João Fernando Machry Sarubbi
Orientador: Henrique Pacca Loureiro Luna

Problemas de Roteamento com Custos de Carga

Tese de Doutorado apresentado ao curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito para a obtenção do Título de Doutor em Ciência da Computação.

Belo Horizonte
Maio 2008

Sumário

Lista de Figuras

Lista de Tabelas

Aos meus pais...

Agradecimentos

Aos meus pais, Nelson e Édina, pelo amor e carinho infinitos, e por estarem presentes em todos os meus momentos.

Ao Prof. Henrique Pacca Loureiro Luna, por ter confiado em mim e me apoiado durante esses 7 anos, por ter me apresentado a pesquisa operacional, por ter sugerido o tema de pesquisa e por ter dado a base e o norte para este trabalho. Ao Prof. Pacca meus profundos agradecimentos.

Ao Prof. Gilberto de Miranda Jr., que teve um papel importantíssimo no desenvolvimento desse trabalho, tornando-se um grande amigo e sendo praticamente um co-orientador; também vale ressaltar suas valiosas dicas tanto no desenvolvimento deste tese como me ajudando na vida profissional.

Aos Profs. Ricardo Saraiva de Camargo e Ricardo Poley Martins Ferreira, pela amizade, força e ajuda nestes anos.

Ao Prof. Geraldo Robson Mateus pelo grande apoio durante esses anos no DCC, pelas dicas e críticas que me tornaram um pesquisador melhor e pelos conhecimentos em pesquisa operacional que adquiri em suas disciplinas.

Aos Prof. Maurício Cardoso de Souza e Abilio Pereira de Lucena Filho, pelas dicas precisas que tanto contribuiram no desenvolvimento desta tese.

Aos Amigos do DCC, Emanoel, Wagner, Habib, Pedro, Martin, Fabíola, Kissia, Gustavo, Martin, Gurvan, Fred, etc e outros pelas discussões técnicas, pelas festas e pelo bom convívio nesses anos.

À amigos que dividi apartamento, pela companhia, amizade e ajuda, Vinicius, Charles, Aioffi, Christiano, Marcus e Letícia.

À minha irmã Ana que tanto amo.

À minha querida namorada Adriana, pelo carinho, amor e por estar ao meu lado nesses 5 anos de tese. E também a sua família que me "adotou"em Belo Horizonte.

Aos meus amigos de Fortaleza.

E, é claro, à DEUS, por estar sempre ao meu lado.

Resumo

O Problema do Caixeiro Viajante (PCV) e o Problema de Roteamento de Veículos (PRV), apesar da enorme gama de aplicações e do grande número de algoritmos desenvolvidos para resolvê-los, não possuem uma estrutura de custo que permita diferenciar produtos e/ou clientes mais importantes. Eles também não se preocupam com o tempo de espera dos consumidores que recebem as mercadorias e não asseguram um dado nível de qualidade de serviço para esses consumidores. Tanto o PCV quanto o PRV se preocupam apenas com o custo egoísta do operador logístico, que deseja retornar ao ponto de partida o mais cedo possível. Nesta tese são apresentados e resolvidos, de forma exata, quatro problemas, em que o custo da carga é relevante para o cálculo do custo total. Esse custo da carga está relacionado com a quantidade e com a qualidade dos produtos transportados, podendo também priorizar clientes mais importantes e/ou produtos perecíveis.

Os problemas estudados neste trabalho são: **Problema de Mínima Latência** (PML); **Problema de Um Veículo de Entrega** (PUVE); **Problema do Caixeiro Viajante Multiproduto** (PCVM); e, **Problema do Caixeiro Viajante Multiproduto Congestionado** (PCVMC). Esses problemas têm como objetivo principal aliar os interesses, muitas vezes conflitantes, dos operadores logísticos que tendem a minimizar o seu próprio custo operacional, mas também precisam maximizar a satisfação dos seus clientes. Para todos os quatro problemas são apresentadas novas formulações matemáticas. Para o PML e para o PUVE, problemas existentes na literatura, são apresentados resultados exatos e resultados heurísticos melhores que os presentes na literatura. Para o PCVM são apresentados três versões de um algoritmo *Cut-and-Branch* baseado no método de Decomposição de Benders. Esse método se mostrou mais eficiente, em termos de tempo computacional, que o resolvedor CPLEX. Além disso, foi desenvolvido um algoritmo Lagrangeano que conseguiu resolver instâncias para as quais o CPLEX não consegue. Para o PCVMC, o problema mais geral, que engloba todos os anteriores e que usa uma função de custo não-linear, também foi desenvolvido um algoritmo baseado no Método de Particionamento de Benders que conseguiu resultados exatos de maneira eficiente.

Palavras Chaves: Problema de Mínima Latência, Problema de Um Veículo de Entrega, Problema do Caixeiro Viajante Multiproduto, Problema do Caixeiro Viajante Multiproduto Congestionado, GRASP, *Cut-and-Branch*, Relaxação Lagrangeana, Decomposição de Benders Generalizada.

Abstract

There are a lot of applications and a lot of algorithms for the Traveling Salesman Problem (TSP) and for the Vehicle Routing Problem (VRP). Despite of that, both problems do not have a cost structure that enables differ more important products and clients. Besides, the TSP and the VRP do not care about the waiting times of all the costumers that want to receive their products as soon as possible. Both, the TSP and the VRP, just care about the single selfish driver that wants to return as soon as possible to the depot.

The objective of this work is to present, to model and to solve problems where a load cost is incurred, besides the time or distance. This load cost can represent the weight, the price of the load or even the priority of some clients. In this work four problems are presented where the load cost is important. The problems are **The Minimum Latency Problem** (MLP); **The Single Vehicle Delivery Problem** (SVDP); **The Multicommodity Traveling Salesman Problem** (MTSP) and **The Congested Multicommodity Traveling Salesman Problem** (CMTSP). All these problems deal with the logistic operator conflict between minimize its own operational costs and maximize the client satisfaction. For all these problems it is presented a new mathematical formulation. For the MLP and the SVDP, the exact and heuristics results are better than found in literature. For the MTSP, it is developed a Cut-and-Branch algorithm that is able to find optimal solutions faster than stand alone CPLEX codes. For instances that CPLEX cannot deal with (very large ones), a Lagrangean Relaxation was deployed to compute lower bounds. A version of the well known GRASP procedure is also provided to ensure the calculation of optimality gaps not attainable by CPLEX. For the CMTSP, the more general problem of this work, that uses a non-linear objective function, it is also developed a Generalized Benders Decomposition algorithm.

Keywords: The Minimum Latency Problem, The Single Vehicle Delivery Problem, The Multicommodity Traveling Salesman Problem, The Congested Multicommodity Traveling Salesman Problem, GRASP, Cut-and-Branch, Lagrangean Relaxation, Generalized Benders Decomposition.

Capítulo 1

Introdução

Um dos problemas da literatura mais abrangente em termos de roteamento é o Problema de Roteamento de Veículos (PRV), do inglês *Vehicle Routing Problem* (VRP) [LN87] [TV02], que também é chamado de *Vehicle Scheduling* [CW64] [Gas67], *Vehicle Dispaching* [CE69] [DR59] ou simplesmente *Delivery problem* [Hay67]. O objetivo do PRV é determinar ótimas rotas usando uma frota de veículos, baseados em um ou mais depósitos, para servir a um conjunto de consumidores. Muitas características adicionais, como restrições operacionais, podem ser impostas nas construções das rotas. Por exemplo: o serviço pode incluir entrega e/ou coleta; a quantidade de produtos carregada em cada rota não pode exceder à capacidade do veículo; o tempo ou a distância total de cada rota não pode ser maior que um limite estabelecido; o serviço deve ocorrer dentro de uma janela de tempo; a frota pode conter veículos heterogêneos; relações de precedência podem existir entre os consumidores; a demanda do consumidor pode não ser completamente conhecida *a priori*; o fluxo para um consumidor pode ser dividido entre veículos diferentes; e algumas características do problema, como demandas ou tempos de viagem, podem variar dinamicamente. O PRV é um dos problemas de otimização combinatória mais estudados devido à sua relevância prática e sua considerável dificuldade.

O importante Problema do Caixeiro Viajante (PCV) [DFJ54] [NW88] [GL00] [ABCC06] é uma referência básica para o Problema de Roteamento de Veículos. O PCV é uma especialização do Problema de Roteamento de Veículos, em que um único veículo, sem restrições de capacidade, precisa percorrer um circuito Hamiltoniano de menor custo. Ele é importante na solução de diversos problemas de transporte e logística, entre outros.

Apesar da enorme gama de restrições e da complexidade do problema, o PRV, e também o PCV, não consideram a carga que o veículo está transportando. No PRV, o peso da carga

é somente um fator de restrição tendo em vista a capacidade limite do veículo, não existe uma relação proporcional entre a quantidade e o valor da carga transportada com o custo de se transportar essa carga por algum trecho.

No PRV e no PCV todos os produtos são idênticos. Entretanto, em situações reais podem haver produtos diferentes a serem transportados em um mesmo veículo. Nesse caso, torna-se importante priorizar, em termos de ordem de entrega, produtos perecíveis e produtos mais frágeis. Aqueles que são perecíveis, por exemplo, podem necessitar de veículos com refrigeração. Nesses casos, à medida que forem entregues, os gastos de refrigeração podem tornar-se cada vez menores. Além disso, quando produtos diferentes são transportados no mesmo veículo simultaneamente, é importante estudar o efeito de se transportar uma determinada carga por alguma região. Por exemplo: é comum que empresas transportadoras que entreguem seus produtos para diferentes localidades deixem para entregar por último aqueles destinados a bairros ou localidades mais perigosas e mais sujeitas a assaltos. Além disso, produtos diferentes têm pesos diferentes. Em certos casos, também é possível contabilizar o custo do veículo para trafegar mais, ou menos, pesado. Esse peso pode acelerar a depreciação do veículo, aumentando gastos com manutenção, e, no caso de transporte terrestre, gerar custos com possíveis multas por excesso de peso.

Além dos diferentes tipos de produtos, empresas transportadoras podem lidar com diferentes tipos de clientes. Alguns desses podem requerer certa prioridade na entrega devido a um contrato estabelecido entre eles e a empresa transportadora. Quando cada cliente recebe somente um tipo de produto em cada entrega, essa prioridade entre clientes pode ser modelada como prioridade de produtos e vice-versa. Isso já não ocorre quando um cliente necessita receber diversos tipos de produtos em uma mesma entrega. No PCV, não existe essa demanda heterogênea para cada cliente, enquanto, no PRV, ela pode existir, embora não ocorra necessariamente uma relação proporcional entre o peso e o custo.

No transporte de produtos, cada cliente pode necessitar de uma quantidade heterogênea de cada um desses. Essa quantidade diferente é um fator que pondera os custos citados. Por exemplo: quando comparados clientes que teoricamente teriam a mesma prioridade devido a um contrato específico, é possível que um tenha uma prioridade maior de entrega que o outro. Para isso, basta que o valor, em dinheiro, requerido por um seja maior que o valor, também em dinheiro, requerido pelo outro. Esse valor é a multiplicação do número de produtos requeridos pelo valor de cada produto. Usando esse raciocínio, um cliente que deseja receber muitos produtos de valor unitário baixo pode se tornar prioritário em relação a um que deseja receber uma quantidade pequena de um produto mais valioso.

Neste trabalho são abordados quatro problemas em relação aos quais o custo da carga é relevante para a solução final.

O primeiro problema estudado é chamado de Problema de Mínima Latência (PML). Ele é um problema variante do PCV, no qual o objetivo é encontrar um circuito Hamiltoniano que partindo de uma origem 1 minimize a soma dos tempos de espera (latência) dos consumidores [CMM67] [Luc90] [BCC$^+$94]. A latência de um vértice pode ser definida como a distância entre esse vértice e o vértice origem. Esse tempo de espera é medido sob o ponto de vista dos consumidores [Tsi92]. O PML pode ser interpretado como um problema de agendamento de tarefas em que a soma dos tempos de cada tarefa precisa ser minimizada [Eij95]. Isto é, no PML o objetivo não é terminar todas as tarefas no menor tempo possível, mas, sim, minimizar a soma das esperas (latências) de cada um dos clientes, em função da qual, cada um deles, solicita uma tarefa distinta. Quanto mais rápido um cliente receber a sua tarefa, maior será sua satisfação. É preciso esclarecer que o tempo para executar cada tarefa depende da tarefa executada anteriormente, da mesma forma que, no PCV, a distância para alcançar um vértice depende do vértice previamente visitado. O PML também pode ser interpretado como um problema de minimizar o custo de um veículo que, saindo carregado, tenha que entregar uma unidade de um produto em cada localidade, e, que se contabilize um custo que é proporcional à quantidade de produtos transportados em cada trecho. Por isso, a solução do PML tende a percorrer, quando o veículo estiver mais carregado de produtos, trechos mais baratos, e, à medida que esses produtos forem entregues, o custo por trecho torna-se cada vez menos importante.

O segundo problema é uma extensão do Problema de Mínima Latência e é conhecido como Problema de Um Veículo de Entrega (PUVE) [BM$^+$89]. Ele consiste em percorrer um circuito Hamiltoniano que minimize o tempo de espera dos consumidores, da mesma forma que no PML. Entretanto, no PUVE, esse tempo de espera de cada cliente é multiplicado pelo número de produtos que são entregues a esse cliente. De uma forma geral, pode-se interpretar o PUVE como um Problema de Mínima Latência com demandas heterogêneas nos vértices. Outra interpretação cabível é que tanto o PML quanto o PUVE são orientados aos clientes, entretanto, no PML, todos os clientes têm a mesma prioridade. Já, no PUVE, a prioridade de cada cliente é proporcional à quantidade de produtos que esse cliente deseja receber. Não necessariamente a menor distância (ou o menor tempo total de viagem) é a melhor solução. Podem existir casos em que seja melhor percorrer uma distância maior para logo nas primeiras visitas entregar mais produtos, deixando o veículo mais vazio e economizando no custo de transportá-los às localidades subseqüentes. No PUVE, o custo da

carga é ainda mais evidente do que no PML. Nele, não se está somente minimizando o tempo de espera dos clientes que recebem os produtos, mas, sim, a soma dos tempos de espera de todos os produtos que são entregues. Se esses produtos forem pessoas, que necessitam ser transportadas, pode-se dizer então que o PUVE maximiza a soma da satisfação dessas pessoas tentando entregar o maior número delas no seu local de destino o mais cedo possível.

O terceiro problema é uma extensão do PCV e do PUVE e é o principal foco deste trabalho. Pelo que se sabe esse problema foi proposto na literatura e modelado pela primeira vez pelo autor deste trabalho e foi denominado de Problema do Caixeiro Viajante Multiproduto (PCVM). Ele consiste em achar um caminho de menor custo que, saindo de uma origem, percorra todas as localidades, passando exatamente uma vez em cada localidade e depois volte para a origem. No PCVM cada cliente pode requerer uma quantidade diferente de produtos distintos. O objetivo do PCVM é percorrer um caminho Hamiltoniano minimizando a soma de dois custos em cada arco: o fixo e o variável. O custo fixo é o mesmo existente no PCV e não está associado à carga transportada. Independente se um veículo está vazio ou cheio, toda empresa de transporte possui um custo fixo para utilizar um veículo. Esse custo está associado a alguns fatores como: custo do operador logístico; preço do combustível que esse veículo vazio gasta ao percorrer um trecho; aluguel do veículo; pedágio. O custo variável está associado à carga transportada em cada trecho. No PUVE, cada cliente recebe uma quantidade heterogênea de um dado produto, entretanto, nesse problema, todos os clientes recebem o mesmo tipo de produto. Já no PCVM, cada cliente pode receber um tipo de produto diferente, isto é, na rede trafegam produtos com diferentes valores agregados. Por simplificação, no PCVM, cada cliente está associado a somente um tipo de produto, não impedindo que clientes diferentes sejam associados a produtos diferentes. Dessa forma é possível priorizar clientes mais importantes. Para isso, basta associar um custo variável maior aos produtos desse cliente. Com o PCVM também é possível associar a cada trecho um custo variável diferente, independente do produto trafegado. Por exemplo: é mais difícil para um veículo mais pesado percorrer uma estrada sem conservação. Nesse caso, com o PCVM, é possível associar um custo de carga maior a essa via de transporte. O PCVM possui uma função mais flexível para aplicações reais. Nele, é possível tratar o conflito que as empresas de transporte deparam ao tentar minimizar seu custo operacional (custo fixo) e maximizar a satisfação de seus clientes garantindo-lhes melhor qualidade de serviço.

O quarto problema, ainda mais abrangente que o PCVM, leva em conta fenômenos de congestionamento. Ele é chamado de Problema do Caixeiro Viajante Multiproduto

Congestionado (PCVMC). Nele, da mesma forma que no PCVM, existem os custos fixos e variáveis associados a cada arco, entretando, além deles, ainda existe um custo de congestionamento associado ao total do produtos trafegados em cada trecho e à capacidade de transporte nesse trecho. Esse custo de congestionamento é modelado por uma função não-linear. Neste trabalho utiliza-se a função de Kleinrock. Isso se deve à sua simplicidade e adequação aos efeitos do congestionamento. No PCVMC, cada arco tem uma capacidade de transporte e, à medida que o fluxo global que passa pelo arco se aproxima da capacidade desse arco, o custo de congestionamento aumenta. O objetivo do PCVMC também é estabelecer o menor circuito Hamiltoniano levando em conta o custo fixo, o custo variável (ambos são contabilizados da mesma forma que no PCVM) e, também, o custo de congestionamento.

