificação.

Além disso, adicionamos duas heurísticas durante a solução do subproblema: limitamos a quantidade de colunas adicionadas pelo algoritmo de precificação e reduzimos a quantidade de colunas do PMR a partir de um *threshold*.

Para a primeira, consideramos adicionar no máximo 100 colunas ao PMR por iteração do algoritmo de precificação. Nesse caso, as colunas adicionadas correspondem às de menor custo reduzido. Estimamos esse valor por meio dos experimentos computacionais, tal que não houve nenhuma vantagem adicionando-se mais de 100 colunas, e, quando adicionado menos, o número de iterações do algoritmo de precificação tendia a aumentar.

Para a segunda, percebemos que, à medida que mais colunas são adicionadas ao PMR, maior é o tempo de solução da relaxação linear dele. Por conta disso, gerenciamos as colunas do PMR de tal forma que colunas que não são relevantes para a solução do subproblema possam ser removidas da formulação do PMR. Para isso, em cada subproblema, temos um *pool* de colunas, e toda vez que a quantidade de colunas do PMR ultrapassa um determinado valor, faz-se um *dumping* de algumas colunas para o *pool*. Baseado no tamanho das instâncias que foram utilizadas nos experimentos computacionais, aplicamos o *dumping* quando a quantidade de colunas no PMR for superior a 2000 colunas. Durante o *dumping*, todas as colunas que tiverem um custo reduzido acima de um determinado valor serão adicionadas ao *pool*. Esse *threshold* é uma constante definida da seguinte forma:

$$1.5\ \sigma \min_{e \in E}(w_e), \tag{4.33}$$

tal que σ é uma constante que indica a maior quantidade de *clusters* que podem ser visitados em uma única rota. Essa constante é a mesma aplicada em [10], em que os autores também utilizaram um *pool* de colunas para fazer o gerenciamento de colunas da formulação. Note que o valor em (4.33) é positivo e, com isso, somente removemos colunas que não fazem parte da base da solução da formulação. Esse *pool* de colunas é mantido, pois nas próximas iterações da geração de colunas pode-se calcular o custo reduzido dessas colunas e, caso seja negativo, reinseri-las à formulação sem a necessidade de executar o algoritmo de precificação.

[6]pacote responsável por resolver formulações lineares, geralmente por meio do algoritmo Simplex.

Capítulo 5

Experimentos Computacionais

Neste Capítulo, são apresentados os detalhes dos experimentos computacionais realizados para avaliar a qualidade do algoritmo desenvolvido neste trabalho. Na Seção 5.1, são descritos o ambiente computacional e as instâncias de testes utilizadas durante os experimentos. Na Seção 5.2, comparamos limites inferiores para diferentes algoritmos de precificação que foram discutidos no Capítulo 4. Por fim, na Seção 5.3, mostramos uma visão geral dos resultados obtidos para todas as instâncias e uma análise comparativa com os resultados da literatura.

5.1 Ambiente Computacional

Um dos principais objetivos dos experimentos computacionais realizados é o de fazer um estudo comparativo com os resultados dos dois principais trabalhos publicados sobre o PRVG até então: [4] e [34]. Portanto, usamos em nossos experimentos o mesmo conjunto de instâncias utilizado em ambos os trabalhos. Além disso, realizamos os experimentos em um ambiente computacional com poder de processamento similar.

5.1.1 Instâncias

As instâncias utilizadas durante todos os experimentos computacionais foram propostas por [4] e estão disponíveis em http://www.personal.soton.ac.uk/tb12v07/gvrp.html. Elas foram construídas a partir de instâncias do PRV usando a mesma técnica de agrupamento de vértices utilizada por [13]. As instâncias originais fazem parte da CVRP-library, disponíveis em http://vrp.atd-lab.inf.puc-rio.br/, e são elas cinco grupos de instâncias — grupos A, B, P, M e G — tal que os três primeiros grupos (*ins-*

tâncias médias) — A, B e P — possuem quantidade de vértices variando de 16 a 101, e os dois últimos grupos (*instâncias grandes*) — M e G — possuem quantidade de vértices variando de 101 a 262.

Para qualquer instância do PRV com n vértices, duas instâncias do PRVG foram construídas com $m = \lceil n/\theta \rceil$ *clusters*, tal que $\theta \in \{2, 3\}$. Para a formação dos *clusters*, os m vértices mais distantes entre si são selecionados para ser o centro dos m *clusters*. Então, cada vértice restante é atribuído ao *cluster* cujo vértice central está mais próximo a ele. A demanda de cada *cluster* é calculada como $d_u = \frac{1}{m} \sum_{i \in C_u} \omega_i$, tal que ω_i é a demanda do consumidor i na instância original do PRV. Por fim, um problema de empacotamento é resolvido para determinar o número de veículos para cada instância.

O parâmetro θ indica o valor relativo da quantidade de vértices em cada *cluster*: quanto maior o valor do parâmetro menor será a quantidade de *clusters* e, consequentemente, maior o número de vértices dentro de cada *cluster*. Foram utilizados dois valores distintos para esse parâmetro, portanto, para cada instância do PRV são construídas duas instâncias para o PRVG.

O nome de cada instância segue o seguinte formato X-nY-kZ-CΩ-VΦ, tal que X especifica o grupo de instância (A, B, P, M ou G); Y representa a quantidade de vértices; Z indica a quantidade de veículos na instância original do PRV; Ω representa o total de *clusters*; e Φ indica a quantidade de veículos para instância do PRVG.

5.1.2 Características Consideradas

O algoritmo discutido neste trabalho foi implementado em linguagem C++11 e compilado utilizando o compilador GCC, versão 6.2. A plataforma IBM ILOG CPLEX, versão 12.6, também foi utilizada para resolver a relaxação linear do PMR durante o método de geração de colunas. Por fim, o pacote CVRPSEP ([26]) foi utilizado para gerar os cortes de *rounded capacity*.

Foi permitido somente a utilização de um núcleo do processador durante toda a execução de cada instância. O tempo limite de execução para as instâncias médias — A, B e P — foi de 7,200 segundos (2 horas), e para as instâncias grandes — M e G — foi de 21,600 segundos (6 horas).

A máquina utilizada para realizar todo o processamento durante os experimentos tem um Intel Core i7-980 3.33GHz. O poder de processamento dessa CPU é de 3.35 GFLOP/s. Como referência, as CPUs utilizadas em [4] e [34] têm poder de processamento de 4.8 GFLOP/s e 3.49 GFLOP/s, respectivamente.

Os tempos de processamento e outros resultados dos trabalhos comparados são

reportados em nossos experimentos com os mesmos valores reportados pelos próprios autores. Não foi implementado algoritmo de terceiros para fins de comparação.

5.2 Comparação dos Limites Inferiores

Nesta Seção, comparamos os valores dos limites inferiores de diferentes algoritmos para o PRVG. Consideramos os algoritmos propostos por [4] e [34] e, também, o algoritmo discutido neste trabalho considerando rotas não elementares construídas por meio de **q**-*routes* ou **ng**-*routes*.

Para reduzir a quantidade de informação a ser analisada, consideramos somente instâncias médias cuja quantidade de vértices é de no mínimo 45. Além disso, utilizamos somente o conjunto de instâncias construídas quando $\theta = 2$. O valor do limite inferior é obtido ao resolver somente o primeiro nó da árvore de enumeração do algoritmo *branch-and-bound*.

A Tabela 5.1 mostra a comparação dos valores dos limites inferiores para cada algoritmo. A primeira coluna contém o nome de cada instância. Da segunda coluna à nona coluna, têm-se os valores dos limites inferiores para diferentes algoritmos. Por fim, na décima coluna temos o valor da solução ótima de cada instância como valor de referência comparativa para os limites inferiores.

Tabela 5.1: Comparação de limites inferiores para instâncias médias utilizando $\theta = 2$.