O objetivo principal deste trabalho é propor formulações e abordagens eficientes para a solução de cada um desses quatro problemas. Todas as formulações deste trabalho são baseadas em fluxos em redes e, apesar desta tese explorar, em sua maior parte, abordagens exatas, também são apresentadas algumas heurísticas eficientes para alguns desses problemas. As heurísticas, além de obter um limite superior de boa qualidade, também têm o objetivo de auxiliar na solução exata. Algumas das principais contribuições desta tese são:

- Dois modelos de programação linear uniproduto para o Problema de Mínima Latência. O primeiro deles é baseado no trabalho de Garvin *et al.* [GCJS57] para o Problema de Roteamento de Veículos. O segundo acrescenta ao primeiro variáveis e restrições baseadas no Problema Quadrático de Atribuição (PQA). Com esse último conseguiram-se resultados até 258 vezes mais rápidos e com *gaps* de relaxação linear 45% menores em relação ao primeiro.

- Dois modelos de programação linear multiproduto que utilizam conservação de fluxo e variáveis e restrições baseadas no Problema Quadrático de Atribuição. O segundo modelo que acrescenta ao primeiro variáveis e restrições baseadas no PQA apresenta um *gap* de relaxação linear extremamente competitivo para instâncias de até 25 vértices. Para muitas instâncias esse *gap* é inferior aos *gaps* do modelo de Bianco *et al.* [BMR93], e também é inferior aos *gaps* dos modelos baseados no PQA que são propostos neste trabalho. Comparando o tempo de computação entre o primeiro e o segundo modelo multiproduto, este último conseguiu resultados muito expressivos. Por exemplo, para instâncias de 20 vértices, o último conseguiu resultados 40 vezes mais rápidos que o primeiro.

- Um modelo de programação linear para o Problema de Mínima Latência baseado no Problema Quadrático de Atribuição. Esse modelo conseguiu limites de relaxação linear inferiores, em média, a 1.5%, mesmo para instâncias de 60 vértices.

- Um modelo de programação linear que conseguiu soluções exatas para instâncias do Problema de Mínima Latência de até 80 vértices. O modelo proposto apresentou *gaps* de relaxação linear, em média, menores que 2% mesmo para instâncias de 80 vértices. Além disso, esses limites foram calculados em menos de 82 segundos.

- Uma heurística baseada na meta-heurística GRASP que utiliza *path-relinking* que conseguiu resultados em geral, menores que 1.5% do ótimo, tanto para instâncias do Problema de Mínima Latência, como para instâncias do Problema de Um Veículo de Entrega. As soluções da heurística também foram utilizadas como primeiro limite superior para o resolvedor CPLEX, auxiliando-o na solução exata dessas instâncias, e tornando o CPLEX, em alguns casos, 80% mais rápido.

- Solução exata de instâncias de 150 vértices do Problema de Mínima Latência. Essas foram baseadas nas instâncias da Biblioteca *QAPLib* e a otimalidade da solução foi provada utilizando tanto a heurística GRASP quanto o modelo de programação linear baseado no Problema do Caminho Mínimo.

- Um conjunto de restrições adicionais que fortaleceram o limite de programação linear do modelo proposto por Bianco *et al.* [BM$^+$89] para o Problema de Um Veículo de Entrega. Esse conjunto torna esse novo modelo mais rápido que o original, quando abordado pelo resolvedor CPLEX.

- Um modelo de programação linear inteira mista para a solução de instâncias do Problema do Caixeiro Viajante Multiproduto. Esse modelo conseguiu provar a otimalidade de instâncias com até 50 vértices. Ele também apresenta uma relaxação de programação linear, na média, não maior que 1.07% do ótimo, mesmo para instâncias de 50 vértices.

- Apresentação de três versões de um algoritmo *Cut-and-Branch* para o PCVM. Esse algoritmo é baseado no Método de Particionamento de Benders, e conseguiu resultados expressivamente mais rápidos que o CPLEX monolítico. A primeira versão é o método *Cut-and-Branch* padrão. A segunda se aproveita da estrutura especial da formulação apresentada. A terceira é uma versão multicorte.

- Implementação de um algoritmo Lagrangeano que conseguiu soluções, com garantia de limites, para instâncias do PCVM e que o CPLEX não obteve. Esse algoritmo utiliza como heurística Lagrangeana uma versão simplificada do algoritmo GRASP presente tanto no capítulo sobre o Problema de Mínima Latência, como no capítulo sobre o Problema de Um Veículo de Entrega.

- Um modelo de programação não-linear inteira mista para o Problema do Caixeiro Viajante Multiproduto Congestionado.

- Um algoritmo baseado no Método de Benders Generalizado que conseguiu soluções exatas para instâncias do Problema do Caixeiro Viajante Multiproduto Congestionado de até 20 vértices.

Este trabalho está organizado da seguinte forma:

No capítulo 2 é estudado o Problema de Mínima Latência. Para esse problema uma revisão bibliográfica é apresentada e são propostas três tipos de formulações. A primeira baseada em conservação de fluxo, a segunda, baseada no Problema Quadrático de Atribuição e a terceira, no Problema de Caminho Mínimo. Além disso, é proposta uma heurística para o problema. Resultados numéricos exatos e heurísticos também são apresentados.

No capítulo 3 é estudado o Problema de Um Veículo de Entrega, uma generalização do Problema de Mínima Latência. Para esse problema uma formulação matemática baseada na formulação de Bianco *et al.* [BM$^+$89] é desenvolvida juntamente com uma heurística que consegue resultados próximos ao ótimo inteiro. A heurística também auxilia na solução exata pelo resolvedor CPLEX. Resultados numéricos exatos e heurísticos também são mostrados.

No capítulo 4 é apresentado o Problema do Caixeiro Viajante Multiproduto (PCVM). O problema é definido formalmente e uma nova formulação linear inteira mista é descrita. Um algoritmo *Cut-and-Branch* e um algoritmo Lagrangeano são desenvolvidos. Esses algoritmos são responsáveis tanto por tornar as soluções das instâncias do PCVM mais rápidas como também para resolver instâncias que o resolvedor CPLEX não consegue trabalhar. Resultados numéricos exatos e heurísticos também são apresentados, juntamente com uma análise deles.

No capítulo 5 é apresentado o Problema do Caixeiro Viajante Multiproduto Congestionado (PCVMC). Uma formulação não-linear inteira mista é apresentada juntamente com um algoritmo baseado no Método de Benders Generalizado. Esse método torna possível

a solução exata de instâncias do PCVMC. Resultados numéricos exatos são apresentados, juntamente com uma análise desses resultados.

No capítulo 6 é feita a conclusão deste trabalho, juntamente com as indicações de trabalhos futuros.

Este trabalho ainda apresenta dois apêndices. O primeiro, Apêndice A, mostra os resultados para todas as instâncias do PML testadas utilizando cada uma das formulações apresentadas. O segundo, Apêndice B, também mostra os resultados para todas as instâncias, mas neste caso são apresentados os resultados relativos as instâncias do PUVE.

Capítulo 2

Problema de Mínima Latência

2.1 Introdução

O Problema de Mínima Latência (PML), do inglês *Minimum Latency Problem*, também conhecido como *Traveling Repairman Problem* e *Deliveryman Problem* é uma variante do Problema do Caixeiro Viajante (PCV) [DFJ54] [Wag69] [Chr79] [NW88]. Nesse problema, o objetivo é encontrar um circuito Hamiltoniano, iniciando-se de uma única origem, que minimize a soma dos tempos de espera dos consumidores. Neste capítulo, é apresentado um exemplo didático do PML e uma comparação com o PCV. Além disso, são implementadas oito novas formulações de programação linear inteira mista para o PML: duas uniproduto e seis multiproduto. Dentre elas quatro são formulações de fluxo baseadas no Problema de Roteamento de Veículos [LN87], três são baseadas no Problema Quadrático de Atribuição (PQA) [KB57] [AJ94] [Mir04] e uma é baseada no Problema do Caminho Mínimo. Além dos modelos, também é apresentada uma meta-heurística GRASP, que utiliza quatro buscas locais alinhadas, juntamente com uma implementação do método de *Path Relinking*. A solução encontrada pela meta-heurística GRASP também é utilizada como primeiro limite superior para o resolvedor CPLEX. O intuito, nesse caso, é diminuir o tempo de computação da formulação utilizada devido ao aumento de eficiência do algoritmo *Branch-and-Bound* do resolvedor CPLEX. Para o PML são apresentados resultados exatos para instâncias aleatórias de até 80 vértices e para instâncias baseadas na biblioteca *QAPlib* de até 150 vértices. Além disso, a maioria dos modelos propostos apresenta *gaps* de relaxação linear, em média, menores que 1.5%.

2.2 Problema de Mínima Latência

O Problema de Mínima Latência é uma variante do PCV no qual a origem é dada e a meta é minimizar a soma dos tempos de chegada em todos os vértices, isto é, a média das latências. A latência de um vértice pode ser definida como o tempo decorrido para visitar o vértice a partir da origem. Esse tempo de espera é dado sob o ponto de vista dos consumidores [Tsi92]. De uma forma geral é possível dizer que o PCV é orientado ao custo de operacionalização do veículo. Já o PML é orientado à satisfação dos clientes que são atendidos pelo veículo. O problema em questão foi introduzido em 1967 por Conway, Maxwell e Miller [CMM67]. O PML pode ser interpretado como um problema de seqüenciamento com uma única máquina, com tempo de processamento dependendo da seqüência e cujo objetivo é minimizar o tempo total das tarefas [Eij95]. De acordo com Goemans e Kleinberg [GK98], apesar da óbvia similaridade com o PCV clássico, o PML é bem menos comportado de um ponto de vista computacional.

Sejam as seguintes variáveis e parâmetros:

$$x_{ij}^k = \begin{cases} 1 & \text{se o arco } (i,j) \text{ é visitado na posição } k \\ 0 & \text{caso contrário} \end{cases}$$

c_{ij} = tempo gasto para percorrer o arco (i, j)

O PML pode ser modelado da seguinte forma:

$$\min \quad \sum_{k=1}^{n}(n-k+1)\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}^k \tag{2.1}$$

sujeito a

$$\sum_{j=1}^{n} x_{ij}^{k} - \sum_{l=1}^{n} x_{ij}^{k-1} = \quad 0 \qquad i = 2, ..., n \;\; k = 2, ..., n \tag{2.2}$$

$$\sum_{j=1}^{n} x_{1j}^{1} = \quad 1 \tag{2.3}$$

$$\sum_{i=1}^{n} x_{i1}^{n} = \quad 1 \tag{2.4}$$

$$\sum_{k=1}^{n} \sum_{i=1}^{n} x_{ij}^{k} = \quad 1 \qquad j = 1, ..., n \tag{2.5}$$

$$\sum_{k=1}^{n} \sum_{j=1}^{n} x_{ij}^{k} = \quad 1 \qquad i = 1, ..., n \tag{2.6}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij}^{k} = \quad 1 \qquad k = 1, ..., n \tag{2.7}$$

$$x_{ij}^{k} \in \quad \{0, 1\} \qquad i, j, k = 1, ..., n \tag{2.8}$$

Essa formulação foi apresentada em 1993 no trabalho de Bianco *et al.* [BMR93].

As restrições (2.2) garantem a conservação do fluxo em cada vértice. Restrições (2.3) e (2.4) garantem que todo circuito deve começar na origem na posição 1 e terminar nesse mesmo vértice na posição n. As equações (2.5) e (2.6) impõem que cada vértice do caminho seja visitado uma, e somente um vez durante o circuito. As restrições (2.7) e uma das restrições (2.5) e (2.6) são redundantes.

Algoritmos de otimização para o PML foram desenvolvidos por Lucena [Luc90], Simchi-Levi e Berman [SLB91], Fischetti, Laporte e Martelo [FLM93] e Wu, Hang e Zhan [WHZ04]. O primeiro propõe um algoritmo enumerativo, baseado em uma formulação não-linear inteira, onde os limites inferiores são obtidos por relaxação Lagrangeana. Lucena relata resultados ótimos para problemas de até 30 vértices. Simchi-Levi e Berman descrevem um método *Branch-and-Bound* baseado na relaxação da árvore geradora mínima. Fischetti *et al.* [FLM93] propõem um algoritmo *Branch-and-Bound* baseado em uma formulação de programação inteira. O trabalho utiliza matróides para gerar o limite inferior. Wu, Hang e Zhan [WHZ04] apresentam resultados ótimos para instâncias de até 23 vértices usando programação dinâmica com *pruning*. Em 2007, Méndez-Días, Zabala e Lucena [MDZL07] apresentaram a mais nova formulação para o Problema de Mínima Latência. Mendés-Dias *et al.* [MDZL07] se concentraram em uma solução poliedral para o PML, conseguindo

gaps de relaxação linear melhores aos *gaps* dos modelos de Fischetti *et al.* [FLM93] e Eijl [Eij95]. Atualmente, somente problemas com até 60 vértices são resolvidos na otimalidade [FLM93] [BCC+94] [Eij95] [WHZ04] [SL07a].

Tanto o PML como o PCV são casos especiais de um problema mais geral chamado de Problema do Caixeiro Viajante com Dependência de Tempo (PCVDT) [Fox73] [PQ78] [FGG79] [FGG80] [Luc90] [WS95]. No PCVDT, o custo de sair de um determinado vértice e chegar em outro não é dado somente pela distância entre esses vértices, como acontece no PCV. Nesse problema, esse custo também é dado pela ordem de visitação desses vértices no circuito Hamiltoniano. O PML é um caso especial do PCVDT em que esse custo é uma função da distância entre os vértices e também da ordem de visitação dos vértices, assim como no PCVDT. Entretanto, no PML, o custo para atravessar um arco é diretamente proporcional ao número de vértices que faltam para completar o circuito Hamiltoniano [BCC+94].

De acordo com Blum *et al.* [BCC+94], o Problema de Mínima Latência é NP-Difícil. Algoritmos polinomiais são conhecidos para grafos muito específicos, vide [ACP+86] [GJT02]. Para mais detalhes, vide Wu *et al.* [WHZ04]. Por ser NP-Difícil, muitos autores têm desenvolvido algoritmos aproximados para o PML, dentre eles é possível citar [BCC+94] [GK98] [AK03] [AW03]. O primeiro fator de aproximação para o PML foi 144 e foi apresentado por Blum *et al.* [BCC+94]. Chaudhuri e al [CGRT03] melhoraram esse fator para 3.59.

2.3 Diferença entre o PCV e o PML

Para entender a diferença entre o Problema do Caixeiro Viajante e o Problema de Mínima Latência é apresentado um exemplo didático. Para um grafo $G = (V, A)$ com seis vértices e cada arco com um custo c_{ij}, são resolvidos o PCV e o PML. A Figura (2.1) mostra o grafo G, lembrando que, para simplificar, o exemplo se refere a um Problema do Caixeiro Viajante simétrico, de forma que são iguais os custos para atravessar um arco em um e no outro sentido. Nesse grafo os valores em *itálico* representam a demanda em cada vértice. Entretanto, no PCV e no PML essas demandas não tem importância no cálculo do custo e da rota ótima.

A Figura (2.2) mostra a solução do PCV para o grafo G, que é 81 (9+10+17+21+12+12). A Figura (2.3) mostra a solução ótima para o PML utilizando como base o mesmo grafo G. A solução ótima inteira para o PML, no grafo G, é 259 (9 + (9+10) + (9+10+12) + (9+10+12+19)+ (9+10+12+19+17) + (9+10+12+19+17+16)). Outra forma de se cal-

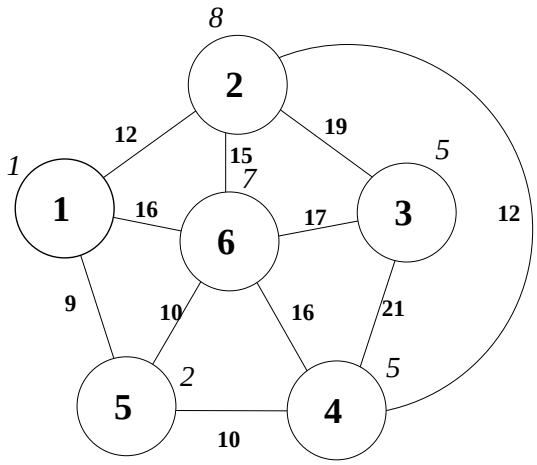


Figura 2.1: Grafo Base.

cular o valor da função objetivo é colocar os valores dos arcos em evidência, tornando a fórmula mais compacta (9*6 +10*5 + 12*4 + 19*3+17*2+16). Analisando de forma que o custo de cada arco seja multiplicado pelo número de clientes que ainda faltam para serem servidos, é possível interpretar o PML como um problema de roteamento. Nessa interpretação, o veículo sai carregado de um produto homogêneo para cada cliente e o custo de cada arco é multiplicado pelo número de produtos que estão sendo transportados nesse arco. A cada vértice visitado é deixada uma unidade de produto, tornando o veículo mais vazio e o custo mais barato para as localidades subseqüentes.

É preciso esclarecer que não é cabível comparar os valores da solução ótima para esses dois exemplos. Isso acontece porque cada dos problemas possui uma contabilização de custos diferente, impactando no valor da função objetivo. O que é preciso verificar é que a rota ótima para cada problema é diferente. Enquanto a solução ótima para o PCV (81) é o menor tempo para percorrer o circuito, a solução ótima para o PML (259) é a menor soma de todos os tempos de espera de todos os clientes. Observa-se que, em vez de 81, a solução do PCV daria 83, se usado o circuito ótimo do PML. Por outro lado, em vez do valor ótimo de 259 para o PML, a latência daria 271, se usado o circuito ótimo para o PCV.