Instância	BC	v_q	$v_{ng(4)}$	$v_{ng(6)}$	$v_{ng(8)}$	$v_{ng(10)}$	$v_{ng(12)}$	BCP	v_{op}
A-n45-k6-C23-V4	595,67	602,21	608,14	608,40	608,40	608,40	608,40	608,40	613
A-n45-k7-C23-V4	630,86	656,64	660,46	661,09	661,98	662,20	662,59	663,21	674
A-n46-k7-C23-V4	573,95	589,79	590,39	590,74	590,74	591,58	591,58	591,58	593
A-n48-k7-C24-V4	630,35	642,56	653,89	653,89	654,12	654,12	654,12	654,12	667
A-n53-k7-C27-V4	589,48	602,05	603,00	603,00	603,00	603,00	603,00	603,00	603
A-n54-k7-C27-V4	665,31	684,83	688,98	689,15	690,00	690,00	690,00	690,00	690
A-n55-k9-C28-V5	668,04	692,82	696,75	698,58	699,00	699,00	699,00	699,00	699
A-n60-k9-C30-V5	750,75	769,00	769,00	769,00	769,00	769,00	769,00	769,00	769
A-n61-k9-C31-V5	621,03	631,79	633,39	634,83	635,00	635,00	635,50	635,50	638
A-n62-k8-C31-V4	722,34	737,68	739,08	740,00	740,00	740,00	740,00	740,00	740
A-n63-k10-C32-V5	759,56	788,70	791,73	792,25	793,25	794,04	794,04	794,04	801
A-n63-k9-C32-V5	864,77	901,86	905,06	906,50	907,00	907,00	907,00	907,00	912
A-n64-k9-C32-V5	733,94	762,73	763,00	763,00	763,00	763,00	763,00	763,00	763
A-n65-k9-C33-V5	665,09	675,76	679,58	681,10	681,21	681,21	681,22	681,22	682
A-n69-k9-C35-V5	648,02	669,87	670,55	670,87	671,33	671,34	672,33	672,49	680
A-n80-k10-C40-V5	916,09	970,09	978,13	980,11	981,68	981,84	982,02	984,22	997
B-n45-k5-C23-V3	496,62	492,61	497,00	497,00	497,00	497,00	497,00	497,00	497
B-n45-k6-C23-V4	466,72	469,74	473,67	474,15	474,50	474,50	474,50	474,50	478
B-n50-k7-C25-V4	446,29	447,25	449,00	449,00	449,00	449,00	449,00	449,00	449
B-n50-k8-C25-V5	890,86	903,48	911,13	912,14	912,14	912,14	912,14	912,14	916
B-n51-k7-C26-V4	650,51	648,82	651,00	651,00	651,00	651,00	651,00	651,00	651

B-n52-k7-C26-V4	450,00	450,00	450,00	450,00	450,00	450,00	450,00	450,00	450
B-n56-k7-C28-V4	483,44	483,62	484,50	484,70	484,70	486,00	486,00	486,00	486
B-n57-k7-C29-V4	748,43	746,50	751,00	751,00	751,00	751,00	751,00	751,00	751
B-n57-k9-C29-V5	933,43	942,00	942,00	942,00	942,00	942,00	942,00	942,00	942
B-n63-k10-C32-V5	806,70	803,88	809,00	809,00	809,00	809,00	809,00	809,00	816
B-n64-k9-C32-V5	507,80	506,93	509,00	509,00	509,00	509,00	509,00	509,00	509
B-n66-k9-C33-V5	802,43	806,86	808,00	808,00	808,00	808,00	808,00	808,00	808
B-n67-k10-C34-V5	663,37	666,02	667,11	667,11	667,11	667,11	667,11	667,11	673
B-n68-k9-C34-V5	700,92	702,00	704,00	704,00	704,00	704,00	704,00	704,00	704
B-n78-k10-C39-V5	791,44	801,00	802,67	803,00	803,00	803,00	803,00	803,00	803
P-n45-k5-C23-V3	330,84	330,24	336,89	337,00	337,00	337,00	337,00	337,00	337
P-n50-k10-C25-V5	377,97	401,16	405,48	405,75	405,75	407,29	407,29	407,36	410
P-n50-k7-C25-V4	337,11	342,75	349,17	350,39	350,49	350,49	350,48	350,49	353
P-n50-k8-C25-V4	342,80	374,54	383,09	385,59	385,71	385,72	385,72	386,00	392
P-n51-k10-C26-V6	405,14	427,00	427,00	427,00	427,00	427,00	427,00	427,00	427
P-n55-k10-C28-V5	387,72	406,95	409,43	411,06	411,94	411,94	411,94	411,94	415
P-n55-k15-C28-V8	508,65	555,00	554,33	555,00	555,00	555,00	555,00	555,00	555
P-n55-k7-C28-V4	342,69	350,17	355,32	355,32	355,44	355,44	355,44	355,44	361
P-n55-k8-C28-V4	347,79	352,50	358,84	359,22	359,22	359,22	359,22	359,22	361
P-n60-k10-C30-V5	406,04	429,40	431,59	434,10	434,44	434,91	434,91	434,91	445
P-n60-k15-C30-V8	522,22	560,57	561,24	564,02	564,75	564,75	564,75	564,75	565
P-n65-k10-C33-V5	461,28	482,70	482,45	483,97	484,69	485,08	485,22	485,22	487
P-n70-k10-C35-V5	468,80	484,44	484,64	485,00	485,00	485,00	485,00	485,00	485
P-n76-k4-C38-V2	374,86	375,05	375,23	379,60	379,60	379,86	380,00	-	411
P-n76-k5-C38-V3	396,56	395,62	395,48	400,25	400,80	400,80	400,91	-	409
P-n101-k4-C51-V2	442,87	442,11	447,24	449,86	450,22	451,10	451,41	-	455

Nas colunas em que estão representados os limites inferiores, têm-se respectivamente: *branch-and-cut* (BC) de [4]; *branch-cut-and-price* utilizando **q**-*routes* (v_q); *branch-cut-and-price* utilizando **ng**-*routes* (v_{ng}) com diferentes tamanhos para o **ng**-*set*; e *branch-cut-and-price* (BCP) utilizando rotas elementares desenvolvido por [34].

Na Tabela 5.2, apresentamos uma análise dos resultados obtidos dos limites inferiores. Nela, podemos observar qual foi o ganho de um algoritmo se comparado a outro. Para isso, consideramos três informações em cada um deles: a quantidade de instâncias resolvidas sem a necessidade de ramificação, o maior *gap* obtido em alguma instância e a média dos *gaps* das instâncias. Consideramos os cálculos de *gap* somente para as instâncias nas quais todos os algoritmos conseguiram encontrar um limite inferior. O *gap* foi calculado fazendo a relação do limite inferior (v_{lb}) e o valor da solução ótima (v_{op}) da instância da seguinte forma:

$$\frac{v_{op} - v_{lb}}{v_{op}}. \tag{5.1}$$

Tabela 5.2: Análise dos resultados de limites inferiores entre diferentes algoritmos para o PRVG.

Algoritmo	Instâncias resolvidas na raiz	*gap* máximo (%)	*gap* médio (%)
[4]	1	12,55	3,60
q-*routes*	5	4,45	1,20

ng-*routes* ($\Delta = 4$)	13	3,01	0,65
ng-*routes* ($\Delta = 6$)	18	2,45	0,53
ng-*routes* ($\Delta = 8$)	20	2,37	0,50
ng-*routes* ($\Delta = 10$)	21	2,27	0,47
ng-*routes* ($\Delta = 12$)	21	2,27	0,47
[34]	21	2,27	0,46

Analisando os valores da Tabela 5.2, podemos observar que tanto o *gap* médio quanto o *gap* máximo tiveram uma redução notável quando comparamos uma implementação de um algoritmo *branch-and-cut* e de um algoritmo *branch-cut-and-price*. Quando comparamos a implementação do algoritmo *branch-cut-and-price* utilizando **q**-*routes* e **ng**-*routes*, também é observada uma melhora significativa nos valores do *gap*, porém, menos expressivas. Já se compararmos os resultados para diferentes tamanhos do **ng**-*set*, quando utilizado **ng**-*routes*, observamos uma redução gradativa dos *gaps*. Essa redução tende a diminuir cada vez que aumentamos o tamanho do **ng**-*set*, de tal forma que, para **ng**-*set* de tamanhos 10 ou 12, a diferença dos *gaps*, se comparada com a implementação utilizando rotas elementares de [34], é muito pequena.