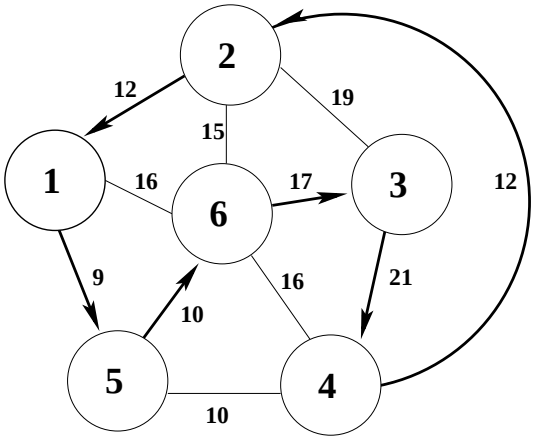


Figura 2.2: Solução PCV.

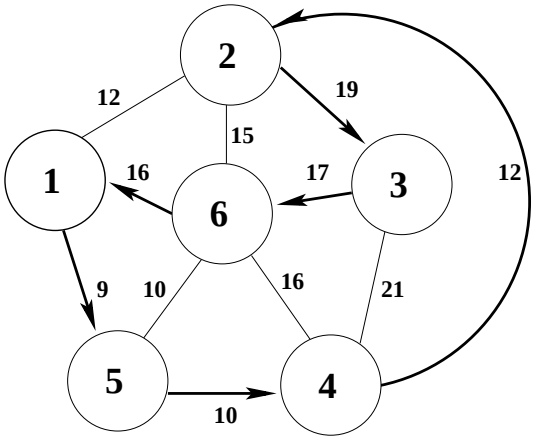


Figura 2.3: Solução PML.

2.4 Formulações de Fluxos para o Problema de Mínima Latência

Nesta seção são apresentadas quatro novas formulações de fluxo para o Problema de Mínima Latência. As duas primeiras são formulações uniproduto; as duas últimas, formulações multiproduto. A segunda formulação uniproduto e a segunda multiproduto usam variáveis e restrições adicionais com o intuito de melhorar os limites de programação linear e os tempos de computação. Essas variáveis adicionais usam a relação existente entre a ordem de visitação dos arcos e o fluxo que passa por esses arcos.

2.4.1 Formulação de Fluxos Uniproduto para o Problema de Mínima Latência

Considere o grafo $G = (V, A)$, onde V é um conjunto de vértices e A é um conjunto de arcos. Suponha que exista uma origem $1 \in V$ e, para cada vértice $k \in V$, que inclui também o vértice 1, uma demanda única deve ser entregue por um circuito que minimize a soma dos tempos de espera de cada vértice. Para esse problema um modelo de programação linear inteira mista, chamado de F1_PML, é desenvolvido. É definido:

$$x_{ij} = \begin{cases} 1 & \text{se o veículo atravessa o arco}(i,j) \\ 0 & \text{caso contrário} \end{cases}$$

c_{ij} = tempo gasto para percorrer o arco (i, j)

f_{ij} = fluxo total de produtos que passam pelo arco (i, j)

$$\text{minimize} \quad \sum_{(i,j)\in A} c_{ij} f_{ij} \tag{2.9}$$

sujeito a

$$\sum_{i\in V|i\neq j} x_{ij} = \quad 1 \qquad \forall j \in V \tag{2.10}$$

$$\sum_{j\in V|i\neq j} x_{ij} = \quad 1 \qquad \forall i \in V \tag{2.11}$$

$$\sum_{j\in V|j\neq 1} f_{1j} = \quad n \tag{2.12}$$

$$\sum_{i\in V} f_{ik} - \sum_{j\in V} f_{kj} = \quad 1 \qquad \forall k \in V - \{1\} \tag{2.13}$$

$$\sum_{i,j\in V} f_{ij} = \quad \tfrac{n(n+1)}{2} \tag{2.14}$$

$$f_{ij} \leq \quad \mid V \mid x_{ij} \qquad \forall i,j \in V \tag{2.15}$$

$$f_{ij} \geq \quad 0 \qquad \forall i,j \in V \tag{2.16}$$

$$x_{ij} \in \quad \{0,1\} \qquad \forall i,j \in V \tag{2.17}$$

onde $n =\mid V \mid$.

A função objetivo (2.9) minimiza o tempo total de espera para entrega dos produtos. As restrições (2.10) e (2.11) garantem que só chega um e só sai um arco de cada vértice. O conjunto de restrições (2.12) assegura que o fluxo que sai do vértice origem é igual ao número de vértices. As restrições (2.13) forçam que em cada vértice seja entregue sua demanda unitária. A restrição (2.14) é uma restrição redundante que restringe o somatório de todos os fluxos na rede em $\frac{n(n+1)}{2}$, onde $n =\mid V \mid$. Essa restrição é responsável por uma melhora nos *gaps* de relaxação linear da formulação. As restrições (2.15) e (2.16) garantem que não pode haver fluxo nos vértices que não foram escolhidos na solução ótima e que esse fluxo é não-negativo. E, finalmente, as restrições (2.17) asseguram a integralidade da solução.

É conveniente salientar que são as variáveis f_{ij} que asseguram a eliminação de subciclos inválidos. Um artifício similar a esse foi utilizado por Miller, Tucker e Zemlin [MTZ60] em 1960 no Problema do Caixeiro Viajante. Nesse artigo [MTZ60], as variáveis que eliminam ciclos contabilizam o tempo. Segundo Laporte [LN87], em 1979, Gavish e Graves [GS79]

também utilizam variáveis que contabilizam o fluxo global na eliminação de ciclos em um modelo para o Problema de Roteamento de Veículos Capacitado.

2.4.1.1 Uma formulação Uniproduto estendida para o Problema de Mínima Latência

A formulação (2.9)-(2.17) é suficiente para descrever o PML. Por outro lado, dada a associação entre o Problema Quadrático de Atribuição (PQA) [KB57] e o PCV, vide seção (2.5.2), é possível ter uma idéia de como tornar a formulação anterior mais forte com a ajuda de variáveis inspiradas nas variáveis binárias do PQA quando esse é usado para resolver instâncias do PCV.

Quando se usa o PQA para resolver o PCV, é importante lembrar que as restrições de atribuição do PQA não são escritas no espaço dos arcos, isto é, não significam que um dado arco é percorrido somente uma vez em alguma direção. Elas asseguram que um dado vértice é visitado em apenas uma ordem de visitação e que uma dada ordem de visitação é usada por apenas um vértice somente uma vez. Para usar esse tipo de restrição de acoplamento com o modelo (2.9)-(2.17), propõe-se um novo conjunto de variáveis p_{ij}, definindo a ordem j de visitação do vértice i. Normalmente, para esse novo conjunto de variáveis, deveria ser criado um novo conjunto de ordens de visitação. Isso não será necessário, pois existem tantas ordens de visitação quanto os vértices de V. Essa observação é a chave para escrever as restrições de atribuição como:

$$\sum_{i \in V} p_{ij} = 1 \quad \forall j \in V \tag{2.18}$$

$$\sum_{j \in V} p_{ij} = 1 \quad \forall i \in V \tag{2.19}$$

$$p_{11} = 1 \tag{2.20}$$

$$\sum_{j \in V} f_{ij} - \sum_{k \in V} (n - k + 1) p_{kj} \geq 0 \quad \forall i \in V \tag{2.21}$$

$$p_{ij} \geq 0 \quad \forall i, j \in V \tag{2.22}$$

onde $n = \mid V \mid$.

A restrição (2.20) garante que o primeiro vértice a ser utilizado é o vértice 1, isto é, o vértice origem. As restrições de acoplamento (2.21) são inequações válidas que conectam as variáveis f com as variáveis p. Essas restrições são obtidas a partir da associação entre

a ordem de visitação dos arcos e o fluxo que passa por eles. Sabe-se que o primeiro arco e ser percorrido tem fluxo n, o segundo, $n-1$, e assim sucessivamente. O conjunto de restrições (2.21) formaliza essa relação. Como será visto posteriormente essas restrições impactam tanto no tempo de solução como no *gap* de relaxação linear do modelo. Essa nova formulação que acrescenta as restrições (2.18)-(2.22) à formulação F1_PML é chamada de F2_PML.

2.4.2 Formulação de Fluxos Multiproduto para o Problema de Mínima Latência

Utilizando como base a formulação de fluxos (2.9)-(2.17) apresentada na seção (2.4.1), nesta seção é apresentada uma nova formulação de fluxos para o PML. Essa nova formulação ao invés de ser uniproduto é multiproduto. O intuito de trabalhar com uma formulação multiproduto é melhorar ainda mais os *gaps* de relaxação linear e, conseqüentemente, tentar melhorar os tempos de computação da formulação (2.9)-(2.17). Para a formulação multiproduto, também são apresentadas novas variáveis e inequações válidas, da mesma forma que na seção (2.4.1.1) é apresentada uma extensão para a formulação uniproduto do PML.

A formulação multiproduto para o PML, denominada de F3_PML, possui o seguinte conjunto de variáveis:

$$x_{ij} = \begin{cases} 1 & \text{se o veículo atravessa o arco } (i,j) \\ 0 & \text{caso contrário.} \end{cases}$$

f_{ijk}: fração da demanda total do produto k transportada no arco (i,j) destinado ao vértice k.

e o seguinte conjunto de parâmetros:

c_{ij}: custo unitário pago pelo operador logístico para atravessar o arco (i,j).

$$\text{minimize} \sum_{i,j,k \in V | i \neq j} c_{ij} f_{ijk} \tag{2.23}$$

sujeito a

$$\sum_{i \in V | i \neq j} x_{ij} = \quad 1 \qquad \forall j \in V \tag{2.24}$$

$$\sum_{j \in V | i \neq j} x_{ij} = \quad 1 \qquad \forall i \in V \tag{2.25}$$

$$f_{ijk} \leq \quad x_{ij} \qquad \forall i, j, k \in V, i \neq j \tag{2.26}$$

$$\sum_{j \in V | j \neq 1} f_{1j1} = \quad 1 \tag{2.27}$$

$$\sum_{j \in V | j \neq 1} (f_{1jk} - f_{j1k}) = \quad 1 \qquad \forall k \in V, k \neq 1 \tag{2.28}$$

$$\sum_{i \in V | i \neq 1} f_{i11} = \quad 1 \tag{2.29}$$

$$\sum_{i \in V | i \neq k} (f_{ikk} - f_{kik}) = \quad 1 \qquad \forall k \in V, k \neq 1 \tag{2.30}$$

$$\sum_{i \in V | i \neq j, i \neq k} (f_{jik} - f_{ijk}) = \quad 0 \qquad \forall k, j \in V, j \neq k, j \neq 1 \tag{2.31}$$

$$\sum_{i,j,k \in V | i \neq j} f_{ijk} = \quad \frac{n(n+1)}{2} \tag{2.32}$$

$$f_{ijk} \geq \quad 0 \qquad \forall i, j, k \in V, i \neq j \tag{2.33}$$

$$x_{ij} \in \quad \{0, 1\} \qquad \forall i, j \in V, i \neq j \tag{2.34}$$

onde $n =\mid V \mid$.

A função objetivo (2.23) soma os custos para todos os arcos da rede. Esse custo está associado à transferência, pelos arcos do caminho, de todos os produtos, do vértice origem ao específico vértice de destino.

As equações (2.24) e (2.25) são as restrições de atribuição introduzidas em 1954 por Dantzig, Fulkerson e Johnson [DFJ54]. Elas garantem que existe somente um arco saindo e um arco chegando em cada vértice. As restrições (2.26) acoplam as variáveis x e as f.

As equações (2.27)-(2.31) garantem a conservação de fluxo na rede. Segundo Laporte e Norbert [LN87], esse tipo de restrições de fluxo é baseada no trabalho de Garvin *et al.* [GCJS57]. A equação redundante (2.32) assegura que o número de fluxos unitários que percorrem todos os arcos é expresso por $\frac{n(n+1)}{2}$, onde $n =\mid V \mid$ (2.32).

O fato que todas as variáveis x_{ij} são binárias é assegurado pelas restrições (2.34). Na formulação (2.23)-(2.34) as demandas estão normalizadas.

2.4.2.1 Uma formulação Multiproduto estendida para o Problema de Mínima Latência

Para a formulação F3_PML, também é apresentada uma versão estendida, chamada de F4_PML. Essa versão, assim como a formulação estendida da formulação F1_PML, possui um conjunto de variáveis p_{ij}, definindo a ordem j de visitação do vértice i. Essas variáveis são inspiradas nas variáveis binárias do Problema Quadrático de Atribuição [KB57], que quando são usadas para resolver instâncias do PCV definem a ordem de visitação dos vértices. Para essa formulação, as restrições de acoplamento tendem a ser mais fortes que as restrições (2.18)-(2.22) devido às características multiproduto da formulação F3_PML.

A formulação F4_PML é a formulação F3_PML acrescida das seguintes restrições:

$$\sum_{i \in V} p_{ij} = 1 \qquad \forall j \in V \tag{2.35}$$

$$\sum_{j \in V} p_{ij} = 1 \qquad \forall i \in V \tag{2.36}$$

$$p_{11} = 1 \tag{2.37}$$

$$p_{ij} \geq 0 \qquad \forall i, j \in V \tag{2.38}$$

$$\sum_{k,j \in V | i \neq j} f_{ijk} = \sum_{l=1}^{n} (n - l + 1) p_{li} \qquad \forall i \in V \tag{2.39}$$

$$\sum_{i,j \in V | i \neq j} f_{ijk} = \sum_{l=1}^{n} (l - 1) p_{lk} \qquad \forall k \in V, k \neq 1 \tag{2.40}$$

onde $n = \mid V \mid$.

As restrições (2.35) e (2.36) também são restrições de atribuição, da mesma forma que as restrições (2.24) e (2.25). Entretanto, estas últimas garantem que um dado vértice somente é visitado em uma dada ordem de visitação e que em uma ordem de visitação somente um vértice é visitado. Note que enquanto as restrição (2.24) e (2.25) não são definidas para i e j iguais, o mesmo não acontece para as restrições (2.35) e (2.36), já que um vértice n pode ser visitado na *n-ésima* posição. A restrição (2.37) garante que o primeiro vértice a ser utilizado é o vértice 1, isto é, o vértice origem. As restrições de acoplamento (2.39) e (2.40) conectam as variáveis f com as variáveis p. Essas restrições foram criadas observando a relação entre a ordem de visitação dos arcos e o fluxo de cada produto que percorre esse arco. Como será visto posteriormente, essas restrições impactam tanto no tempo de solução do modelo como no *gap* de relaxação linear.

2.5 Formulação para o PML baseada no Problema Quadrático de Atribuição

2.5.1 Problema Quadrático de Atribuição

O Problema Quadrático de Atribuição (PQA), do inglês, *Quadratic Assignment Problem*, atribui facilidades para localidades. O objetivo do PQA é minimizar o custo total de transporte dos produtos intermediários. O PQA foi inicialmente proposto em 1957 por Koopmans e Beckmann [KB57] e desde então tem sido amplamente estudado, tanto na procura de métodos de solução mais rápidos e eficientes, como na procura de melhores limites e linearizações. Existem muitas técnicas de obtenção de limites inferiores para o PQA baseadas em programação linear. Dentre essas técnicas se destaca a apresentada por Adams e Johnson em 1994 [AJ94]. Apesar de gerar bons limites, os próprios autores relatam que o custo computacional dessa técnica é proibitivo.

Em 1976, Sahni e Gonzalez [SG76] mostraram que o PQA é um problema NP-Difícil. Até hoje problemas de tamanho maior que 30 ainda são um desafio para os pesquisadores.

De acordo com [KB57] o modelo para o PQA pode ser definido como: dado um conjunto K de n facilidades, um conjunto I de n localidades, duas matrizes de duas dimensões representando, respectivamente, custos entre pares de localidades e demandas entre pares de facilidades, pergunta-se: onde devem ser localizadas as facilidades para que o custo do transporte dos produtos intermediários seja minimizado? Os autores também definem custos de transporte entre produtos intermediários como $b_{kl}c_{ij}x_{ki}x_{lj} \quad \forall k, l \in K, \forall i, j \in I$, sendo b_{kl} a demanda entre o par de facilidades kl, c_{ij}, o custo de transporte entre o par ij, e x_{ki} (x_{lj}) uma variável inteira que estabelece a atribuição entre a facilidade k (l) e a localidade i (j). As variáveis x devem constituir uma atribuição, implicando na seguinte formulação:

$$\text{minimizar} \quad \sum_{k \in K} \sum_{i \in I} \sum_{l \in K} \sum_{j \in I} b_{kl} c_{ij} x_{ki} x_{lj} \tag{2.41}$$

sujeito a

$$\sum_{k \in K} x_{ki} = \quad 1 \qquad \forall i \in I \tag{2.42}$$

$$\sum_{i \in I} x_{ki} = \quad 1 \qquad \forall k \in K \tag{2.43}$$

$$x_{ki} \in \quad \{0,1\} \qquad \forall k \in K, i \in I \tag{2.44}$$

2.5.2 PQA como Problema de Mínima Latência

No artigo original do PQA, Koopmans e Beckmann [KB57] mostram que esse problema é uma generalização do Problema do Caixeiro Viajante (PCV). Eles mostram que, se a matriz b_{kl} for da forma descrita abaixo, o PQA se transforma em um PCV.

$$b_{kl} = \begin{bmatrix} 0 & 1 & 0 & . & . & 0 \\ 0 & 0 & 1 & . & . & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & 1 \\ 1 & 0 & 0 & . & . & 0 \end{bmatrix}$$

Padronizando a origem como posição 1, a primeira localidade a ser visitada como posição 2, a segunda como posição 3, e assim sucessivamente, é possível considerar:

- Um e somente um produto é entregue em cada localidade;
- Primeiramente um produto é entregue na posição 2, depois um produto é entregue na posição 3, e assim sucessivamente;
- A última localidade a ser visitada é a origem;
- A cada localidade visitada é acrescentado o custo do respectivo arco.

Como a diferença entre o PCV e o PML está apenas na função objetivo, então é possível transformar o PQA no PML apenas manipulando a matriz b_{kl}. Para isso é preciso lembrar que o PML é um PCV com uma contabilização de custo diferente. No PML o custo do primeiro arco a ser visitado é multiplicado por n, o segundo por $n-1$, e assim sucessivamente, até o último arco a ser visitado, que termina na origem, e onde o custo é multiplicado por uma unidade.

Dessa forma, se a matriz b_{kl} original do PQA for modificada para a matriz mostrada abaixo, e se for adicionada uma restrição que obrigue que o primeiro vértice seja o vértice origem, o PQA será transformado em um PML.

$$b_{kl} = \begin{bmatrix} 0 & n & 0 & . & . & 0 \\ 0 & 0 & n-1 & . & . & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & 2 \\ 1 & 0 & 0 & . & . & 0 \end{bmatrix}$$

Essa idéia de modificar a matriz B transformando o PQA em problema de escalonamento de tarefas foi apresentada por Geoffrion e Graves [GG76].

2.5.3 Formulação de Fluxos para o Problema Quadrático de Atribuição

Existem diferentes formulações matemáticas para o PQA, cada uma com diferentes características estruturais que abrangem diferentes técnicas de solução. De acordo com [Law63] [AJ94], é mais difícil desenvolver estratégias de solução para formulações quadráticas inteiras, como a formulação (2.41)-(2.44). Por esse motivo é importante o estudo de técnicas que reescrevem o PQA como um problema de programação linear inteira.

Nos últimos 40 anos muitos pesquisadores têm trabalhado em propor métodos para linearizar o termo quadrático da função objetivo acrescentando variáveis adicionais. O trabalho de Lawler [Law63] é uma referência, derivando no limite de Gilmore-Lawler e em uma família de linearizações correlatas.

Muitas dessas linearizações utilizam variáveis de fluxos como variáveis adicionais com o intuito de eliminar o termo quadrático da função objetivo.

A formulação de Adams e Johnson [AJ94], que também pode ser interpretada através de variáveis de fluxo, é mostrada a seguir:

$$\text{minimizar} \sum_{k \in K} \sum_{i \in I} \sum_{l \in K} \sum_{j \in I} b_{kl} c_{ij} y_{kilj} \tag{2.45}$$

sujeito às restrições (2.42), (2.43), (2.44) e

$$\sum_{j \in I} y_{kilj} = x_{ki} \quad \forall k, l \in K, i \in I, i \neq j, k \neq l \tag{2.46}$$

$$\sum_{l \in K} y_{kilj} = x_{ki} \quad \forall k \in K, i, j \in I, i \neq j, k \neq l \tag{2.47}$$

$$y_{kilj} = y_{ljki} \quad \forall k, l \in K, i, j \in I, i \neq j, k < l \tag{2.48}$$

$$y_{kilj} \geq 0 \quad \forall k, l \in K, i, j \in I, i \neq j, k \neq l \tag{2.49}$$

Hahn *et al.* [HHJ+01] mostram como eliminar as restrições (2.48), gerando a formulação:

$$\text{minimizar} \sum_{k \in K} \sum_{i \in I | l \neq k} \sum_{l \in K} \sum_{j \in I | i \neq j} [b_{kl} c_{ij} + b_{lk} c_{ji}] y_{kilj} \tag{2.50}$$

sujeito às restrições (2.42), (2.43), (2.44) e

$$\sum_{j \in I | j \neq i} y_{kilj} = x_{ki} \quad \forall k, l \in K, k < l, i \in I \tag{2.51}$$

$$\sum_{l \in K | j \neq i} y_{ljki} = x_{ki} \quad \forall k, l \in K, k > l, i \in I \tag{2.52}$$

$$\sum_{l \in K | k \leq l} y_{kilj} + \sum_{l \in K | k \geq l} y_{ljki} = x_{ki} \quad \forall k \in K, i, j \in I, i \neq j \tag{2.53}$$

$$y_{kilj} \geq 0 \quad \forall k, l \in K, i, j \in I, i \neq j, k \neq l \tag{2.54}$$

É possível verificar que o produto das variáveis de atribuição pode ser interpretado como as variáveis de fluxo entre as facilidades k e l através do arco ij, representado pela variável f_{klij}, originando a seguinte formulação:

$$\text{minimizar} \sum_{(i,j) \in I^2 | i \neq j} \sum_{(k,l) \in K^2 | k < l} [c_{ij} b_{kl} + c_{ji} b_{lk}] f_{klij} \tag{2.55}$$

sujeito às restrições (2.42), (2.43), (2.44) e

$$\sum_{j \in I | j \neq i} f_{klij} = x_{ki} \quad \forall i \in I, \forall (k,l) \in K, k < l \tag{2.56}$$

$$\sum_{i \in I | i \neq j} f_{klij} = x_{lj} \quad \forall j \in I, \forall (k,l) \in K, k < l \tag{2.57}$$

$$\sum_{l \in K | k < l} f_{klij} + \sum_{l \in K | k > l} f_{lkji} = x_{ki} \quad \forall k \in K, \forall i, j \in I, i \neq j \tag{2.58}$$

$$f_{klij} \geq 0 \quad \forall i, j \in I, i \neq j, \forall (k,l) \in K \tag{2.59}$$

As restrições (2.56) podem ser vistas como fluxos de produtos que deixam a localidade i enquanto as restrições (2.57), fluxos que chegam na localidade j, para um dado par kl. As restrições (2.58) asseguram que, se a facilidade k é construída na localidade i, para cada arco (i,j) ou (j,i) existe somente uma facilidade l que é atendida nessa ligação.

Sarubbi, Miranda, Luna e Camargo [SMLC07] mostram, que se for considerada uma matriz de demanda esparsa, a formulação (2.55) - (2.59) tem uma função objetivo degenerada, uma vez que possui variáveis de fluxo desnecessárias. Em [SMLC07] é desenvolvido um modelo para o PQA com matrizes esparsas que utiliza uma regra para a eliminação de diversas variáveis de fluxo.

Sarubbi *et al.* [SMLC07] com o objetivo de descartar variáveis de fluxo criaram o seguinte conjunto:

$$\theta = \{(k,l) \in K^2 \mid k < l \text{ e } (b_{kl} \neq 0 \text{ ou } b_{lk} \neq 0)\}$$

A definição do conjunto θ evita a criação de variáveis de fluxo para pares kl quando ambos os coeficientes de demanda são nulos. Dessa forma, o PQA para uma matriz de demanda esparsa pode ser formulado como o seguinte modelo M1_PML:

$$\text{minimizar} \sum_{(i,j) \in I^2 | i \neq j} \sum_{(k,l) \in \theta} [c_{ij} b_{kl} + c_{ji} b_{lk}] f_{klij} \tag{2.60}$$

sujeito às restrições (2.42), (2.43), (2.44) e

$$\sum_{j \in I | j \neq i} f_{klij} = x_{ki} \quad \forall i \in I, \forall (k,l) \in \theta \tag{2.61}$$

$$\sum_{i \in I | i \neq j} f_{klij} = x_{lj} \quad \forall j \in I, \forall (k,l) \in \theta \tag{2.62}$$

$$\sum_{l \in \theta | k < l} f_{klij} + \sum_{l \in \theta | k > l} f_{lkji} \leq x_{ki} \quad \forall k \in K, \forall i,j \in I, i \neq j \tag{2.63}$$

$$f_{klij} \geq 0 \quad \forall i,j \in I, i \neq j, \forall (k,l) \in \theta \tag{2.64}$$

2.5.4 Formulação de Fluxos para o Problema de Mínima Latência baseada no Problema Quadrático de Atribuição

Como a matriz b_{kl} que transforma o PQA no PML é muito esparsa, é possível utilizar a formulação (2.60)-(2.64) para computar, de forma eficiente, um limite inferior justo para o PML. Uma restrição adicional deve ser inserida no modelo para garantir que o vértice origem seja o primeiro a ser visitado. Essa restrição pode ser escrita como:

$$x_{11} = 1 \tag{2.65}$$

Como o modelo matemático proposto por [SMLC07], (2.60)-(2.64), é eficiente para PQA com matrizes esparsas, e como foi visto na seção (2.5.2), a matriz b_{kl} que transforma o PQA em um PML é esparsa, o modelo (2.60)-(2.64), acrescido da restrição (2.65), pode ser usado para resolver o PML.

Como será visto posteriormente, a formulação (2.60)-(2.65) é muito forte e gera limites inferiores muito próximos do ótimo inteiro. Apesar disso, para as instâncias maiores que 30 vértices, não foi possível, utilizando os recursos de máquina e *software* disponíveis, provar a otimalidade dessas instâncias em um tempo competitivo. A abordagem utilizada para resolver instâncias maiores, no ótimo, foi relaxar as restrições (2.63) gerando uma formulação com menos restrições e mais fácil de ser resolvida. Essa nova formulação, denominada de *Formulação_Relaxada*, além de ser mais rápida, gera limites de relaxação linear não muito distantes da formulação (2.60)-(2.65). Esses limites são apresentados na seção de resultados computacionais.

2.5.4.1 Formulação de Fluxos com 3 Índices para o Problema de Mínima Latência Baseada no Problema Quadrático de Atribuição

A formulação (2.60)-(2.65), definida na seção anterior, possui variáveis com quatro índices, as variáveis f_{klij}. Entretando, quando as restrições (2.63) são retiradas, gerando a *Formulação_Relaxada*, as variáveis de 4 índices podem ser reescritas como variáveis de 3 índices. Isso se deve ao índice l das variáveis da *Formulação_Relaxada* que pode ser eliminado sem perda de informação para a formulação. Sendo assim, é possível escrever uma formulação de 3 índices equivalente, em termos de limite de programação linear, à *Formulação_Relaxada*. Para isso é definido o seguinte conjunto de variáveis:

$$x_{ki} = \begin{cases} 1 & \text{se o vértice } i \text{ é visitado na ordem } k \\ 0 & \text{caso contrário.} \end{cases}$$

$f_{kij} \geq 0$: fluxo transportado pelo arco (i, j) quando o mesmo é visitado na posição k.
e o seguinte conjunto de parâmetros:

c_{ij}: custo de arco (i, j).

Tem-se então a seguinte formulação de programação linear inteira mista, M2_PML:

$$\min \sum_{k=1}^{n} \sum_{(i,j)\in A} (n-k+1) c_{ij} f_{kij} \tag{2.66}$$

sujeito a

$$\sum_{k=1}^{n} x_{kj} = 1 \qquad \forall j \in V, \tag{2.67}$$

$$\sum_{j\in V} x_{kj} = 1 \qquad \forall k = 1...n, \tag{2.68}$$

$$\sum_{j\in V} f_{kij} = x_{ki} \qquad \forall k = 1...n, \ \ \forall i \in V \ \ i \neq j \tag{2.69}$$

$$\sum_{i\in V} f_{kij} = x_{(k+1)j} \qquad \forall k = 1...n, \ \ \forall j \in V \ \ i \neq j \tag{2.70}$$

$$x_{11} = 1 \tag{2.71}$$

$$x_{(n+1)i} = 1 \tag{2.72}$$

$$f_{ijk} \geq 0 \qquad \forall i \in V, \ \ \forall j \in V \ \ \forall k = 1...(n+1) \tag{2.73}$$

$$x_{ki} \in \{0, 1\} \qquad \forall i \in V \ \ \forall k = 1...(n+1) \tag{2.74}$$

Em 1995, Vander Wiel e Sahinidis [WS95] apresentaram uma formulação muito similar para o PCVDT. O PCVDT e o PML são fortemente relacionados, mas não são o mesmo problema. Como no PML a escolha do vértice origem é relevante, não é possível usar diretamente a formulação de Vander Wiel e Sahinidis sem assegurar que um vértice origem seja dado. Também é necessário definir um conjunto de posições K variando de 1 a $n-1$, modificando as restrições de acoplamento (2.69) e (2.70). Uma restrição adicional deve garantir que o último vértice visitado também é a origem. Para isso é forçada a existência de um fluxo não-nulo f_{i1n}.

2.6 Formulação para o PML baseada no Problema do Caminho Mínimo com Restrições Adicionais

Outra estratégia de resolver de forma exata e de computar limites inferiores justos para o PML é considerar um grafo $G = (V, A)$ dirigido e conexo, onde V representa um conjunto de vértices e A é uma coleção de arcos. Esse grafo G possui uma origem 1. Entretanto, ao invés de se trabalhar com o grafo G original, é preciso transformar esse em um grafo multinível $G'(V', A')$. Nesse grafo G', $V - \{1\}$ vértices são criados para cada $V - \{1\}$ vértice do grafo original G. Cada um desses $V - \{1\}$ vértices que são criados no grafo G' pode ser interpretado como um nível k. Um vértice artificial, $1'$, representando um espelho do vértice origem, é criado e ligado a todos os vértices do último nível k. Dessa forma, o grafo G' possui $\mid V \mid$ níveis. Cada nível representa a ordem de visitação do seu respectivo vértice na seqüência. Uma vez que no PML o custo depende da seqüência, para cada arco o custo original será multiplicado por um fator F_{fator}. Esse F_{fator} é calculado pela expressão $F_{fator} = \mid V \mid -k+1$. As Figuras (2.4) e (2.5) ilustram como é representado um grafo multinível G' oriundo de um simples grafo G de 4 vértices. O procedimento para obter o grafo G' a partir do grafo G original é mostrado no algoritmo (2.1).

Criado o grafo G' a partir do grafo G, o objetivo se torna encontrar o menor caminho, em G', do vértice origem 1 para o vértice origem espelhado $1'$, vide Figura (2.5). Esse caminho precisa percorrer todos os níveis somente uma vez. Isso é facilmente garantido criando G' como um dígrafo e inibindo arcos entre vértices de mesmo nível. O problema desse grafo é que vértices em diferentes níveis podem estar associados a um mesmo vértice no grafo original G. Um exemplo são os vértices 2_1 e 2_3 da Figura (2.5), pois ambos estão associados ao vértice 2 do grafo original, apesar de um estar associado à posição 1 e outro à posição 3. Na solução do PML esses dois vértices não podem ser visitados no mesmo

Algoritmo 2.1: Método para Transformar Grafo G no Grafo G'

```
Data: G(V, A), n
NumeroVerticesGrafoG'= (n-1)(n-1)+2;
for i ← 2 to n do
 | c'[1][i] = n c[1][i];
end
for k ← 2 to n − 1 do
 | for i ← 2 + (n − 1)(k − 2) to n + (n − 1)(k − 2) do
 | | for j ← (n + 1) + (n − 1)(k − 2) to 2n − 1 + (n − 1)(k − 2) do
 | | | i1= (i-(n-1)(k-1));
 | | | j1= (j-(n-1)(k-2));
 | | | c'[i][j]= (n-1+k) c[i1][j1];
 | | end
 | end
end
i=1;
for j ← NumeroVerticesGrafoG' − n + 1 to NumeroVerticesGrafoG' − 1 do
 | c'[j][NumeroVerticesGrafoG'] = c[i][1];
 | j=j+1;
end
```

caminho, mas, como pode-se ver na Figura (2.5), o grafo G' permite que isso ocorra. Para garantir que vértices em níveis diferentes, mas que estejam associados ao mesmo vértice no grafo original, não sejam visitados duas vezes, é preciso que esse caminho mínimo respeite algumas restrições adicionais.