Além dos experimentos de limites inferiores, verificamos o comportamento das implementações do algoritmo *branch-cut-and-price* utilizando **q**-*routes* e **ng**-*routes* para resolver todas as instâncias propostas por [4]. De certa forma, o tempo de solução das instâncias foi coerente com a análise dos limites inferiores, de tal forma que, com um limite inferior maior, foi possível resolver uma maior quantidade de instâncias dentro do tempo limite. No entanto, à medida que as rotas encontradas pelo algoritmo de precificação tendem a ficar mais próximas da elementariedade, ou seja, quando aumentamos muito o valor do **ng**-*set*, então o custo computacional não supera o ganho de um limite inferior maior. Isso faz com que o tempo de solução de instâncias maiores aumente ou até mesmo impeça algumas instâncias de serem resolvidas dentro do tempo limite. Durante os nossos experimentos, observamos que um **ng**-*set* de tamanho 10 foi o caso em que obtivemos melhores resultados, superando um **ng**-*set* de tamanho 12 e até mesmo o algoritmo de [34] que utiliza rotas elementares. Os resultados desse experimento estão detalhados na Seção 5.3.

5.3 Desempenho do Algoritmo *Branch-cut-and-price*

Nesta Seção, discutimos os resultados obtidos ao tentar resolver as instâncias médias — A, B e P — e grandes — M e G — propostas por [4].

Como discutido neste trabalho, a metodologia apresentada envolve algumas variações do algoritmo de precificação, sendo que todas exploram diferentes formas de gerar

rotas não elementares. Em nossos experimentos, tentamos resolver todas as instâncias utilizando **q**-*routes* ou **ng**-*routes* durante o algoritmo de precificação. Nessa última ainda é possível variar o tamanho do **ng**-*set*. Consideramos os mesmos valores de **ng**-*set* utilizados nos experimentos de limites inferiores, discutidos na Seção 5.2.

O caso em que obtivemos o melhor resultado, ou seja, o maior número de instâncias resolvidas, foi utilizando **q**-*routes* com um **ng**-*set* de tamanho 10 durante o algoritmo de precificação. Os resultados estão detalhados nas tabelas A.1 e A.2, apresentadas na seção de apêndice. Nessas tabelas, a coluna *UB* representa o melhor limite superior encontrado para a instância; *LB* corresponde ao limite inferior obtido; *BB* indica a quantidade de nós resolvidos durante o *branch-and-bound*; e *CPU* representa o tempo total em segundos para a execução completa do algoritmo com aquela instância. No caso em que utilizamos um **ng**-*set* maior, há uma pequena melhora do limite inferior, no entanto, o tempo necessário para encontrar rotas válidas no algoritmo de precificação aumenta de forma mais significativa. Para o caso que reduzimos o valor do **ng**-*set* abaixo de 10, foi possível encontrar rotas válidas em um menor tempo, no entanto, a quantidade de nós a serem resolvidos aumentou de tal forma que algumas instâncias não puderam ser resolvidas dentro do tempo limite. Por fim, utilizando **q**-*routes*, o resultado foi bem inferior se comparado a quando utilizamos **ng**-*routes*, devido ao limite inferior significativamente menor, como observado na Seção 5.2.

Tabela 5.3: Sumário dos resultados reportados nas tabelas A.1 e A.2.

		[4]				[34]				Metodologia Proposta			
Tipo	θ	$\overline{CPU}$	$\overline{BB}$	NS	$\overline{gap}$	$\overline{CPU}$	$\overline{BB}$	NS	$\overline{gap}$	$\overline{CPU}$	$\overline{BB}$	NS	$\overline{gap}$
A	2	861	1.094	2	0,49	347	9	1	0,05	70	33	0	0
B	2	159	400	0	0,00	299	5	0	0	9	4	0	0
P	2	1.414	1.653	4	1,55	955	11	3	-	176	12	0	0
M	2	16.823	3.179	3	2,96	-	-	-	-	16.221	239	3	0,57
G	2	21.606	1	1	11,87	-	-	-	-	21.600	154	1	1,92
A	3	804	1.291	2	0,78	32	3	0	0	13	5	0	0
B	3	34	270	0	0,00	55	7	0	0	14	8	0	0
P	3	713	1.772	1	0,48	917	5	3	-	157	10	0	0
M	3	13.058	6.770	2	2,63	-	-	-	-	10.872	120	1	0,48
G	3	21.602	181	1	15,09	-	-	-	-	21.600	262	1	1,85

Analisando os resultados das tabelas A.1 e A.2, observamos que os resultados foram significativamente melhores se comparados aos obtidos em [4] e [34]. De tal forma que, com o nosso algoritmo, foi possível resolver todas as instâncias médias e, com isso, resolvendo a única instância média — A-n80-k10-C40-V5 — que não havia sido resolvida até então. Além disso, tivemos uma redução no valor do *gap* nas instâncias grandes.

A Tabela 5.3 sumariza os nossos resultados comparados com os dos autores citados. Para cada tipo e grupo de instâncias, são apresentados: o tempo médio ($\overline{CPU}$) de execução do algoritmo, a quantidade média de nós ($\overline{BB}$) resolvidos durante o *branch-and-bound*, a quantidade de instâncias não resolvidas (NS) e o *gap* médio ($\overline{gap}$). É possível observar

que em todos os conjuntos de instâncias obtivemos *gaps* menores, tempo médio de solução menor e menos instâncias não resolvidas. Vale ressaltar que, para as instâncias grandes — M e G —, uma instância que não havia sido resolvida por [4] foi resolvida utilizando o nosso algoritmo e, além disso, o *gap* para essas instâncias foi reduzido substancialmente.

Por fim, do grupo de instâncias propostas por [4] para o PRVG, *restam seis instâncias* dos tipos M e G que ainda não foram resolvidas.

Capítulo 6

PRVG aplicado a veículos não-holonômicos

Neste Capítulo, discutimos sobre uma variação de um PRV presente em um contexto da literatura robótica. Nessa variação, são utilizados veículos com características não-holonômicas de movimento e visitas em regiões de interesse. Recentemente, esse tipo de problema teve maior destaque devido aos avanços tecnológicos em veículos não tripulados, sendo eles terrestres, marítimos ou aéreos (veja [32]). Na literatura, encontram-se diversas propostas de soluções, em sua maioria heurísticas, que resolvem variações similares, a exemplo de [25], [19] e [29].

Nas seções seguintes, demonstramos como essa variação de um PRV e o PRVG podem se relacionar, sendo, dessa forma, a motivação inicial para o desenvolvimento da metodologia sobre o PRVG apresentada neste trabalho. Essa relação é feita por um processo de discretização utilizado para transformar o problema, originalmente não linear, em um problema alternativo, porém linear, e que pode ser modelado como um PRVG.

Por fim, discutimos como, no contexto da literatura robótica, os problemas de roteamento geralmente utilizam uma função objetivo que não é comum nos problemas mais clássicos de roteamento na literatura combinatória. Com isso, a proposta de modelagem por meio de um PRVG torna-se menos atrativa no contexto robótico, mas, ainda assim, importante do ponto de vista metodológico como uma solução exata para esse tipo de problema.

6.1 Veículos não-holonômicos e problemas de roteamento relacionados

A coordenação de vários agentes móveis (neste caso, veículos ou robôs) é tema de grande importância na Robótica Móvel, sendo o foco de diversos trabalhos na literatura.

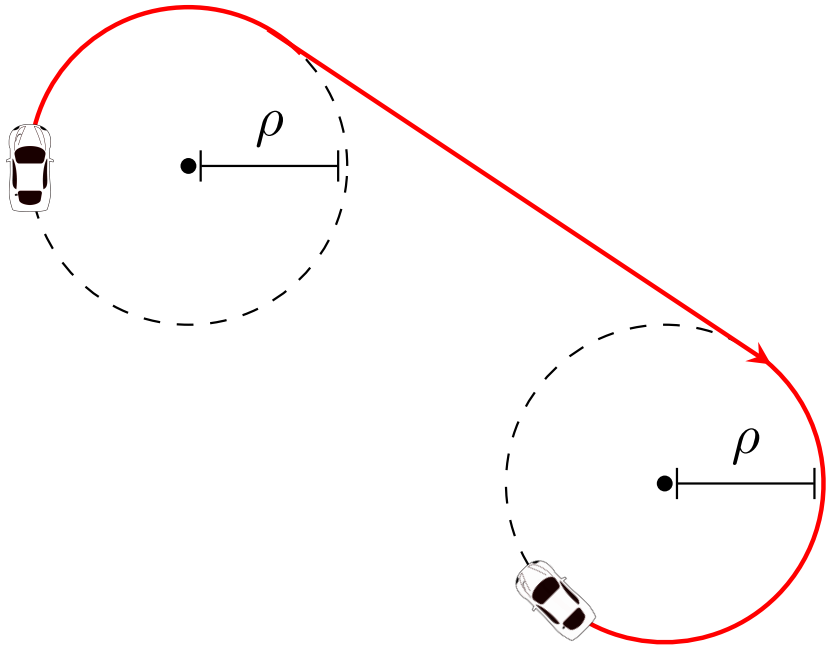
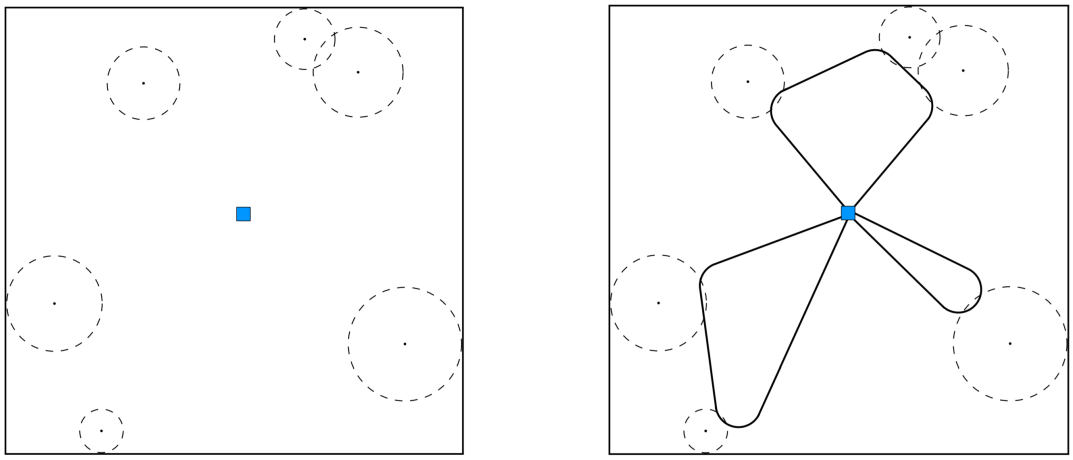


Figura 6.1: Exemplo de um trajeto de um veículo não-holonômico entre duas posições conhecidas, tal que o ângulo de orientação do veículo é conhecido nas posições inicial e final. A trajetória do veículo é definida por uma *curva de Dubin* ([12]) e leva em consideração o raio de curvatura mínimo ρ do veículo.

Isso se deve ao fato de que múltiplos veículos são capazes de executar uma tarefa de maneira mais eficiente, permitindo delegar funções a cada um deles. Além disso, o sistema se torna mais robusto a falhas individuais. É interessante observar que a maioria dos veículos presentes em nosso cotidiano (por exemplo, automóveis, motocicletas, aviões e navios) apresenta determinadas restrições associadas aos movimentos que podem realizar. Veículos com essa característica são denominados *veículos não-holonômicos* (Figura 6.1). Ao utilizar-se desse tipo de veículo para realização de uma tarefa, o custo associado à execução do caminho poderá ser bem maior do que a simples distância euclidiana linear entre as posições inicial e final, variando de acordo com as características específicas de cada veículo. Por exemplo, veículos marítimos geralmente apresentam um raio de curvatura maior durante o seu trajeto se comparado a um veículo terrestre, portanto, terão um impacto maior no custo total de um trajeto simplificado pelo somatório das distâncias euclidianas entre os pontos de interesse.

Um veículo não-holonômico que realiza um trajeto em velocidade constante entre duas posições conhecidas e com ângulos de orientações do veículo definidos nas posições inicial e final tem a sua trajetória definida pela chamada *curva de Dubins* (Figura 6.1), apresentada em [12]. O comprimento de uma curva de Dubins pode ser calculado com complexidade polinomial ([12]), com isso, veículos não-holonômicos podem ser considera-



(a) **(b)**

Figura 6.2: Ilustração de um problema de roteamento de veículos não-holonômicos com visita a regiões de interesse ([29]). (a) Disposição inicial de algumas regiões (círculos tracejados) e do depósito (quadrado azul). (b) Solução com rotas visitando todas as regiões de interesse.

dos em problemas de roteamento de veículos, e as distâncias entre os pontos de visita são definidas por uma função de curva de Dubins.

Além disso, em particular na área de robótica, problemas de roteamento de veículos não-holonômicos geralmente são aplicados quando existem *regiões de interesses* a serem visitadas, e.g., sensores de redes sem fio ([32]). Nesse tipo de problema, não existe um ponto específico a ser visitado, mas, sim, regiões em um espaço bidimensional (Figura 6.1). Portanto, nesse tipo de problema, além das restrições e do objetivo de um clássico problema de roteamento de veículos, devem ser considerados: o percurso não euclidiano dos trajetos veiculares e os múltiplos possíveis pontos de visita às áreas de interesse.

6.2 Roteamento de veículos não-holonômicos na literatura

Nesta Seção, discutimos brevemente sobre os recentes trabalhos da literatura que obtiveram destaque em otimização de roteamento de veículos não-holonômicos. Nesses trabalhos, sempre é considerada alguma variação do PCV clássico. Esse tipo de variação pode ser característica de uma aplicação específica ou dentro de um contexto, que, no caso da robótica, é bastante utilizado para expressar situações que envolvem veículos não-holonômicos, regiões de interesse de visita e multiveículo.

[39]. Os autores propuseram um *Algoritmo de Alternação* que transforma uma solução clássica do PCV em uma solução do PCV para veículos não-holonômicos. Nesse algoritmo, dada uma solução clássica do PCV, alterna-se entre a sequência de arestas que compõem o caminho solução para impor as orientações do veículo nos vértices dessas arestas. E, por fim, é calculado o comprimento do caminho de Dubins utilizando o mesmo caminho da solução inicial, porém considerando as orientações impostas.

[25]. Nesse trabalho, os autores apresentaram um algoritmo que retorna uma solução subótima do PCV para veículos não-holonômicos por meio de um processo de discretização do ângulo de orientação do veículo. Com esse processo, é apresentado uma formulação linear e inteira baseada em fluxo em arcos para um dado conjunto de ângulos de orientações. Como ressaltado pelos autores, utilizar essa formulação para encontrar a solução ótima do problema demonstrou-se viável somente para instâncias consideradas pequenas (até 30 vértices e 32 ângulos de orientações para cada vértice). Para instâncias maiores, os autores transformaram o problema discretizado em um PCVG e, em seguida, em um Problema do Caixeiro Viajante Assimétrico (PCVA), que pode ser resolvido com algoritmos existentes na literatura. O método de discretização foi comparado com a solução do Algoritmo de Alternação e demonstrou um valor do comprimento da rota de solução significativamente menor na maioria dos casos, mesmo que para um número relativamente baixo de discretização (em torno de 4 ângulos de orientações por vértice).

[19]. Nesse trabalho, os autores trabalharam com o PCV para veículos holonômicos e com *vizinhança*. O conceito de vizinhança define regiões de interesse que devem ser visitadas pelo veículo, em vez de pontos de visita se comparado ao modelo clássico. Os autores apresentaram uma solução semelhante ao que foi proposto em [25], no entanto, o conceito de discretização não foi explicitamente definido, mas, sim, um *sample* de configurações do veículo que projetam a sua posição e orientação nas regiões de interesse. Além disso, os autores abordam casos em que existe sobreposição das regiões de interesse e apresentam estratégias de solução que consideram esse tipo de situação.

[29]. Um dos primeiros trabalhos a tratar multiveículos em um PCV para veículos não-holonômicos e com vizinhança. Nesse trabalho, os autores apresentaram dois algoritmos: no primeiro são utilizadas técnicas já existentes na literatura, como o Algoritmo de Alternação; e no segundo é apresentado um algoritmo memético que resolve o problema. Quando comparadas as duas propostas, o segundo algoritmo supera o primeiro, conseguindo rotas em torno de 20% mais curtas.