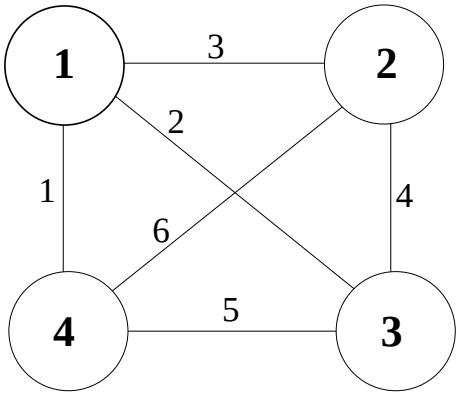


Figura 2.4: Grafo Exemplo

Para criar uma formulação de programação linear inteira mista que represente o PML como um problema de caminho mínimo com restrições adicionais, é definido o seguinte

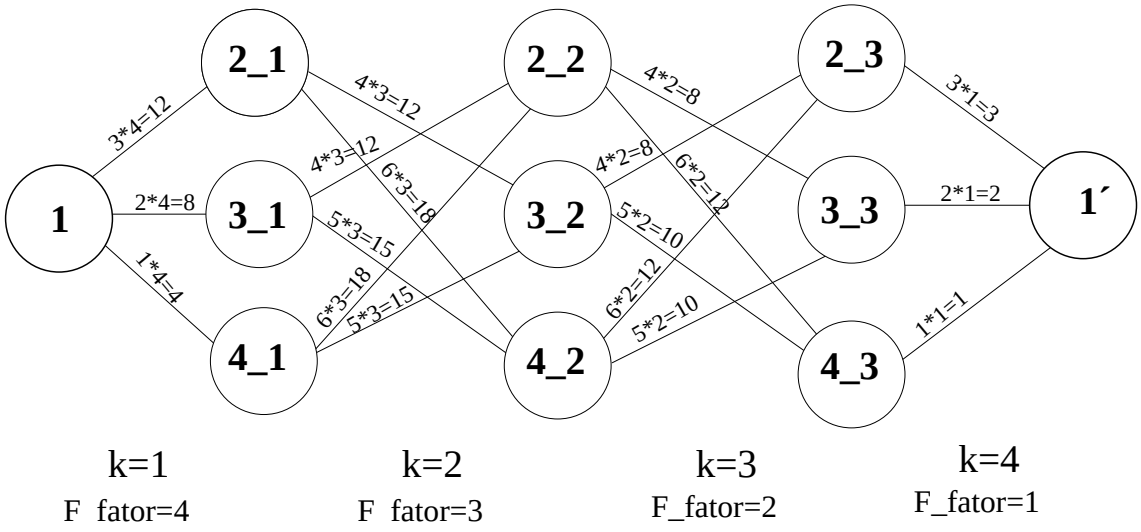


Figura 2.5: Grafo multinível G'

conjunto de variáveis:

$$f_{ijk} = \begin{cases} 1 & \text{se o arco } (i,j) \text{ é visitado na ordem } k \\ 0 & \text{caso contrário.} \end{cases}$$

e o seguinte conjunto de parâmetros:

c_{ij}: custo fixo pago pelo operador logístico para atravessar o arco (i, j).

A formulação matemática é definida como:

$$\min \sum_{j|j>1} n\ c_{1j} f_{1j1} + \sum_{k=2}^{n-1} \sum_{i=2}^{n} \sum_{j=2|j\neq i}^{n} (n-k+1)\ c_{ij} f_{ijk} + \sum_{i=2}^{n} c_{i1} f_{i1n} \tag{2.75}$$

sujeito a:

$$\sum_{j=2}^{n} f_{1j1} = \quad 1 \tag{2.76}$$

$$\sum_{j=2}^{n} f_{j1n} = \quad 1 \tag{2.77}$$

$$\sum_{j=2|i\neq j}^{n} f_{ij2} = \quad f_{1i1} \qquad \forall i = 2,..,n \tag{2.78}$$

$$\sum_{j=2|i\neq j}^{n} (f_{jik} - f_{ij(k+1)}) = \quad 0 \qquad \forall k = 2,..,n-2 \quad \forall i = 2,..,n \tag{2.79}$$

$$\sum_{i=2|i\neq j}^{n} f_{ij(n-1)} = \quad f_{j1n} \qquad \forall j = 2,..,n \tag{2.80}$$

$$\sum_{i=2|i\neq j}^{n} \sum_{k=2}^{n-1} f_{ijk} = \quad 1 - f_{1j1} \qquad \forall j = 2,..,n \tag{2.81}$$

$$f_{ijk} \in \quad \{0,1\} \qquad \forall i,j,k = 1,..,n \tag{2.82}$$

A função objetivo (2.75) soma os custos de todos os arcos do caminho. É importante destacar que existem apenas $\mid V \mid$ arcos incidentes tanto na origem como no vértice origem espelhado. A função objetivo possui 3 termos. O primeiro se refere aos arcos ligados à origem 1, o último aos arcos ligados à origem espelhada $1'$, e o segundo, ao restante dos arcos. Restrições (2.76) e (2.77) garantem que apenas uma unidade de fluxo sairá da origem e chegará na origem espelhada $1'$. Restrições (2.78), (2.79) e (2.80) asseguram a conservação do fluxo. Os conjuntos de restrições (2.78) e (2.80) são relativos à origem e à origem espelhada, respectivamente. Já o conjunto de restrições (2.79) se refere aos arcos do grafo entre o nível 1 e o $\mid V \mid -\{1\}$. Restrições (2.81) são responsáveis em transformar um simples problema de caminho mínimo em um PML. Essas restrições garantem que existe apenas um arco incidente em cada conjunto de vértices para todos os níveis k, forçando a solução do caminho mínimo a se tornar a solução do PML. Restrições (2.82) asseguram que todas as variáveis de fluxo f_{ijk} são binárias.

Observa-se que para um grafo completo G', o modelo (2.75)-(2.82) possui $2*(n-1)+(n-2)*(n-1)*(n-2)$ variáveis, onde $n =\mid V \mid$. Apesar do fato de esse modelo possuir n^3 variáveis binárias, ele possui menos variáveis que os modelos apresentados na seção (2.5.2). Na seção (2.8), será mostrado, empiricamente, que a formulação (2.75)-(2.82) é mais rápida

que as três formulações baseadas no PQA e que foram apresentadas na seção (2.5.2).

2.7 Uma Meta-heurística GRASP para o Problema de Mínima Latência

De acordo com Vander Wiel e Sahinidis [WS95], existe um número limitado de heurísticas para o PCVDT. Em sua abordagem exata para o PML, Lucena [Luc90] apresenta uma adaptação de uma heurística de troca usada no PCV. Ele gera limites superiores justos para instâncias aleatórias de até 30 vértices. Em 1993, Bianco, Mingozzi e Ricciardelli [BMR93] usaram o algoritmo do *Vizinho Mais Próximo* [JM97] para gerar uma solução inicial que depois foi melhorada utilizando-se uma heurística de trocas. Para instâncias de até 35 vértices a heurística apresentou soluções a 6% do ótimo e nenhuma solução ótima foi encontrada com essa técnica. Os mesmos autores apresentaram uma heurística baseada em programação dinâmica que ofereceu resultados melhores. Bianco, Mingozzi e Ricciardelli [BMR93] mostraram soluções, em média, a 4.2% do ótimo para instâncias de até 60 vértices. Essas instâncias usaram distância Euclidiana. Para instâncias geradas somente a partir de números aleatórios entre [1,100], os mesmos autores conseguiram resultados melhores, chegando a valores de, em média, 2.7% do ótimo. Em 1995, Fischetti, Laporte e Martello [FLM93] apresentaram uma heurística de trocas que gerou soluções a 14% do ótimo. Nesta seção é apresentada uma heurística para o PML. O objetivo de conseguir limites superiores perto do ótimo não se limita somente a encontrar uma solução justa em um tempo razoável. O intuito de se desenvolver uma heurística para o PML também está relacionado a facilitar o resolvedor CPLEX a encontrar a solução ótima. Isso pode ser conseguido utilizando a solução gerada pela meta-heurística como primeiro limite superior para o resolvedor. Esse limite superior, se for melhor que o primeiro limite superior encontrado pelo resolvedor, aumentará a eficiência do algoritmo *Branch-and-Bound* presente no CPLEX.

O método GRASP, do inglês, *Greedy Randomized Adaptive Search Procedure*, em português (procedimento de busca guloso, adaptativo e aleatório) [FR95] [RR03] [Rav07] [dSDR03], foi o escolhido para encontrar soluções viáveis para o PML.

Essa meta-heurística é dividida em duas fases: a construção de uma solução viável e um subseqüente procedimento de Busca Local. Essas duas fases são repetidas a cada iteração. Na Fase de Construção, uma função gulosa e aleatória é usada para construir uma solução inicial. Essa solução é então utilizada como solução inicial para a Fase de Busca Local. O algoritmo GRASP proposto também tem uma terceira fase chamada de

Path Relinking [Glo96] [LM99]. Essa fase é usada para tentar melhorar a solução dada pela Busca Local. O resultado final é simplesmente a melhor solução encontrada após todas as iterações.

Algoritmo 2.2: GRASP

```
Data: Max_Iteracoes
for i ← 1 to Max_Iteracoes do
    Fase_Construção();
    Busca_Local();
    if Solução_Encontrada_for_Melhor() then
        Atualize_Melhor_Solução();
        Busca_Local();
    else
        Path_Relinking(Solução_Busca_Local, Melhor_Solução);
    end
    if Solução_Encontrada_for_Melhor() then
        Atualize_Melhor_Solução();
        Busca_Local();
    end
end
```

Na Fase de Construção, uma técnica gulosa e aleatória gera soluções viáveis. Cada solução viável é iterativamente construída, um elemento por vez. Foi escolhido o algoritmo do vizinho mais próximo [JM97] como o algoritmo da Fase de Construção. Entretanto, ao invés de se escolher sempre o melhor elemento, isto é, o vizinho mais próximo, é criada uma lista restrita de candidatos (LRC), isto é, uma lista de bons elementos. Aleatoriamente, um desses elementos é escolhido a cada passo e não necessariamente o melhor elemento. Um parâmetro da LRC, α, determina o quanto o algoritmo é guloso ou aleatório. Quando $\alpha = 0$, tem-se uma solução totalmente gulosa. Por outro lado, quando $\alpha = 1$, tem-se uma solução totalmente aleatória. Outra característica interessante é que, ao invés de se trabalhar com um α fixo em cada iteração, trabalhou-se com um α variável dentro da mesma iteração. A cada passo da Fase de Construção, isto é, a cada novo vértice inserido na solução, o parâmetro α aumenta proporcionalmente de um $\alpha_{inicial}$ até um α_{final}. Essa estratégia foi escolhida pois os primeiros vértices do caminho têm mais influência que os últimos, no cálculo do custo do PML. Nesse problema, o custo de cada arco é multiplicado pelo número de arcos que faltam para completar o circuito Hamiltoniano. O algoritmo (2.3) apresenta o esquema que define a Fase de Construção proposta.

O algoritmo de Busca Local, que será apresentado, também difere dos algoritmos

Algoritmo 2.3: Fase de Construção

Data: $\alpha, \alpha_{inicial}, \alpha_{final}, N, Origem$
$\alpha = \alpha_{inicial}$;
Solução_Inicial = {Origem};
Vértice = Origem;
for $i \leftarrow 2$ *to* N **do**
    Construa_LRC(Vértice,α);
    Vértice = Aleatoriamente_Escolha_Próximo_Vértice_da_Lista_LRC();
    Solução_Inicial = Solução_Inicial $\cup$ {Vértice};
    $\alpha = \alpha + (\alpha_{final} - \alpha_{inicial})/N$;
end

padrões. Ao invés de se trabalhar com somente um algoritmo de Busca Local foi proposto o uso de 4 algoritmos simples que são encadeados. Após serem executadas as quatro buscas locais é verificado se em pelo menos alguma delas foi encontrada uma solução melhor. Se isso for verdade o procedimento de busca local é chamado novamente, mas agora a solução que será explorada é a melhor solução encontrada pela busca local na iteração anterior.

Primeiramente, é utilizado um algoritmo de troca de posições de vértices. Esse algoritmo troca a posição de todos os vértices, dois a dois. Um exemplo de uma simples troca entre dois elementos é mostrado na Figura (2.6). O segundo método é um algoritmo de inserção. Dada uma solução parcial oriunda da Fase de Construção, esse método de Busca Local funciona da seguinte forma: para cada elemento, salvo a origem, ele é inserido em todas as posições anteriores possíveis. Um exemplo de uma inserção simples é mostrado na Figura (2.7), onde o vértice 6 é colocado na segunda posição. O terceiro algoritmo é um tipo de algoritmo de troca de posições, mas ao invés de mudar-se todas as combinações de dois vértices, como na primeira Busca Local, são mudadas todas as combinações de dois vértices consecutivos. Um exemplo é mostrado na Figura (2.8). O último método de Busca Local é como o terceiro, mas nesse caso é mudado todas as combinações de três vértices consecutivos. Um exemplo é mostrado na Figura (2.9). Para as Figuras (2.6), (2.7), (2.8) e (2.9), a origem é sempre o vértice 1 e a ordem de visitação é dada pelo sentido horário a partir da origem.

Depois da Fase de Busca Local, é feita a seguinte análise: se em qualquer uma das quatro buscas Locais é encontrada uma solução melhor que a solução da presente na busca local anterior, essa nova solução é armazenada e as quatro buscas locais são executadas mais uma vez; caso contrário, é chamada a fase de *Path Relinking* (PR). *Path Relinking* foi

Algoritmo 2.4: Busca Local

Data: *it*
Procedimento_Troca_Dois_Vértices();
Procedimento_Inserção();
Procedimento_Troca_Dois_Vértices_Consecutivos();
Procedimento_Troca_Três_Vértices_Consecutivos();

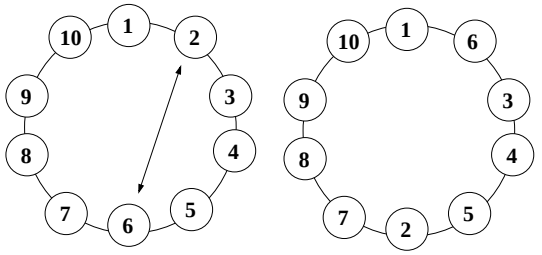


Figura 2.6: Busca Local Troca 2 Vértices.

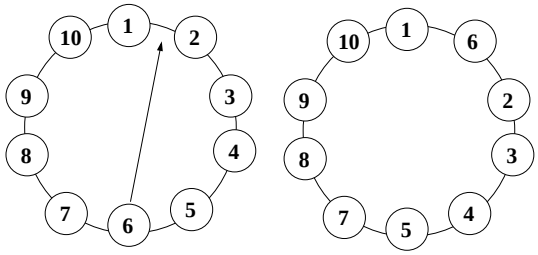


Figura 2.7: Busca Local de Inserção

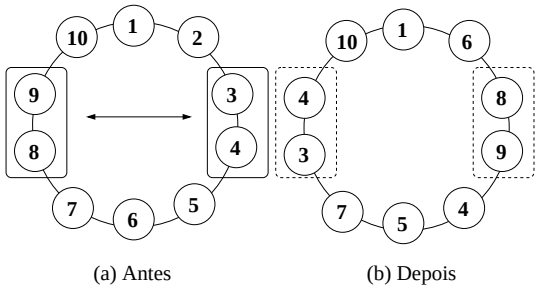


Figura 2.8: Busca Local Troca Conjunto de 2 Vértices.

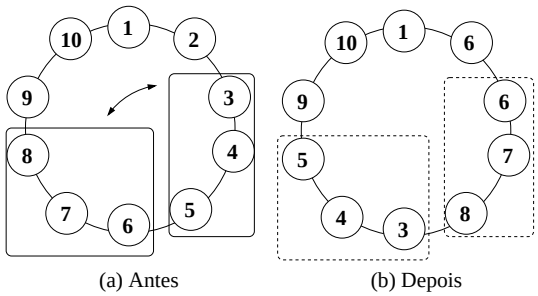


Figura 2.9: Busca Local Troca Conjunto de 3 Vértices.

proposto originalmente por Glover [Glo96] como uma estratégia de conectar boas soluções por Busca Tabu e *Scatter Search*. Existem muitas formas de implementar essa técnica. Para mais detalhes, vide [RR05]. *Path Relinking* com GRASP como uma estratégia aplicada em cada Fase de Busca Local foi primeiramente usado por Laguna e Martí [LM99]. Neste trabalho, PR funciona da seguinte forma: são consideradas todas as soluções intermediárias entre a melhor solução e a solução corrente. Esse caminho, *path*, é construído trocando-se posições entre dois vértices. Se uma solução melhor é encontrada, chama-se novamente a Fase de Busca Local, passando como parâmetro a nova solução.

2.8 Resultados Computacionais

Os testes foram realizados em uma máquina com um processador Pentium IV de 2.4Ghz com 1.0 Gbyte de memória RAM. O sistema operacional utilizado foi o Linux. Foi utilizado o resolvedor CPLEX 9.1.3 da ILOG. Foram utilizados dois conjuntos de instâncias. O primeiro conjunto foi gerado aleatoriamente, da mesma forma que Fischetti, Laporte e Martello fizeram em [FLM93]. O segundo conjunto utilizou a matriz de custos da biblioteca *QAPLIB*.

2.8.1 Instâncias geradas a partir de Fischetti, Laporte e Martello

Nesta seção, as instâncias foram geradas da seguinte forma: para cada classe de problemas foram criadas 10 instâncias de tamanho n. Cada uma dessas instâncias foi obtida gerando custos inteiros aleatórios no intervalo $[1, 100]$ e depois triangularizando esses valores através de um algoritmo de caminhos mínimos, da mesma forma que Fischetti, Laporte e Martello fizeram em [FLM93].

Para esse conjunto de instâncias, os resultados computacionais foram divididos em dois tipos: os resultados exatos e os resultados oriundos da heurística GRASP, isto é, os resultados heurísticos.