6.3 Problemas de roteamento de veículos não-holonômicos apresentados como um PRVG

Como discutido neste Capítulo, um problema de roteamento de veículos não-holonômicos com regiões de interesse é uma variação de um PRV clássico de tal forma que o custo entre os pontos de interesse passa a não ser mais definido por uma distância euclidiana, e sim por uma função

$$\mathcal{D} : (\mathbb{R}^2, [0, 2\pi[) \times (\mathbb{R}^2, [0, 2\pi[) \rightarrow \mathbb{R}^+, \tag{6.1}$$

que define a menor curva de Dubins de um veículo não-holonômico entre duas posições em um plano, considerando as orientações inicial e final. Por conta disso, essa variação do problema impõe uma função objetivo definida por uma função gradiente não linear e, consequentemente, o espaço de solução deixa de ser obrigatoriamente convexo, impossibilitando o uso de métodos exatos geralmente utilizados em modelos lineares para roteamento de veículos. Como resultado, roteamento de veículos não-holonômicos com regiões de interesse atrai diversos estudos que utilizam algoritmos heurísticos para encontrar uma solução subótima para o problema, a exemplo de [5] e [29], uma vez que uma solução exata ainda é bastante inviável na prática.

6.3.1 Formulação linear para um problema mais restrito

Como alternativa às metodologias heurísticas para o problema de roteamento de veículos não-holonômicos com regiões de interesse, investigamos uma solução exata para esse problema que utiliza a metodologia apresentada neste trabalho para o PRVG. O método exato que propomos não garante encontrar a solução ótima do problema definido na seção anterior, mas, sim, a solução ótima de um problema cujo espaço de solução é um subconjunto do espaço de solução do problema original. Apesar disso, os resultados obtidos podem ser bastante satisfatórios para fins práticos.

Nessa proposta, restringimos o problema original de tal forma que o novo espaço de solução seja convexo, no entanto, sem a possibilidade de garantir que a solução ótima do problema original faça parte desse subespaço. Ou seja, o que estamos propondo é um método exato que resolve um problema alternativo e mais restrito.

Para gerar o problema alternativo, que possui um espaço de solução convexo, tornamos finito (discretização finita) o conjunto que define as possíveis posições e orientações

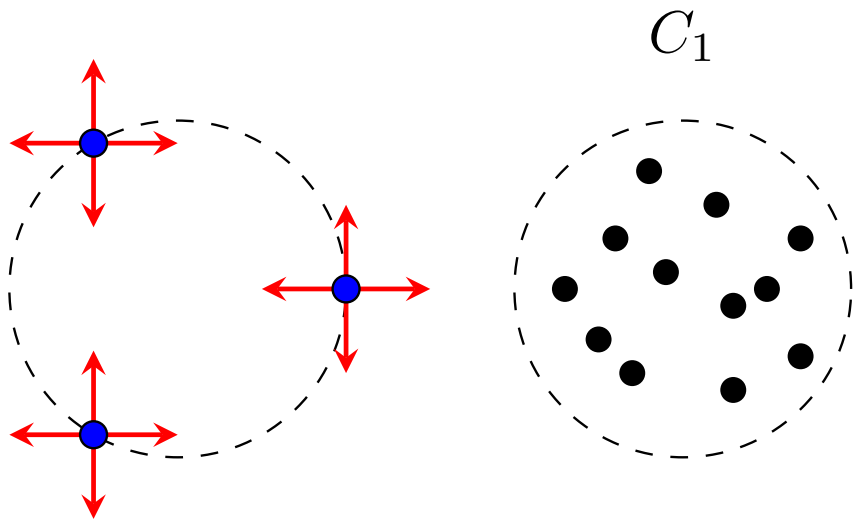


Figura 6.3: Ao lado esquerdo, tem-se uma representação visual de um exemplo de discretização do conjunto de configurações de um veículo. Representadas pelos círculos em azul as possíveis posições de um veículo dentro da região de interesse, e representadas pelas setas em vermelho as possíveis orientações de um veículo naquela posição. Ao lado direito, tem-se a representação abstrata dos vértices e do *cluster* formado para uma instância do PRVG.

dos veículos em cada região de interesse, a exemplo do ilustrado na Figura 6.3. Dessa forma, em cada região de interesse temos um número finito de configurações (posição e orientação) possíveis que um veículo pode ter em algum momento. Como qualquer uma dessas configurações corresponde a uma região de interesse específica sendo atendida por um veículo, então é possível modelar o problema utilizando esses conjuntos finitos como um PRVG, tal que cada região de interesse corresponde a um *cluster* e cada possível configuração de uma região representa um vértice nesse *cluster*. Um processo de discretização similar foi apresentada em [25] para o PCV com veículos não-holonômicos.

Portanto, nesse PRVG, além das definições de capacidade dos veículos e demandas a serem atendidas em cada região de interesse, tem-se cada *cluster* representando uma região de interesse e cada vértice representando uma configuração que pertence a $(\mathbb{R}^2, [0, 2\pi[)$. Por fim, utilizando a equação (6.1), é possível construir a matriz de custos entre os pares de vértices do PRVG.

6.3.2 Critérios de discretização

A transformação do problema original em problema alternativo restrito impõe um processo de discretização do conjunto que define as possíveis configurações (posição e orientação) de um veículo em cada região de interesse. Esse processo permite definirmos um critério arbitrário responsável por definir o subconjunto das possíveis configurações de posição e orientação em cada região de interesse. Obviamente que, qualquer que seja o critério utilizado, quanto maior for a quantidade de elementos definidos no subconjunto, a tendência é obter um valor de solução ótima do problema restrito mais próximo do valor de solução ótima do problema original. No entanto, quando definimos mais configurações possíveis em cada região de interesse, também aumentamos o tamanho da instância do PRVG, pois o número de vértices é maior. Ou seja, o critério utilizado influencia no valor de solução do problema restrito para uma instância de mesmo tamanho.

Geralmente, em problemas de roteamento em que existem regiões com demandas a serem atendidas, considera-se uma região atendida no momento em que um veículo tangencia qualquer ponto da região. Portanto, é suficiente definir as posições das possíveis configurações de cada região de interesse somente na borda da região.

O critério utilizado para definir as configurações de cada região de interesse é simples, porém, no geral, atende o caso médio de instâncias diversificadas. Define-se uma distribuição uniforme das possíveis posições de um veículo ao longo da circunferência de cada região de interesse (para o caso de regiões de interesse circulares) e também uma distribuição uniforme das possíveis orientações dentro do conjunto $[0, 2\pi[$. A exemplo da Figura 6.3, tem-se uma distribuição uniforme de três pontos de visita e de quatro orientações, nesse caso, totalizando 12 possíveis configurações de visita dentro da região. Esse processo possibilita ter configurações nas regiões de interesse que atendam a veículos vindos de qualquer direção e sentido, evitando o pior caso.

Além disso, é possível incluir algoritmos heurísticos simples para obter melhores definições de configurações em alguns *corner cases*. Por exemplo, regiões de interesse muito distantes de outras regiões de interesse podem ter as posições e orientações definidas de tal forma que favoreçam veículos vindo de uma específica direção e sentido. No entanto, neste trabalho, limitamo-nos à distribuição uniforme sem a utilização de algoritmos heurísticos.

6.3.3 Limitações por uma modelagem PRVG

Geralmente, em problemas de roteamento de veículos robóticos, o interesse da função objetivo corresponde a realizar toda a tarefa no menor tempo possível ou a minimizar o consumo de combustível em cada robô. Ou seja, nesse tipo de função objetivo, tem-se a minimização da rota de maior custo, a exemplo de [29]. Em um modelo linear baseado em geração de colunas, como apresentado na Seção 4.4, esse tipo de função objetivo é modelado através de inequações para cada rota do problema mestre restrito. Ou seja, para toda coluna r que for gerada para o problema mestre restrito, também deve ser adicionada uma restrição $c_r\lambda_r \leq T$, tal que T passa a ser a variável a ser minimizada. Nesse tipo de modelo, a convergência de um método de geração de colunas perde eficiência e a metodologia apresentada no Capítulo 4 deste trabalho passa a ser pouco eficiente na prática.

Capítulo 7

Conclusão

Neste trabalho, foi abordado o Problema de Roteamento de Veículos Generalizado (PRVG). Na literatura, são encontrados dois trabalhos que apresentam um algoritmo prático para resolver o problema. Em [4], é apresentado um algoritmo *branch-and-cut*; e em [34], é apresentado um algoritmo *branch-cut-and-price*. O segundo demonstrou uma grande vantagem em relação ao limite da relaxação linear do problema, no entanto, o alto custo do algoritmo de precificação fez com que, na prática, o algoritmo não fosse tão vantajoso em relação ao primeiro. A fim de explorar novas estratégias de solução, propusemos um novo algoritmo *branch-cut-and-price*, tal que nossa estratégia corresponde a não impor a elementariedade da rota durante o algoritmo de precificação. Dessa forma, reduzimos o *overhead* do algoritmo de precificação, porém perdemos um pouco a qualidade do limite inferior.

Em nossa proposta, durante o algoritmo de precificação, utilizamos os seguintes tipos de rotas: **q**-*routes* e **ng**-*routes*. Nesta última, escolhemos diferentes tamanhos limite do conjunto **ng**-*set*. Analisando os limites inferiores, observamos que o ganho no valor do limite inferior foi relevante somente até um **ng**-*set* de tamanho 8 ou 10. Após isso, o ganho se tornava mínimo ou inexistente. Por fim, tentamos resolver todas as instâncias propostas por [4] utilizando tanto **q**-*routes* quanto **ng**-*routes*. De todos os casos, observamos melhores ganhos utilizando **ng**-*routes* com um **ng**-*set* de tamanho 10. No geral, em nossos experimentos utilizando **ng**-*routes*, tivemos limites inferiores e tempos de solução melhores se comparados com **q**-*routes*.

Comparando com os resultados da literatura, a nossa proposta reduziu significativamente o tempo para resolver as instâncias propostas por [4]. Nosso algoritmo resolveu duas instâncias que estavam em aberto, além de reduzir o tempo de solução da maioria das instâncias que foram resolvidas. Por fim, das 158 instâncias do PRVG, [4] não resolveram 16 delas; [34] não resolveram 17; e no nosso trabalho não resolvemos 6 instâncias. As instâncias não resolvidas fazem parte do conjunto *grande* de instâncias propostas e permanecem não resolvidas por nenhum algoritmo ainda proposto.

As instâncias que não puderam ser resolvidas em nenhum trabalho publicado até o momento seguem como motivação para trabalhos futuros relacionados ao PRVG. Dentro do contexto de um algoritmo *branch-cut-and-price* e a exemplo da metodologia adotada

neste trabalho, ainda existem diversos resultados na literatura do PRV clássico que podem ser estendidos ou aplicados diretamente ao PRVG, a exemplo de [35], e que são possíveis potenciais para reduzir o tempo do algoritmo de precificação para as instâncias maiores que não foram resolvidas.

Por fim, dentro do contexto robótico que foi discutido no Capítulo 6, apesar de não ter sido possível uma aplicação direta do PRVG clássico, a discretização proposta permite modelar o problema como uma variação do PRVG que tem uma função objetivo *min-max* do comprimento das rotas. Dessa forma, assim como diversas metodologias propostas originalmente para o PRV e que foram adaptadas para o PRVG, também existe oportunidade para estender estudos da literatura do *PRV min-max* para uma variação *generalizada*.

Referências Bibliográficas

[1] R. Baldacci, E. Bartolini, and G. Laporte. Some applications of the generalized vehicle routing problem. *Journal of the Operational Research Society*, 61(7):1072–1077, Jul 2010.

[2] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1 – 6, 2011.

[3] M. L. Balinski and R. E. Quandt. On an integer program for a delivery problem. *Oper. Res.*, 12(2):300–304, April 1964.

[4] Tolga Bektaş, Güneş Erdoğan, and Stefan Røpke. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, 45(3):299–316, August 2011.

[5] Deepak Bhadauria, Onur Tekdas, and Volkan Isler. Robotic data mules for collecting data over sparse sensor fields. *Journal of Field Robotics*, 28(3):388–404, 2011.

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

[7] Benoit Crevier, Jean-François Cordeau, and Gilbert Laporte. The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, 176(2):756 – 773, 2007.

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

[9] George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[10] Mauro Dell'Amico, Giovanni Righini, and Matteo Salani. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2):235–247, 2006.

[11] Feillet Dominique, Dejax Pierre, Gendreau Michel, and Gueguen Cyrille. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Netw.*, 44(3):216–229, October 2004.

[12] L. E. Dubins. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[13] Matteo Fischetti, Juan José Salazar González, and Paolo Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997.

[14] L. R. Ford and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 5(1):97–101, 1958.

[15] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, May 2006.

[16] B. Gavish and S.C. Graves. *The Traveling Salesman Problem and Related Problems.* Working paper: Graduate School of Management. University of Rochester, Graduate School of Management, 1979.

[17] Gianpaolo Ghiani and Gennaro Improta. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research*, 122(1):11 – 17, 2000.

[18] Stefan Irnich and Guy Desaulniers. *Shortest Path Problems with Resource Constraints*, pages 33–65. Springer US, Boston, MA, 2005.

[19] Jason Isaacs and Joao Hespanha. Dubins traveling salesman problem with neighborhoods: A graph-based approach. *Algorithms*, 6:84–99, 02 2013.

[20] Imdat Kara and Tolga Bektas. Integer linear programming formulation of the generalized vehicle routing problem. In *EURO/INFORMS Joint International Meeting, Istanbul, July*, pages 06–10, 2003.

[21] Imdat Kara, Gilbert Laporte, and Tolga Bektas. A note on the lifted miller–tucker–zemlin subtour elimination constraints for the capacitated vehicle routing problem. *European Journal of Operational Research*, 158(3):793 – 795, 2004.

[22] Gilbert Laporte, François V. Louveaux, and Luc van Hamme. An integer l-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Oper. Res.*, 50(3):415–423, May 2002.

[23] Gilbert Laporte and Yves Nobert. Generalized travelling salesman problem through n sets of nodes: An integer programming approach. *INFOR: Information Systems and Operational Research*, 21(1):61–75, 1983.

[24] Gilbert Laporte, Yves Nobert, and Serge Taillefer. Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation Science*, 22(3):161–172, 1988.

[25] J. Le Ny, E. Feron, and E. Frazzoli. On the dubins traveling salesman problem. *IEEE Transactions on Automatic Control*, 57(1):265–270, jan. 2012.

[26] Jens Lysgaard. Cvrpsep: A package of separation routines for the capacitated vehicle routing problem, 2003.

[27] Jens Lysgaard, Adam N. Letchford, and Richard W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, Jun 2004.

[28] Marius M. Solomon and Jacques Desrosiers. Time window constrained routing and scheduling problems. *Transportation Science*, 22(1):1–13, February 1988.

[29] D. G. Macharet, A. Alves Neto, V. F. da Camara Neto, and M. F. M. Campos. Efficient target visiting path planning for multiple vehicles with bounded curvature. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3830–3836, Nov 2013.

[30] Rafael Martinelli, Diego Pecin, and Marcus Poggi. Efficient elementary and restricted non-elementary route pricing. *European Journal of Operational Research*, 239(1):102 – 111, 2014.

[31] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, October 1960.

[32] K. J. Obermeyer. Path planning for a UAV performing reconnaissance of static ground targets in terrain. In *Proceedings of the AIAA Conference on Guidance, Navigation and Control*, Chicago, IL, USA, August 2009.

[33] Diego Pecin, Artur Pessoa, Marcus Poggi, and Eduardo Uchoa. *Improved Branch-Cut-and-Price for Capacitated Vehicle Routing*, pages 393–403. Springer International Publishing, Cham, 2014.

[34] Mohammad Reihaneh and Ahmed Ghoniem. A branch-cut-and-price algorithm for the generalized vehicle routing problem. *Journal of the Operational Research Society*, Apr 2017.

[35] Giovanni Righini and Matteo Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research*, 36(4):1191 – 1203, 2009.

[36] R. Roberti. *Exact Algorithms for Different Classes of Vehicle Routing Problems*. PhD thesis, University of Bologna, 2012.

[37] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

[38] M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995.

[39] Ketan Savla, Emilio Frazzoli, and F. Bullo. On the point-to-point and traveling salesperson problems for dubins' vehicle. volume 2, pages 786 – 791 vol. 2, 07 2005.

[40] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*, CP '98, pages 417–431, London, UK, UK, 1998. Springer-Verlag.

[41] Paolo Toth, Daniele Vigo, Paolo Toth, and Daniele Vigo. *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2014.