2.8.1.1 Resultados exatos

Todas as instâncias foram resolvidas utilizando oito formulações. A primeira chamada de F1_PML é a formulação de fluxos uniproduto (2.9)-(2.17). A segunda, chamada de F2_PML, é a formulação F1_PML acrescida das restrições (2.18)-(2.22). A terceira, chamada de F3_PML, é a formulação multiproduto (2.23)-(2.34). A quarta, chamada de F4_PML, é a formulação multiproduto (2.23)-(2.34) acrescida das restrições (2.35)-(2.40). A quinta, chamada de M1_PML, é a formulação baseada no PQA, isto é, a formulação (2.60)-(2.65) que possui 4 índices. A sexta formulação, chamada de M2_PML, é composta pelas equações (2.66)-(2.74). O modelo M2_PML possui variáveis de fluxo de 3 índices e é equivalente à *Formulação_Relaxada* que possui variáveis de fluxo de 4 índices. Como as formulações *Formulação_Relaxada* e (2.66)-(2.74) são equivalentes, foi escolhida uma para que o *software* CPLEX resolvesse. Foi escolhida então a formulação (2.66)-(2.74) como M2_PML, por essa possuir um número menor de variáveis que a *Formulação_Relaxada*. A formulação M3_PML, (2.75)-(2.82), é a baseada no Problema do Caminho Mínimo. A última, chamada de M4_PML, é a formulação apresentada por Bianco *et al.* [BMR93]

no trabalho intitulado *The Traveling Salesman Problem with Cumulative Costs*. Essa formulação é composta pelas equações (2.1) - (2.7). Nesse trabalho, os autores relatam que a formulação apresentada por eles é muito complexa para ser resolvida pelos resolvedores da época. Por isso, em [BMR93], nenhum resultado foi apresentado utilizando-se puramente essa formulação .

Para resolver de forma exata as instâncias do PML foi utilizado o resolvedor CPLEX. Todas as formulações utilizaram o Método da Barreira de Newton para calcular o limite de programação linear. As formulações F1_PML, F2_PML, F3_PML, F4_PML e M1_PML foram resolvidas uma vez para cada instância utilizando os parâmetros padrões do CPLEX. As formulações M2_PML, M3_PML e M4_PML foram resolvidas quatro vezes para cada instância: a primeira, com os parâmetros padrões do CPLEX, da mesma forma que a formulação M1_PML; a segunda, mudando-se alguns parâmetros do CPLEX; a terceira, com os parâmetros padrões do CPLEX, assim como a primeira, mas usando-se como primeiro limite superior para o CPLEX o valor calculado pela heurística GRASP; e, a quarta vez, não se utilizando os parâmetros padrões do CPLEX, assim como na segunda vez, e usando-se o limite superior pré-computado pela heurística GRASP, assim como na terceira vez.

A Tabela (2.1) relativa as formulações F1_PML, F2_PML, F3_PML e F4_PML apresenta sempre a média dos tempos e dos *gaps* para cada conjunto de 10 instâncias de mesmo tamanho. Nessa tabela tem-se: o tamanho dos problemas, campo *Tamanho*; a média dos *gaps* de relaxação linear, campos *Gap RL F1_PML (%)*, *Gap RL F2_PML (%)*, *Gap RL F3_PML (%)* e *Gap RL F4_PML (%)*; e a média dos tempos, em segundos, para calcular o ótimo inteiro, campos *Tempo (s) Ótimo F1_PML*, *Tempo (s) Ótimo F2_PML*, *Tempo (s) Ótimo F3_PML* e *Tempo (s) Ótimo F4_PML*.

Pela Tabela (2.1) é possível verificar que houve um ganho ao se utilizar as restrições redundantes (2.18)-(2.22) e (2.35)-(2.40). Isto é, as formulações F2_PML e F4_PML foram sempre melhores - em termos de *gap* de relaxação linear e tempo de computação - que as formulações F1_PML e F3_PML, respectivamente. O ganho no tempo também está relacionado ao *gap* de relaxação linear que diminui drasticamente quando se acrescentam as restrições redundantes das formulações F2_PML e F4_PML. Com isso, o procedimento de enumeração tende a ser mais rápido, pois o CPLEX precisa fechar um *gap* menor. Os experimentos da Tabela (2.1) comprovam a afirmação anterior, mesmo se sabendo que, no procedimento de enumeração, cada vértice da árvore de *Branch-and-Bound* se torna mais difícil de ser resolvido quando se acrescentam as restrições redundantes.

Tabela 2.1: Comparação dos modelos de Fluxos para o PML

Tamanho	Gap RL F1_PML (%)	Tempo (s) Ótimo F1_PML	Gap RL F2_PML (%)	Tempo (s) Ótimo F2_PML	Gap RL F3_PML (%)	Tempo (s) Ótimo F3_PML	Gap RL F4_PML (%)	Tempo (s) Ótimo F4_PML
10	20.96	0.29	11.60	0.29	4.77	0.87	0.30	0.22
12	16.96	0.54	11.32	0.52	5.14	2.65	0.08	0.22
15	21.87	13.43	11.82	1.56	8.07	23.88	0.23	3.03
20	21.10	2883.30	13.16	12.21	9.66	853.18	0.52	21.15
25	*****	*****	13.19	276.78	*****	*****	1.46	305.46
30	*****	*****	13.78	716.31	*****	*****	2.14	1355.13

Também é possível fazer uma comparação entre as formulações uniproduto - F1_PML e F2_PML - e as formulações multiproduto - F3_PML e F4_PML. Estas últimas apresentaram um *gap* de relaxação linear melhor que as primeiras, como é de se esperar. Entretanto, quando se analisa o tempo de computação, a formulação multiproduto nem sempre é mais rápida que a formulação uniproduto correspondente. Isso acontece principalmente devido ao número de variáveis da formulação uniproduto, que tende a ser bem menor que o número de variáveis da formulação multiproduto correspondente, tornando estas últimas mais difíceis.

A Tabela (2.2) relativa aos modelos M1_PML e M2_PML apresenta sempre a média dos tempos e dos *gaps* para cada conjunto de 10 instâncias de mesmo tamanho. Nessa tabela tem-se: o tamanho dos problemas, campo *Tamanho*; a média dos *gaps* de relaxação linear, campo *Gap RL M1_PML (%)*; a média dos tempos, em segundos, para calcular o limite inferior, campo *Tempo (s) RL M1_PML*; e, também, a média dos tempos, em segundos, para calcular o ótimo inteiro, campo *Tempo (s) Ótimo M1_PML*. Os 4 campos da tabela referentes ao modelo M1_PML são duplicados, mas usando-se como referência o modelo M2_PML. No campo *Tempo (s) Ótimo M2_PML*, a média dos tempos é calculada sempre utilizando o melhor resultado, para cada instância, entre a estratégia de utilizar o resolvedor CPLEX com os parâmetros padrões e a estratégia de utilizar o CPLEX com a ênfase na integralidade da solução. Isto é viável pois essas estratégias podem ser computadas em paralelo, quando uma conseguir provar a otimalidade, a outra técnica pode ser descartada. O último campo da Tabela (2.2), *Tempo (s) Ótimo M2_PML Usando LS GRASP*, também apresenta a média dos tempos para calcular o ótimo inteiro, para cada conjunto de 10 instâncias. Mas, nesse caso, o CPLEX utiliza como primeiro limite superior a solução encontrada pela heurística GRASP. Assim como no campo *Tempo (s) Ótimo M2_PML*, esse campo mostra a melhor solução utilizando duas abordagens diferentes para o resolvedor

Tabela 2.2: Comparação de Modelos para o PML Baseados no PQA

Tamanho	Gap RL M1_PML (%)	Tempo (s) RL M1_PML	Tempo (s) Ótimo M1_PML	Gap RL M2_PML (%)	Tempo (s) RL M2_PML	Tempo (s) Ótimo M2_PML	Tempo (s) Ótimo M2_PML Usando LS GRASP
10	0.14	0.02	0.04	1.06	0.01	0.04	0.04
12	0.72	0.04	0.27	1.17	0.02	0.05	0.05
15	0.13	0.16	0.41	0.23	0.05	0.09	0.08
20	0.70	0.54	8.95	0.72	0.14	0.81	0.56
25	0.86	3.37	56.09	1.26	0.14	3.04	1.99
30	0.95	10.42	136.24	1.85	0.71	11.34	7.77
35	1.32	32.57	711.07	1.89	1.47	37.77	28.26
40	1.25	85.39	****	1.66	2.35	84.38	67.27
45	1.10	130.19	****	1.66	3.90	263.28	175.34
50	1.18	221.92	****	1.57	5.68	1692.63	1353.96
55	1.72	367.11	****	2.08	8.34	4953.04	3278.87
60	1.05	614.43	****	1.49	11.65	17558.04	15566.82

CPLEX: a primeira com os parâmetros padrões, e a segunda, com a ênfase do resolvedor na integralidade da solução.

Nos campos *Gap RL M1_PML (%)* e *Gap RL M2_PML (%)* são apresentadas as médias dos *gaps* de relaxação linear para todos os conjuntos de instâncias. Cada um desses conjuntos possui 10 instâncias e, em cada uma delas, o *gap* de relaxação linear é calculado da seguinte forma:

$$Gap\ \ RL\ \ Mod(\%) = 100(\frac{VO - Limite\ Inferior}{VO}) \tag{2.83}$$

onde VO é o valor ótimo.

A Tabela (2.3) apresenta uma comparação entre duas formulações para o PML, ambas baseadas no Problema de Caminho Mínimo. A primeira formulação é composta das equações (2.75)-(2.82), que foi proposta neste trabalho e chamada de M3_PML. A segunda é a formulação apresentada por Bianco *et al.* [BMR93] e chamada de M4_PML. Os experimentos computacionais deste trabalho mostraram que os limites de programação linear são os mesmos tanto para a formulação M3_PML como para a M4_PML. Eles também mostram que os tempos para calcular esses limites são praticamente os mesmos em ambas as formulações M3_PML e M4_PML. Por isso, os campos relacionados aos limites de programação linear têm como valores os resultados encontrados pela formulação M3_PML.

A Tabela (2.3) possui os seguintes campos: *Tamanho*, que mostra o tamanho das instâncias testadas; *Tempo (s) Relaxação Linear*, apresenta o tempo, em segundos, para calcular o limite de programação linear usando-se a formulação M3_PML; *Gap (%) LI/Ótimo Médio* e *Gap (%) LI/Ótimo Máximo*, campos que mostram, respectivamente, a média dos

Tabela 2.3: Comparação de Modelos para o PML Baseados no Problema do Caminho Mínimo

Tamanho	Tempo (s) Relaxação Linear	Gap (%) LI/Ótimo Médio	Gap (%) LI/Ótimo Máximo	Tempo (s) M3_PML	Tempo (s) M3_PML com LS GRASP	Tempo (s) M4_PML	Tempo (s) M4_PML com LS GRASP
10	0.01	1.06	6.72	0.02	0.02	0.02	0.02
12	0.02	1.17	5.91	0.08	0.04	0.07	0.04
15	0.05	0.23	2.23	0.12	0.11	0.11	0.11
20	0.15	0.72	3.23	0.45	0.32	0.40	0.29
25	0.30	1.26	4.19	2.62	1.51	2.09	2.14
30	0.85	1.80	3.44	7.20	2.90	7.28	3.03
35	1.75	1.89	5.14	25.47	11.34	34.47	19.69
40	3.21	1.66	2.81	72.18	21.85	89.01	22.59
45	4.00	1.66	2.91	202.28	54.69	259.82	65.74
50	8.48	1.57	3.54	597.38	398.85	267.29	704.94
55	13.03	2.08	4.14	3982.42	2770.32	1038.31	648.95
60	20.62	1.49	2.22	4471.03	4920.19	1776.25	1273.06
65	26.45	1.86	3.47	19040.55	*****	*****	5619.83
70	39.05	1.66	2.68	25233.44	*****	*****	*****
75	55.44	1.69	2.64	26063.74	*****	*****	*****
80	81.89	1.05	1.54	61562.92	*****	*****	*****

gaps de relaxação linear para cada conjunto de instâncias e o maior *gap* de relaxação linear encontrado em cada conjunto; *Tempo (s) M3_PML* apresenta a média dentre os melhores tempos utilizando-se a formulação M3_PML com os parâmetros padrões do CPLEX e com o CPLEX configurado na ênfase da otimalidade da solução; *Tempo (s) M3_PML LS GRASP*, esse campo também apresenta a média dos melhores resultados utilizando-se a formulação M3_PML, mas nesse caso sempre é passado para o resolvedor CPLEX a solução encontrada pelo algoritmo GRASP como primeiro limite superior; Os campos *Tempo (s) M4_PML* e *Tempo (s) M4_PML com LS GRASP* são calculados da mesma forma que os dois anteriores, mas nesse caso utilizando-se a formulação M4_PML ao invés da M3_PML.

2.8.1.2 Resultados heurísticos

Para os experimentos heurísticos, Tabela (2.4), também são apresentadas as médias dos *gaps*, dos tempos e do número de iterações executadas pela heurística GRASP mostrada anteriormente. Essas médias são relativas a cada conjunto de 10 instâncias de tamanhos iguais, instâncias essas definidas na seção (2.8.1.1). A Tabela (2.4) possui os seguintes campos: *Tamanho*, que mostra qual o tamanho do conjunto de instâncias analisado. *Tempo (s) Solução GRASP*, campo que mostra a média dos tempos para se encontrar o limite superior executando o algoritmo GRASP com *Path Relinking* após um certo número de iterações. A média dessas iterações é apresentada no campo *Iterações GRASP Média*. O

campo *Gap Ótimo GRASP (%)* mostra a média dos *gaps*, em porcentagem, entre o valor ótimo e a solução encontrada pelo algoritmo GRASP. Esse *gap* é calculado pela seguinte fórmula:

$$Gap\ Otimo\ GRASP\ (\%) = 100(\frac{Sol\ GRASP - VO}{Sol\ GRASP}) \tag{2.84}$$

onde *VO* é o valor ótimo e *Sol Grasp* é a solução encontrada pelo algoritmo GRASP. Para cada uma das dez instâncias esse *gap* é calculado e, no campo *Gap Ótimo GRASP (%)*, é relatada a média aritmética desses *gaps*.

É representada nas tabelas a relação, em porcentagem, entre a melhor solução encontrada pelo algoritmo GRASP e o limite inferior encontrado pela relaxação linear dos modelos M1_PML e M2_PML. Esses valores estão armazenados nos campos (*Gap LI M1_PML GRASP (%)*) e (*Gap LI M2_PML GRASP (%)*) e, para cada instância, são calculados pela seguinte fórmula:

$$Gap\ LI\ Mod\ GRASP\ (\%) = 100(\frac{Sol\ GRASP - Limite\ Inferior}{Sol\ GRASP}) \tag{2.85}$$

onde *VO* é o valor ótimo e *Sol Grasp* é a solução encontrada pelo algoritmo GRASP. Assim como no campo *Gap Ótimo GRASP (%)*, os campos (*Gap LI M1_PML GRASP (%)*) e (*Gap LI M2_PML GRASP (%)*) apresentam somente a média aritmética dos *gaps* para cada conjunto de 10 instâncias.

O limite inferior é relativo ao modelo correspondente, isto é, o campo (*Gap LI M1_PML GRASP (%)*) está relacionado ao limite inferior do modelo M1_PML, assim como o campo (*Gap LI M2_PML GRASP (%)*) está associado ao limite inferior do modelo M2_PML. É importante lembrar que os limites de programação linear das formulações M2_PML, M3_PML e M4_PML são os mesmos, tornando irrelevante criar outros campos para a Tabela (2.4).

O algoritmo GRASP teve como parâmetros de entrada: $\alpha_{inicial} = 0.2$ e $\alpha_{final} = 0.5$ para as instâncias de até 35 vértices. Para as maiores utilizou-se também $\alpha_{inicial}$ igual a 0.05, 0.08, 0.10, 0.15, 0.18, 0.20 e 0.25. O resultado utilizado foi então o menor limite superior encontrado para cada um dos $\alpha_{inicial}$ citados anteriormente.

2.8.2 Instâncias baseadas na Biblioteca QAPLIB

Uma vez que o Problema Quadrático de Atribuição é uma generalização do PML, utilizaram-se instâncias baseadas na biblioteca $QAPLIB$. Essas instâncias foram reescritas descar-

Tabela 2.4: Resultados GRASP para o PML

Tamanho	Tempo (s) Solução GRASP	Iterações GRASP Média	Gap Ótimo GRASP (%)	Gap LI M1_PML GRASP (%)	Gap LI M2_PML GRASP (%)
10	0.00	1.9	0.00	0.14	1.16
12	0.00	27	0.00	0.72	1.17
15	0.00	26.6	0.00	0.12	0.23
20	0.06	72.7	0.00	0.72	0.70
25	0.93	166	0.00	0.22	0.28
30	2.99	634.3	0.06	0.98	1.90
35	7.06	745.4	0.00	1.32	1.89
40	81.05	5936.3	0.04	1.33	1.77
45	110.88	5594.2	0.03	1.11	1.69
50	200.17	6651.1	0.56	1.73	2.11
55	359.53	7836.7	1.06	2.76	3.12
60	774.12	10151.1	1.90	2.93	3.36
65	852.79	8471.4	3.02	4.58	4.82
70	1557.0	10475.0	3.89	5.35	5.48
75	2371.52	12802.5	5.17	*****	6.77
80	1699.30	9806.66	6.18	*****	7.17

tando as matrizes de demandas que são irrelevantes para o PML, deixando apenas as matrizes de custo. As instâncias baseadas no PQA foram resolvidas pelo CPLEX utilizando-se a formulação M3_PML. Isso se deve ao fato de a formulação M3_PML ter se mostrado eficiente para resolver as instâncias baseadas em Fischetti *et al.* [FLM93].

A estratégia nesta seção, especialmente para instâncias com um número grande de vértices, baseou-se na combinação de limites inferiores justos e na computação de limites superiores rápidos e eficientes que são providos pela heurística GRASP apresentada na seção (2.7). Esses resultados são apresentados na Tabela (2.5). As instâncias testadas estão escritas como $m- < nome-da-instancia-original-QAPLIB > ***$, onde $***$ é o número de vértices. Foi apresentado também, quando possível, o tempo de computação do CPLEX no cálculo do ótimo inteiro. Nas instâncias marcadas por (**), não foi possível, usando o Método da Barreira de Newton, calcular o limite inferior. Isso ocorreu pela falta de memória do computador pessoal utilizado. Nesses casos, foi utilizado o método Dual Simplex para calcular o limite de programação linear dessas instâncias. Isso explica o alto custo computacional para calcular apenas o limite inferior dessas instâncias.

2.9 Análise dos Resultados

As formulações F1_PML, F2_PML, F3_PML e F4_PML são o eixo de ligação entre os quatro problemas que são apresentados neste trabalho. Todos os problemas desta tese

Tabela 2.5: Resultados - Instâncias QAPLIB Modificadas

Instância	Tempo (s) CPLEX	Limite Inferior	Ótimo	Lim Inf /Ótimo (%)	Tempo (s) CPLEX LP	Solução GRASP	Iterações GRASP	Gap (%) LS /Ótimo
m-had12	0.12	82.66	88	6.64	0.02	88	2	0.00
m-had12	0.41	109.98	125	12.10	0.03	125	1	0.00
m-had16	0.24	140.71	146	3.62	0.06	146	1	0.00
m-had20	4.69	215.12	232	7.28	0.13	232	13	0.00
m-rou12	0.22	947.75	1099	13.76	0.03	1099	23	0.00
m-rou15	1.23	1068.5	1274	16.13	0.07	1274	13	0.00
m-scr12	0.13	78	78	0.00	0.02	78	3	0.00
m-scr15	0.50	122	122	0.00	0.06	122	11	0.00
m-scr20	1.78	210	210	0.00	0.06	210	13	0.00
m-nug12	0.08	78	78	0.00	0.02	78	3	0.00
m-nug14	0.12	105	105	0.00	0.03	105	0	0.00
m-nug15	0.33	121	121	0.00	0.04	121	1	0.00
m-nug16a	0.55	136.28	138	1.25	0.05	138	1	0.00
m-nug17	0.51	154	154	0.00	0.06	154	1	0.00
m-nug18	0.73	171	171	0.00	0.08	171	4	0.00
m-nug20	0.98	210	210	0.00	0.13	210	0	0.00
m-nug21	1.57	232	232	0.00	0.15	232	142	0.00
m-nug22	0.33	253	253	0.00	0.22	253	0	0.00
m-nug24	4.44	300	300	0.00	0.25	300	0	0.00
m-nug25	6.55	326	326	0.00	0.29	326	4	0.00
m-nug27	4.60	379	379	0.00	0.43	379	265	0.00
m-nug30	28.55	465	465	0.00	0.63	465	263	0.00
m-sko42	1637.33	903	903	0.00	2.92	903	0	0.00
m-sko49	2763.15	1226	1226	0.00	6.61	1230	0	0.33
m-sko56	18343.65	1596	1596	0.00	10.57	1602	0	0.37
m-sko64	53128.41	2080	2080	0.00	18.80	2080	0	0.00
m-sko72	*****	2628	2628	0.00	36.34	2628	0	0.00
m-sko81	*****	3322	*****	0.18**	63.62	3328	0	0.18**
m-sko90	*****	4095	*****	0.19**	*****	4103	0	0.19**
m-sko100a	*****	5050	5050	0.00	200.10	5050	0	0.00
m-tho30	26.24	465	465	0.00	0.67	465	398	0.00
m-tho40	271.55	820	820	0.00	2.42	820	45	0.00
m-tho150	*****	11325	11325	0.00	20960.02*	11325	0	0.00
m-will50	*****	1275	*****	0.16**	6.29	1277	5097	0.16**

terão formulações derivadas da formulação F3_PML. A F1_PML é a formulação mais simples, mas é a versão multiproduto dela, F3_PML, que é a base para algumas formulações apresentadas nos capítulos posteriores. A F1_PML é uma formulação que usa da idéia simples de fazer um fluxo percorrer por uma rede, garantindo a conservação desse fluxo e calculando os custos no decorrer desse caminho. O fraco desempenho tanto nos *gaps* de relaxação linear quanto no tempo de computação está associado tanto ao ínfimo número de variáveis e restrições - na formulação F1_PML são usadas apenas restrições para garantir a conservação do fluxo e inibir subciclos ilegais - como também está associado à característica uniproduto da formulação. A formulação F2_PML também é de extrema importância para este trabalho. O acréscimo das variáveis p e das restrições (2.18)-(2.22) na formulação F1_PML, melhorando tanto o tempo para calcular o ótimo inteiro como diminuindo o *gap* de relaxação linear, será usado nos capítulos posteriores impactando de forma muito positiva os resultados computacionais. A formulação F3_PML, por ser multiproduto, mostrou-se mais forte que as anteriores, mas não necessariamente mais rápida. A formulação F4_PML, também multiproduto, se mostrou mais rápida que a F3_PML, mas nem sempre mais rápida que a F2_PML.

O primeiro modelo baseado no PQA, (2.60)-(2.65), obteve resultados muito expressivos. Seus limites de relaxação linear não ultrapassaram, em média, 2% do ótimo. Com ele foram resolvidas, até a otimalidade, instâncias de tamanho 30. Quando foi utilizado o modelo M2_PML, isto é, o modelo (2.66)-(2.74), conseguiu-se resolver, até a otimalidade, instâncias de até 60 vértices. Apesar de o algoritmo de Fischetti *et al.* [FLM93] também apresentar resultados de 60 vértices com um tempo computacional menor, utilizando uma máquina mais antiga, os modelos apresentados têm como força seus limites inferiores. O *gap* de relaxação linear desses modelos foi extremamente baixo, na média menos de 2%, mesmo para instâncias de tamanho 60. Esse *gap* mostra que o modelo (2.66)-(2.74) e, principalmente, o modelo (2.60)-(2.65) são bem fortes e abrem caminhos para novas técnicas que se aproveitem desses limites inferiores para calcular soluções ótimas, ou mesmo soluções heurísticas, mas com uma boa garantia de limites. Além disso, até onde se sabe, não foram encontrados para o PML limites inferiores tão justos quantos os limites calculados pela formulação M1_PML.

Apesar de os modelos M2_PML e M3_PML terem o mesmo limite inferior que o modelo apresentado por Bianco, Mingozzi e Ricciardelli [BMR93], M4_PML, uma contribuição deste trabalho é apresentar a equivalência, em relação aos limites de programação linear, entre esses modelos e a *Formulação_Relaxada*. Esta última, que deriva do modelo M1_PML,

é baseada no Problema Quadrático de Atribuição.

Os modelos que apresentaram melhores resultados em termos de tempo de computação foram os modelos M3_PML e M4_PML. Esses dois modelos são baseados no Problema do Caminho Mínimo e geraram os mesmos limites de programação linear para todas as instâncias testadas. Esses limites, aliás, são os mesmos da *Formulação_Relaxada* e, conseqüentemente, são os mesmos da formulação M2_PML. Apesar da equivalência em termos de limites de programação linear entre as formulações M3_PML e M4_PML, elas foram escritas de forma diferente. A primeira se baseou no grafo multinível G' apresentado na seção (2.6). A segunda se baseou no grafo normal G. Isso explica a variação das médias dos tempos de computação para cada conjunto de 10 instâncias analisado. Apesar de essas médias serem, em geral, muito próximas, elas não revelam a discrepância dos tempos de computação entre as formulações e entre os tipos de configurações do resolvedor CPLEX. Por exemplo: a última instância testada com 45 vértices teve um tempo de computação de 273.52 segundos utilizando a formulação M3_PML com os parâmetros padrões para o CPLEX; já a mesma instância usando os mesmos parâmetros do CPLEX mas utilizando a formulação M4_PML levou 696.68 para ser resolvida. A mesma instância, mas com os parâmetros do CPLEX modificados, levou 222.8 segundos para ser resolvida usando a formulação M3_PML e 109.64 utilizando a formulação M4_PML.

Com os modelos M3_PML e M4_PML, foi possível encontrar o ótimo inteiro para instâncias assimétricas de até 80 vértices. Pelo que se sabe, até hoje ainda não tinham sido resolvidas instâncias assimétricas maiores que 55 vértices. Para as instâncias baseadas no PQA o resultado foi ainda mais impressionante. Para elas foi possível provar a otimalidade para instâncias de até 150 vértices. Isso foi possível devido à força do limite inferior calculado pela relaxação linear utilizando a formulação M3_PML e, também, à força do limite superior, utilizando o algoritmo GRASP. Na instância $m - tho150$, esses limites se igualaram provando a otimalidade da mesma.

O algoritmo GRASP também se mostrou eficiente. Na maioria das instâncias o limite superior encontrado foi idêntico à solução ótima. Por exemplo: até grafos de 45 vértices foram testadas 90 instâncias baseadas no trabalho de Fischetti *et al.* [FLM93]. Dentre essas 90 instâncias somente em três delas não foi encontrado o ótimo inteiro pela heurística. Mesmo para as instâncias maiores, entre 50 e 60 vértices, o *gap* entre as soluções ótimas e os valores encontrados pelo algoritmo estiveram, em média, próximos dos 1.5%. O algoritmo GRASP foi especialmente eficiente nas instâncias baseadas na $QAPLIB$. Para essas instâncias, na maioria dos casos, o algoritmo GRASP encontrou a solução ótima e,

caso não tenha encontrado, em nenhum caso, o *gap* entre a solução encontrada e o ótimo foi superior a 0.37%. Duas instâncias devem ser observadas: nas instâncias *sko81* e *sko90*, não foram encontradas as soluções ótimas. Apesar disso, é sabido que a solução encontrada, para essas instâncias, não está mais de 0.18% e 0.19%, respectivamente, da solução ótima.

O papel da heurística GRASP não foi somente encontrar limites superiores justos para o PML. Quando se utiliza a solução encontrada pelo método GRASP como primeira solução para resolver o problema, via CPLEX, conseguiram-se, em média, resultados exatos mais rápidos. Isso acontece quando essa solução foi melhor que a primeira solução que o CPLEX calcularia caso não tivesse recebido esse limite superior. Por exemplo: na instância P60-2, o CPLEX gerou um *gap* inicial entre a primeira solução inteira e o valor do limite de programação linear de 68.28%. Nesse caso, ele levou 17790.2 segundos para provar a otimalidade dessa instância; já quando o CPLEX se utilizou do limite superior pré-computado pelo GRASP, ele levou somente 6872.25 segundos para provar a otimalidade dessa instância. Nesse caso, o CPLEX só precisou "fechar"um *gap* de 4.72%. O ganho, nesse caso, foi de 61.37% em relação ao tempo total. Como pode-se ver na Tabela (2.2), sempre a média dos tempos de solução do CPLEX, quando esse utilizou como limite superior o valor encontrado pelo GRASP, é menor. Apesar disso, nem sempre é mais vantajoso utilizar o limite superior pré-computado pelo algoritmo GRASP. Em alguns casos, a primeira solução inteira que o CPLEX calcula é melhor que a solução passada pelo GRASP. Nesses casos, o uso desse limite superior pré-computado não é vantajoso. Pela Tabela (2.2), notou-se que, em geral, à medida que os problemas crescem, o uso de um limite superior pré-computado é mais vantajoso. A exceção está nas instâncias de 60 vértices, que obtiveram um ganho de apenas 7.61% utilizando o limite superior pré-computado pelo GRASP. Isso se deve principalmente à instância P60-5 que, utilizando a solução do GRASP, teve um tempo de computação 62% pior. O interessante nesse fato é que essa instância foi uma das duas instâncias desse conjunto em que a solução passada para o CPLEX estava a mais de 3% do ótimo. Em alguns casos, por exemplo, mesmo quando a solução passada para o CPLEX é a ótima, o CPLEX gasta mais tempo do que quando utiliza o método original. Provavelmente essa diferença de tempo está associada à execução, em paralelo, de processos da máquina, interferindo no desempenho do algoritmo. Para mais detalhes sobre as instâncias, vide o apêndice A.

2.10 Conclusão

Neste capítulo foram tratadas técnicas para resolver eficientemente instâncias do Problema de Mínima Latência. Para isso, foi mostrada a relação desse problema com o Problema Quadrático de Atribuição e com o Problema de Caminho Mínimo. Neste capítulo foram apresentados três modelos matemáticos baseados diretamente no PQA e um baseado no Problema do Caminho Mínimo. Também foram apresentadas duas formulações de fluxos, uma uniproduto e outra multiproduto. Mostrou-se também o impacto de acrescentar-se um novo conjunto de variáveis, com algumas restrições adicionais, para essas formulações. Essa idéia será usada no decorrer deste trabalho.

O modelo M1_PML conseguiu resultados exatos somente para instâncias de até 30 vértices, mas se mostrou, pelo que se sabe, o modelo com limites de programação linear mais fortes já apresentados para o PML. A *Formulação Relaxada* e o modelo M2_PML conseguiram resultados ótimos para instância de até 60 vértices. Esses dois modelos se mostraram equivalentes em relação aos limites de programação linear. Com o modelo M3_PML e M4_PML conseguiu-se resultados exatos para instâncias assimétricas de até 80 vértices. Até hoje, pelo que se sabe, não tinham sido apresentados resultados exatos para instâncias assimétricas do PML maiores que 55.

Além dos modelos exatos foi apresentada uma heurística especializada para resolver o Problema de Mínima Latência. Essa heurística foi baseada na meta-heurística GRASP e apresentou resultados muito expressivos. Na maioria das instâncias a heurística encontrou o ótimo inteiro. Mesmo para as instâncias em que o ótimo inteiro não foi encontrado, o *gap* entre a solução encontrada e o ótimo nunca ultrapassou, em média, 2%. Esse resultado é mais expressivo que o resultado apresentado por Bianco *et al.* [BMR93], que apresentou resultados em média 4.2% do ótimo para instâncias de 60 vértices.

Outra vantagem da heurística foi utilizar os resultados encontrados pela mesma como primeiro limite superior para o resolvedor CPLEX. Como foi mostrado na Tabela (2.2), foram obtidos ganhos médios de até 30% em relação ao método tradicional de utilizar o CPLEX sem um limite superior calculado para a instância. Foi mostrado também que esse ganho aumenta à medida que as instâncias crescem de tamanho. Outra constatação é que essa técnica será mais eficiente à medida que os limites superiores passados para o CPLEX forem mais próximos do ótimo inteiro. Para algumas instâncias, aquelas que o limite passado para o CPLEX é maior que o limite que o resolvedor calcularia, caso não fosse passado para ele nenhum limite, é melhor utilizar o método padrão. A heurística também se mostrou eficiente para as instâncias baseadas na biblioteca $QAPLIB$. Nessas

instâncias, o *gap* entre a solução ótima e o valor encontrado pelo GRASP não ultrapassou 0.37%, mesmo considerando instâncias de 150 vértices.

Capítulo 3

Problema de Um Veículo de Entrega

3.1 Introdução

O Problema de Um Veículo de Entrega (PUVE), do inglês *Single Vehicle Delivery Problem*, é uma generalização do Problema de Mínima Latência (PML) apresentado no capítulo anterior. No PUVE, assim como no Problema de Mínima Latência, o objetivo é encontrar o menor circuito Hamiltoniano que minimize a soma dos tempos de espera de todos os clientes. A diferença entre o PML e o PUVE é que este último apresenta demandas heterogêneas em cada vértice e o tempo de espera de um dado cliente é multiplicado pelo número de produtos que são entregues a esse cliente. Se todos os clientes possuírem demandas homogêneas, o PUVE se transforma no PML.

Este capítulo é dividido da seguinte forma: a seção (3.2) formaliza o Problema de Um Veículo de Entrega; a seção (3.3) mostra um exemplo didático diferenciando o PML do PUVE; a seção (3.4) apresenta dois conjuntos de formulações utilizadas para lidar com o PUVE; a primeira baseada na formulação de fluxos apresentada no capítulo anterior, e a segunda, baseada na formulação de Bianco *et al.* [BM$^+$89]; a seção (3.5) mostra um algoritmo GRASP especializado para o PUVE; a seção (3.6) apresenta os resultados computacionais utilizando as formulações propostas, os limites superiores gerados pela heurística GRASP e o efeito de se utilizar esse limite superior como solução inicial viável para o resolvedor CPLEX; a seção (3.7) analisa os resultados encontrados; e, finalmente, a seção (3.8) conclui o capítulo.

3.2 Problema de Um Veículo de Entrega

O Problema de Um Veículo de Entrega foi estudado por Bianco, Mingozzi, Ricciardelli e Spadoni [BM+89] e se caracteriza por um único veículo que precisa entregar d_k unidades de um produto homogêneo em cada vértice k de um grafo $G = (V, A)$. O veículo parte da origem 1 e precisa, depois de percorrer todos os vértices, entregar $d_1 = 1$ unidade do produto de volta à origem. Essa unidade de produto reflete a necessidade de o operador logístico retornar ao ponto de partida com o seu veículo.

O objetivo nesse problema é minimizar a média das latências de cada produto entregue para todos os clientes. Em outras palavras, pode-se dizer que o PUVE é um PML com demandas heterogêneas. No PUVE, a ordem de visitação e o número de produtos que são entregues em cada vértice são relevantes para o cálculo da rota final. De uma forma geral, pode-se dizer que, se o PML é *orientado aos clientes*, o PUVE também é *orientado à quantidade de produtos entregues*. O PUVE é computacionalmente mais difícil de resolver que o PML, uma vez que o PUVE possui diferentes demandas em cada vértice e, freqüentemente, é mais interessante visitar primeiramente um vértice com uma demanda mais alta do que um que esteja mais próximo.