Apêndice A

Resultados Gerais

Tabela A.1: Resultados computacionais para instâncias geradas utilizando $\theta = 2$.

		[4]			[34]			Metodologia Proposta		
Instância	UB	LB	BB	CPU	LB	BB	CPU	LB	BB	CPU
A-n32-k5-C16-V2	519	519.00	1,847	113	519.00	3	39	519.00	3	3
A-n33-k5-C17-V3	451	451.00	38	2	451.00	1	3	451.00	1	2
A-n33-k6-C17-V4	465	465.00	5	1	465.00	1	1	465.00	1	1
A-n34-k5-C17-V3	489	489.00	5	1	489.00	1	4	489.00	1	1
A-n36-k5-C18-V2	505	505.00	612	32	505.00	1	21	505.00	3	5
A-n37-k5-C19-V3	432	432.00	1	1	432.00	1	24	432.00	1	2
A-n37-k6-C19-V3	584	584.00	264	28	584.00	1	6	584.00	1	1
A-n38-k5-C19-V3	476	476.00	27	3	476.00	1	6	476.00	1	0
A-n39-k5-C20-V3	557	557.00	544	46	557.00	1	12	557.00	63	38
A-n39-k6-C20-V3	544	544.00	42	5	544.00	1	7	544.00	1	1
A-n44-k6-C22-V3	608	608.00	210	23	608.00	1	7	608.00	1	1
A-n45-k6-C23-V4	613	613.00	112	7	613.00	9	35	613.00	5	7
A-n45-k7-C23-V4	674	674.00	3,184	1,465	674.00	53	141	674.00	105	66
A-n46-k7-C23-V4	593	593.00	43	10	593.00	3	29	593.00	3	5
A-n48-k7-C24-V4	667	667.00	1,829	300	667.00	69	147	667.00	111	77
A-n53-k7-C27-V4	603	603.00	40	16	603.00	1	69	603.00	1	7
A-n54-k7-C27-V4	690	690.00	372	68	690.00	1	36	690.00	1	5
A-n55-k9-C28-V5	699	699.00	577	83	699.00	1	13	699.00	1	3
A-n60-k9-C30-V5	769	769.00	215	76	769.00	1	48	769.00	1	3
A-n61-k9-C31-V5	638	638.00	243	44	638.00	5	57	638.00	3	8
A-n62-k8-C31-V4	740	740.00	210	123	740.00	1	181	740.00	1	12
A-n63-k10-C32-V5	801	801.00	5,430	4,355	801.00	29	216	801.00	37	42
A-n63-k9-C32-V5	912	*900.29*	4,749	7,200	912.00	11	275	912.00	9	27
A-n64-k9-C32-V5	763	763.00	1,831	1,204	763.00	1	333	763.00	1	4
A-n65-k9-C33-V5	682	682.00	54	29	682.00	3	83	682.00	1	7
A-n69-k9-C35-V5	680	680.00	2,569	818	680.00	29	364	680.00	47	119
A-n80-k10-C40-V5	997	*957.36*	4,487	7,200	*984.22*	1	7,200	997.00	483	1,440
B-n31-k5-C16-V3	441	441.00	0	0	441.00	1	3	441.00	1	4
B-n34-k5-C17-V3	472	472.00	0	0	472.00	1	15	472.00	1	6
B-n35-k5-C18-V3	626	626.00	0	0	626.00	1	33	626.00	1	5
B-n38-k6-C19-V3	451	451.00	3	1	451.00	1	20	451.00	1	1
B-n39-k5-C20-V3	357	357.00	2	0	357.00	1	73	357.00	1	4
B-n41-k6-C21-V3	481	481.00	79	3	481.00	1	14	481.00	1	2
B-n43-k6-C22-V3	483	483.00	82	9	483.00	3	91	483.00	17	12
B-n44-k7-C22-V4	540	540.00	17	3	540.00	1	26	540.00	1	3
B-n45-k5-C23-V3	497	497.00	5	1	497.00	1	183	497.00	1	3
B-n45-k6-C23-V4	478	478.00	717	54	478.00	23	195	478.00	17	16
B-n50-k7-C25-V4	449	449.00	23	1	449.00	1	110	449.00	1	4
B-n50-k8-C25-V5	916	916.00	7,180	3,249	916.00	11	94	916.00	5	7
B-n51-k7-C26-V4	651	651.00	4	0	651.00	1	141	651.00	1	5

B-n52-k7-C26-V4	450	450.00	0	0	450.00	1	277	450.00	1	6
B-n56-k7-C28-V4	486	486.00	18	3	486.00	1	390	486.00	1	27
B-n57-k7-C29-V4	751	751.00	21	2	751.00	1	370	751.00	1	14
B-n57-k9-C29-V5	942	942.00	115	22	942.00	1	54	942.00	1	3
B-n63-k10-C32-V5	816	816.00	75	12	816.00	19	635	816.00	13	25
B-n64-k9-C32-V5	509	509.00	3	1	509.00	1	838	509.00	1	6
B-n66-k9-C33-V5	808	808.00	34	14	808.00	1	395	808.00	1	7
B-n67-k10-C34-V5	673	673.00	229	36	673.00	41	791	673.00	15	28
B-n68-k9-C34-V5	704	704.00	27	9	704.00	1	1,395	704.00	1	9
B-n78-k10-C39-V5	803	803.00	570	248	803.00	1	727	803.00	1	18
P-n16-k8-C8-V5	239	239.00	0	0	239.00	1	0	239.00	1	0
P-n19-k2-C10-V2	147	147.00	0	0	147.00	1	1	147.00	1	0
P-n20-k2-C10-V2	154	154.00	0	0	154.00	1	1	154.00	1	0
P-n21-k2-C11-V2	160	160.00	0	0	160.00	1	4	160.00	1	0
P-n22-k2-C11-V2	162	162.00	3	0	162.00	1	4	162.00	1	0
P-n22-k8-C11-V5	314	314.00	0	0	314.00	3	0	314.00	1	0
P-n23-k8-C12-V5	312	312.00	17	1	312.00	1	0	312.00	1	0
P-n40-k5-C20-V3	294	294.00	29	2	294.00	1	31	294.00	1	3
P-n45-k5-C23-V3	337	337.00	16	2	337.00	1	104	337.00	1	5
P-n50-k10-C25-V5	410	410.00	2,715	1,163	410.00	9	17	410.00	11	4
P-n50-k7-C25-V4	353	353.00	387	27	353.00	13	75	353.00	7	13
P-n50-k8-C25-V4	392	*378.38*	9,514	7,200	392.00	17	101	392.00	33	17
P-n51-k10-C26-V6	427	427.00	213	39	427.00	1	2	427.00	1	1
P-n55-k10-C28-V5	415	415.00	4,623	1,537	415.00	13	79	415.00	13	10
P-n55-k15-C28-V8	555	*545.32*	4,537	7,200	555.00	1	1	555.00	1	0
P-n55-k7-C28-V4	361	361.00	967	125	361.00	21	174	361.00	19	40
P-n55-k8-C28-V4	361	361.00	359	39	361.00	13	135	361.00	5	13
P-n60-k10-C30-V5	443	*433.03*	7,048	7,200	443.00	139	372	443.00	105	73
P-n60-k15-C30-V8	565	*553.88*	5,122	7,200	565.00	3	4	565.00	1	1
P-n65-k10-C33-V5	487	487.00	2,951	1,806	487.00	7	140	487.00	3	10
P-n70-k10-C35-V5	485	485.00	405	176	485.00	1	71	485.00	1	5
P-n76-k4-C38-V2	383	383.00	108	26	-	1	7,200	383.00	21	437
P-n76-k5-C38-V3	405	405.00	108	16	-	1	7,200	405.00	21	314
P-n101-k4-C51-V2	455	455.00	647	169	-	1	7,200	455.00	33	3,280
M-n101-k10-C51-V5	542	542.00	4,183	2,492	-	-	-	542.00	7	85
M-n121-k7-C61-V4	719	*707.67*	5,291	21,600	-	-	-	*716.40*	265	21,600
M-n151-k12-C76-V6	659	*629.92*	2,637	21,600	-	-	-	*653.68*	391	21,600
M-n200-k16-C100-V8	791	*744.86*	606	21,601	-	-	-	*782.29*	291	21,600
G-n262-k25-C131-V12	3,249	*2,863.48*	0	21,606	-	-	-	*3,186.69*	154	21,600

Tabela A.2: Resultados computacionais para instâncias geradas utilizando $\theta = 3$.

		[4]			[34]			Metodologia Proposta		
Instância	UB	LB	BB	CPU	LB	BB	CPU	LB	BB	CPU
A-n32-k5-C11-V2	386	386.00	5	0	386.00	3	1	386.00	5	1
A-n33-k5-C11-V2	315	315.00	7	1	315.00	1	1	315.00	1	0
A-n33-k6-C11-V2	370	370.00	23	1	370.00	1	0	370.00	1	0
A-n34-k5-C12-V2	419	419.00	26	2	419.00	5	2	419.00	3	4
A-n36-k5-C12-V2	396	396.00	81	1	396.00	17	16	396.00	21	102
A-n37-k5-C13-V2	347	347.00	3	1	347.00	1	16	347.00	1	7
A-n37-k6-C13-V2	431	431.00	309	19	431.00	1	1	431.00	1	2
A-n38-k5-C13-V2	367	367.00	3	1	367.00	1	1	367.00	1	6
A-n39-k5-C13-V2	364	364.00	150	5	364.00	1	4	364.00	41	72
A-n39-k6-C13-V2	403	403.00	5	1	403.00	1	2	403.00	1	4
A-n44-k6-C15-V2	503	503.00	2,019	324	503.00	1	2	503.00	1	0

A-n45-k6-C15-V3	474	474.00	46	3	474.00	1	3	474.00	1	4
A-n45-k7-C15-V3	475	475.00	69	7	475.00	1	3	475.00	1	3
A-n46-k7-C16-V3	462	462.00	349	23	462.00	5	11	462.00	1	3
A-n48-k7-C16-V3	451	451.00	304	19	451.00	1	7	451.00	1	3
A-n53-k7-C18-V3	440	440.00	85	6	440.00	1	46	440.00	1	5
A-n54-k7-C18-V3	482	482.00	430	57	482.00	1	11	482.00	1	4
A-n55-k9-C19-V3	473	473.00	72	14	473.00	1	6	473.00	1	2
A-n60-k9-C20-V3	595	595.00	2,884	885	595.00	3	41	595.00	3	7
A-n61-k9-C21-V4	473	473.00	160	15	473.00	1	8	473.00	1	3
A-n62-k8-C21-V3	596	596.00	2,532	860	596.00	3	58	596.00	3	10
A-n63-k10-C21-V4	593	593.00	1,541	280	593.00	5	14	593.00	1	3
A-n63-k9-C21-V3	642	*625.63*	8,483	7,200	642.00	9	67	642.00	13	25
A-n64-k9-C22-V3	536	536.00	79	22	536.00	1	147	536.00	1	7
A-n65-k9-C22-V3	500	500.00	174	22	500.00	1	10	500.00	1	3
A-n69-k9-C23-V3	520	520.00	10,201	4,752	520.00	1	21	520.00	37	56
A-n80-k10-C27-V4	710	*679.43*	4,813	7,200	710.00	3	355	710.00	3	15
B-n31-k5-C11-V2	356	356.00	2	0	356.00	7	2	356.00	3	1
B-n34-k5-C12-V2	369	369.00	0	0	369.00	1	4	369.00	1	25
B-n35-k5-C12-V2	501	501.00	1	0	501.00	1	2	501.00	1	7
B-n38-k6-C13-V2	370	370.00	33	1	370.00	1	1	370.00	1	6
B-n39-k5-C13-V2	280	280.00	0	0	280.00	1	26	280.00	1	27
B-n41-k6-C14-V2	407	407.00	14	1	407.00	1	8	407.00	1	6
B-n43-k6-C15-V2	343	343.00	0	1	343.00	1	8	343.00	1	7
B-n44-k7-C15-V3	395	395.00	41	2	395.00	5	8	395.00	1	3
B-n45-k5-C15-V2	410	410.00	6	1	410.00	1	26	410.00	1	5
B-n45-k6-C15-V2	336	336.00	24	5	336.00	1	6	336.00	1	9
B-n50-k7-C17-V3	393	393.00	0	0	393.00	1	33	393.00	1	4
B-n50-k8-C17-V3	598	598.00	250	29	598.00	1	8	598.00	1	3
B-n51-k7-C17-V3	511	511.00	4	0	511.00	1	11	511.00	1	3
B-n52-k7-C18-V3	359	359.00	0	0	359.00	1	21	359.00	1	3
B-n56-k7-C19-V3	356	356.00	656	24	356.00	7	317	356.00	1	4
B-n57-k7-C19-V3	558	558.00	0	1	558.00	1	42	558.00	1	8
B-n57-k9-C19-V3	681	681.00	2,699	472	681.00	1	43	681.00	21	32
B-n63-k10-C21-V3	599	599.00	65	11	599.00	1	15	599.00	1	3
B-n64-k9-C22-V4	452	452.00	26	2	452.00	1	18	452.00	1	3
B-n66-k9-C22-V3	609	609.00	1,063	104	609.00	77	445	609.00	117	129
B-n67-k10-C23-V4	558	558.00	72	7	558.00	43	96	558.00	21	22
B-n68-k9-C23-V3	523	523.00	1,250	110	523.00	1	98	523.00	1	6
B-n78-k10-C26-V4	606	606.00	11	9	606.00	1	33	606.00	1	7
P-n16-k8-C6-V4	170	170.00	0	0	170.00	1	0	170.00	1	0
P-n19-k2-C7-V1	111	111.00	0	0	111.00	1	0	111.00	1	0
P-n20-k2-C7-V1	117	117.00	7	0	117.00	1	0	117.00	1	0
P-n21-k2-C7-V1	117	117.00	1	0	117.00	1	0	117.00	1	0
P-n22-k2-C8-V1	111	111.00	0	0	111.00	1	1	111.00	1	0
P-n22-k8-C8-V4	249	249.00	0	0	249.00	1	0	249.00	1	0
P-n23-k8-C8-V3	174	174.00	0	0	174.00	1	0	174.00	1	0
P-n40-k5-C14-V2	213	213.00	12	1	213.00	3	10	213.00	3	17
P-n45-k5-C15-V2	238	238.00	230	11	238.00	3	17	238.00	1	7
P-n50-k10-C17-V4	292	292.00	40	5	292.00	1	1	292.00	1	1
P-n50-k7-C17-V3	261	261.00	110	6	261.00	9	58	261.00	9	19
P-n50-k8-C17-V3	262	262.00	53	7	262.00	1	1	262.00	1	1
P-n51-k10-C17-V4	309	309.00	1,483	118	309.00	11	5	309.00	49	8
P-n55-k10-C19-V4	301	301.00	217	18	301.00	1	4	301.00	1	2
P-n55-k15-C19-V6	378	378.00	195	36	378.00	1	0	378.00	1	0
P-n55-k7-C19-V3	271	271.00	819	78	271.00	15	92	271.00	29	52
P-n55-k8-C19-V3	274	274.00	580	54	274.00	13	75	274.00	17	35
P-n60-k10-C20-V4	325	325.00	227	283	325.00	27	25	325.00	25	19

P-n60-k15-C20-V5	382	*379.25*	11,507	7,200	382.00	3	2	382.00	1	1
P-n65-k10-C22-V4	372	372.00	4,237	1,028	372.00	7	26	372.00	11	16
P-n70-k10-C24-V4	385	385.00	5,541	1,468	385.00	17	87	385.00	3	9
P-n76-k4-C26-V2	309	309.00	840	123	-	1	7,200	309.00	31	677
P-n76-k5-C26-V2	309	309.00	561	90	-	1	7,200	309.00	1	42
P-n101-k4-C34-V2	370	370.00	13,879	6,582	-	1	7,200	370.00	41	2853
M-n101-k10-C34-V4	458	458.00	17,916	5,238	-	-	-	458.00	13	101
M-n121-k7-C41-V3	527	527.00	4,041	3,791	-	-	-	527.00	1	389
M-n151-k12-C51-V4	483	*465.59*	3,402	21,601	-	-	-	483.00	1	143
M-n200-k16-C67-V6	605	*563.13*	1,719	21,600	-	-	-	*599.24*	465	21,600
G-n262-k25-C88-V9	2,476	*2,102.38*	181	21,602	-	-	-	*2,430.01*	262	21,600