Apesar da óbvia similiaridade com o Problema do Caixeiro Viajante (PCV) clássico, tanto o PML como o PUVE aparentam ser mais difíceis de serem resolvidos computacionalmente que o PCV [GK98]. No PUVE e no PML, o objetivo é minimizar a latência total de todos os produtos entregues, enquanto, no PCV, o objetivo é minimizar o tempo de viagem de um único operador logístico. Esses problemas são mais complexos porque ambos incorporam os objetivos conflitantes dos consumidores, ao invés de se preocupar somente com o tempo de viagem do operador logístico, como no PCV original.

Bianco, Mingozzi, Ricciardelli e Spadoni [BM+89] propuseram uma formulação para o PUVE, mas nenhuma solução ótima foi apresentada com essa formulação. Eles também propuseram uma heurística baseada no algoritmo de programação dinâmica que conseguiu resultados ótimos para instâncias de 25 vértices. Apesar disso, para instâncias maiores a heurística não aparentou ser muito promissora. Por exemplo: para instâncias com 45 vértices os autores estiveram, em média, a 5.9% entre os limites superiores e inferiores e, para instâncias de 50 vértices, estiveram, em média, a 9.1%.

Neste capítulo, duas formulações baseadas em fluxos e três baseadas no modelo de Bianco *et al.* [BM+89] são apresentadas para lidar com instâncias do PUVE. Também é mostrada a importância de gerar bons limites superiores para que os resolvedores comerciais, utilizando esse limite como primeiro limite superior, possam diminuir o tempo de

computação. Esses limites superiores são providos por uma heurística GRASP especializada para o PUVE.

3.3 Diferença entre o PML e o PUVE

Para entender a diferença entre o Problema de Mínima Latência e o Problema de Um Veículo de Entrega será mostrado um exemplo. Para um grafo $G = (V, A)$ com seis vértices e cada arco com um custo c_{ij}, são resolvidos o PML e o PUVE. A Figura (3.1) mostra o grafo G. Esse grafo é o mesmo apresentado no Capítulo (2), quando se tratou da diferença entre o PCV e o PML. Na Figura (3.1), o número que aparece externamente ao lado de cada vértice indica a demanda de unidades do produto requerida no vértice.

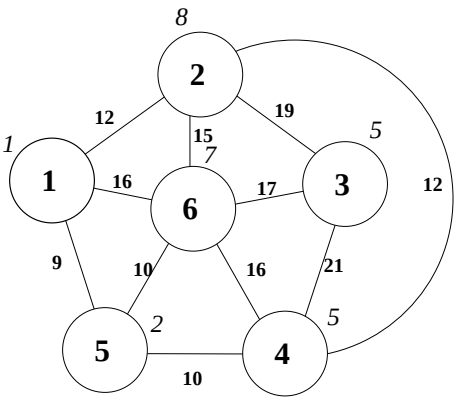


Figura 3.1: Grafo Base.

A Figura (3.2) mostra a solução ótima para o PML utilizando como base o mesmo grafo G. A solução ótima inteira para o PML no grafo G é 259 (9 + (9+10) + (9+10+12) + (9+10+12+19)+ (9+10+12+19+17) + (9+10+12+19+17+16)). Outra forma de se calcular o valor da função objetivo é (9*6 +10*5 + 12*4 + 19*3+17*2+16).

A Figura (3.3) mostra a solução ótima para o PUVE utilizando como base o mesmo grafo G. Essa solução é diretamente influenciada pelas demandas d_k de cada vértice. Isso já não acontece no PCV e nem no PML, onde essas demandas não têm nenhuma relevância. A solução ótima inteira para o PUVE com $\sum d_i = 28$ é 1064 $((12 * 28) + (15 * 20) + (17 * 13) + (21 * 8) + (10 * 3) + (9 * 1))$. Observe que os vértices com maiores demandas foram visitados primeiramente na solução ótima do PUVE, como no caso dos vértices 2 e 6 da Figura (3.3).

Também nesse caso não é possível comparar os valores da solução ótima para esses dois

exemplos, pois cada um possui uma contabilização de custos diferente, impactando no valor da função objetivo. O que cabe verificar é que a rota ótima para cada problema é diferente. Enquanto a solução ótima para o PML é a menor soma de todos os tempos de espera para todos os vértices, a solução ótima do PUVE é a menor somatória dos tempos de espera de cada unidade do produto que deve ser entregue em cada cliente. Variações nas demandas de cada cliente impactam na solução final. De uma forma geral pode-se dizer que o PML é *orientado à satisfação dos clientes* que recebem os produtos, já no PUVE essa satisfação é ponderada pelo número de produtos entregues em cada um desses clientes.

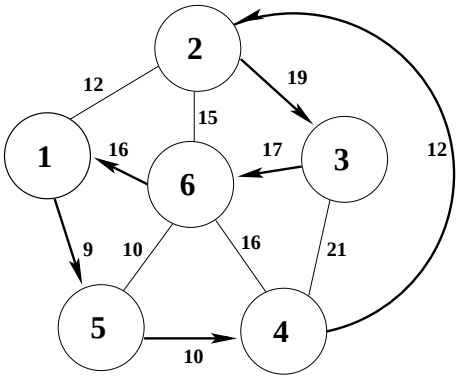


Figura 3.2: Solução PML.

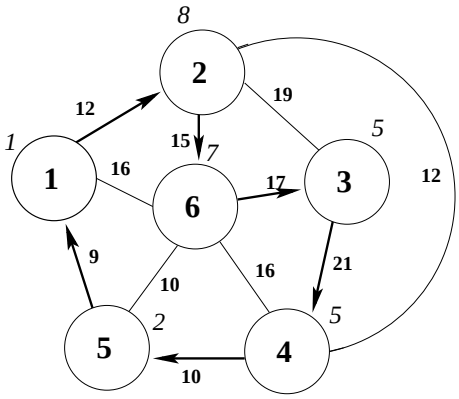


Figura 3.3: Solução PUVE.

3.4 Formulações para o Problema de Um Veículo de Entrega

Para o Problema de Um Veículo de Entrega são apresentados, neste capítulo, dois conjuntos de formulações de programação linear inteira mista. O primeiro conjunto é baseado na formulação de fluxos multiproduto para o Problema de Mínima Latência apresentada no Capítulo 2, (2.23)-(2.34). Para esse conjunto são propostas duas formulações, a última com variáveis e restrições associando a ordem de visitação de um arco com o fluxo global que passa por esse arco, da mesma forma que no PML. O segundo conjunto é baseado no modelo de Bianco *et al.* [BM$^+$89]. Para esse modelo também são acrescidas restrições adicionais.

3.4.1 Formulação de Fluxos para o Problema de Um Veículo de Entrega

Para o PML foram apresentadas duas formulações de fluxos uniproduto e duas multiproduto. Para o PUVE, não serão apresentadas as formulações uniproduto. Isso se deve à impossibilidade de desenvolver uma formulação uniproduto que utilize restrições que associem a ordem de visitação com o fluxo em cada arco. No PML, como cada vértice demanda uma unidade de um produto é possível conhecer o fluxo global em um arco apenas sabendo-se o número de vértices já visitados. No PUVE, devido à heterogeneidade das demandas, é impossível associar, com uma formulação uniproduto, o fluxo em um arco e o número de vértices já visitados.

Utilizando como base a formulação de fluxos (2.23)-(2.34) apresentada no Capítulo 2, é possível criar uma formulação de fluxos multiproduto para o Problema de Um Veículo de Entrega. Essa formulação, chamada de F1_PUVE, possui o seguinte conjunto de variáveis:

$$x_{ij} = \begin{cases} 1 & \text{se o veículo atravessa o arco } (i,j) \\ 0 & \text{caso contrário.} \end{cases}$$

f_{ijk}: fração da demanda total do produto k transportada no arco (i,j) destinado ao vértice de demanda k.

e o seguinte conjunto de parâmetros:

c_{ij}: custo pago pelo operador logístico para atravessar o arco (i,j).

d_k: demanda do vértice k.

$$\text{minimize} \sum_{i,j,k \in V | i \neq j} c_{ij} d_k f_{ijk} \tag{3.1}$$

sujeito a

$$\sum_{i \in V | i \neq j} x_{ij} = \quad 1 \qquad \forall j \in V \tag{3.2}$$

$$\sum_{j \in V | i \neq j} x_{ij} = \quad 1 \qquad \forall i \in V \tag{3.3}$$

$$f_{ijk} \leq \quad x_{ij} \qquad \forall i, j, k \in V, i \neq j \tag{3.4}$$

$$\sum_{j \in V | j \neq 1} f_{1j1} = \quad 1 \tag{3.5}$$

$$\sum_{j \in V | j \neq 1} (f_{1jk} - f_{j1k}) = \quad 1 \qquad \forall k \in V, k \neq 1 \tag{3.6}$$

$$\sum_{i \in V | i \neq 1} f_{i11} = \quad 1 \tag{3.7}$$

$$\sum_{i \in V | i \neq k} (f_{ikk} - f_{kik}) = \quad 1 \qquad \forall k \in V, k \neq 1 \tag{3.8}$$

$$\sum_{i \in V | i \neq j, i \neq k} (f_{jik} - f_{ijk}) = \quad 0 \qquad \forall k, j \in V, j \neq k, j \neq 1 \tag{3.9}$$

$$\sum_{i,j,k \in V | i \neq j} f_{ijk} = \quad \frac{n(n+1)}{2} \tag{3.10}$$

$$f_{ijk} \geq \quad 0 \qquad \forall i, j, k \in V, i \neq j \tag{3.11}$$

$$x_{ij} \in \quad \{0, 1\} \qquad \forall i, j \in V, i \neq j \tag{3.12}$$

onde $n =\mid V \mid$.

A função objetivo (3.1) soma os custos para todos os arcos da rede. Esse custo está associado à transferência, pelos arcos do caminho, de todos os produtos, do vértice origem ao específico vértice de destino.

As equações (3.2) e (3.3) são as restrições de atribuição introduzidas em 1954 por Dantzig, Fulkerson e Johnson [DFJ54]. Elas garantem que existe somente um arco saindo e um arco chegando a cada vértice. As restrições (3.4) acoplam as variáveis x e as f.

As equações (3.5)-(3.9) garantem a conservação de fluxo na rede. Esse tipo de restrições de fluxo foi baseado no trabalho de Garvin *et al.* [GCJS57]. A equação redundante (3.10) assegura que o número de fluxos unitários que percorrem todos os arcos é expresso por $\frac{n(n+1)}{2}$, onde $n =\mid V \mid$ (3.10).

O fato que todas as variáveis x_{ij} são binárias é assegurado pelas restrições (3.12).

Nota-se que a formulação (3.1)-(3.12) só difere da formulação (2.23)-(2.34) na função

objetivo (3.1). Em (3.1), o custo de transportar uma fração do produto destinado ao vértice k é multiplicado pela demanda d_k desse vértice. Em (2.23) essa multiplicação também existe, mas como a demanda de cada vértice é unitária, é possível retirar o parâmetro d_k da função objetivo (2.23). Na formulação (3.1)-(3.12) as demandas estão normalizadas.

3.4.1.1 Uma Formulação Estendida para o Problema de Um Veículo de Entrega

A formulação F1_PUVE é suficiente para descrever o PUVE. Por outro lado, inspirando-se na associação entre o Problema de Mínima Latência e o Problema Quadrático de Atribuição (PQA) [KB57], já apresentada no capítulo anterior, é possível criar variáveis e restrições adicionais com o intuito de melhorar o *gap* de relaxação linear e também o tempo computacional da formulação F1_PUVE. A associação entre o PML e o PQA é derivada da associação do PCV com o PQA, pois este último pode ser usado para resolver instâncias do PCV. Nesse caso, as variáveis binárias do PQA são associadas à ordem de visitação dos vértices no PCV. As melhoras citadas anteriormente são esperadas devido às melhoras nos tempos e nos *gaps* ocorridas quando restrições que associam o fluxo nos arcos com a ordem de visitação desses arcos foram introduzidas na formulação de fluxos do Problema de Mínima Latência, vide seção (2.4).

Quando se usa o PQA para resolver o PCV, é importante lembrar que as restrições de atribuição do PQA não são escritas no espaço dos arcos, isto é, não significam que um dado arco é percorrido somente uma vez em alguma direção. Elas asseguram que um dado vértice é visitado em apenas uma ordem de visitação e que uma dada ordem de visitação é usada por apenas um vértice somente uma vez. Para usar esse tipo de restrição de acoplamento com o modelo (3.1)-(3.12), propõe-se o acréscimo de um novo conjunto de variáveis p_{ij}, definindo a ordem j de visitação do vértice i. Normalmente, para esse novo conjunto de variáveis, deveria ser criado um novo conjunto de ordens de visitação. Isso não será necessário, pois existem tantas ordens de visitação quantos os vértices de V. Essa observação é a chave para escrever as restrições de atribuição como:

$$\sum_{i \in V} p_{ij} = 1 \qquad \forall j \in V \tag{3.13}$$

$$\sum_{j \in V} p_{ij} = 1 \qquad \forall i \in V \tag{3.14}$$

$$p_{11} = 1 \tag{3.15}$$

$$p_{ij} \geq 0 \qquad \forall i, j \in V \tag{3.16}$$

$$\sum_{k,j \in V | i \neq j} f_{ijk} = \sum_{l=1}^{n} (n - l + 1) p_{li} \qquad \forall i \in V \tag{3.17}$$

$$\sum_{i,j \in V | i \neq j} f_{ijk} = \sum_{l=1}^{n} (l - 1) p_{lk} \qquad \forall k \in V, k \neq 1 \tag{3.18}$$

onde $n = \mid V \mid$.

As restrições (3.13) e (3.14) também são restrições de atribuição, da mesma forma que as restrições (3.2) e (3.3). Entretanto, essas últimas garantem que um dado vértice somente é visitado em uma dada ordem de visitação e que em uma ordem de visitação somente um vértice é visitado. Note que enquanto as restrições (3.2) e (3.3) não são definidas para i e j iguais, o mesmo não acontece para as restrições (3.13) e(3.14), já que um vértice n pode ser visitado na *n-ésima* posição. A restrição (3.15) garante que o primeiro vértice a ser utilizado é o vértice 1, isto é, o vértice origem. As restrições de acoplamento (3.16) e (3.17) são inequações válidas que conectam as variáveis f com as variáveis p. Como será visto posteriormente, essas restrições impactam tanto no tempo de solução do modelo como no *gap* de relaxação linear. A formulação F1_PUVE, acrescida das restrições (3.13)-(3.18), é chamada de formulação F2_PUVE.

3.4.2 Formulação de Bianco, Mingozzi, Ricciardelli e Spadoni

Em 1989, Bianco, Mingozzi, Ricciardelli e Spadoni [BM$^+$89] apresentaram uma formulação para o PUVE. Essa formulação possui o seguinte conjunto de variáveis:

$$x_{ij}^{q} = \begin{cases} 1 & \text{se o veículo atravessa o arco } (i,j) \text{ carregado com uma carga } q \\ 0 & \text{caso contrário.} \end{cases}$$

e o seguinte conjunto de parâmetros:

c_{ij}: custo fixo pago pelo operador logístico para atravessar o arco (i, j).

d_q: demanda do vértice q.

Q: soma das demandas de todos os vértices.

Para esse problema, Bianco *et al.* [BM+89] apresentaram a seguinte formulação de programação linear inteira mista:

$$\text{minimize} \sum_{q=d_1}^{Q} q \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}^{q} \tag{3.19}$$

sujeito a

$$\sum_{j=1}^{n} x_{ij}^{q-d_i} - \sum_{l=1}^{n} x_{li}^{q} = 0, \quad i = 2..n, \quad q = d_i + d_1..Q \tag{3.20}$$

$$\sum_{j=2}^{n} x_{1j}^{Q} = 1 \tag{3.21}$$

$$\sum_{i=2}^{n} x_{i1}^{d_1} = 1 \tag{3.22}$$

$$\sum_{q=d_j+d_1}^{Q} \sum_{i=1, i\neq j}^{n} x_{ij}^{q} = 1, \quad j = 2..n \tag{3.23}$$

$$x_{ij}^{q} \in \{0, 1\}, q = d_1..Q, \quad \forall (i, j) \in A \tag{3.24}$$

A função objetivo (3.19) soma os custos para todos os arcos e para todas as cargas possíveis. O conjunto de restrições (3.20) garante a conservação do fluxo para todos os vértices, salvo a origem. A restrição (3.21) assegura que sai um fluxo de carga Q da origem. A restrição (3.22) assegura que chega um fluxo de carga d_i na origem. As restrições (3.23) garantem que somente existe um arco incidente a cada vértice, salvo a origem. Elas também garantem que, nesse arco, somente uma carga pode ser transportada. As restrições (3.24) garantem que as variáveis x_{ij}^{q} são binárias.

Após serem executados alguns testes com o modelo (3.19)-(3.24), verificou-se que em algumas instâncias a solução ótima gerou mais de um ciclo partindo da origem. Após o modelo (3.19)-(3.24) ser analisado, chegou-se à conclusão que essa falha ocorre porque a restrição (3.21) permite que somente um arco deixe a origem com demanda Q, mas algumas vezes outro arco pode deixar a origem com uma demanda diferente de Q, criando mais de um ciclo. Para corrigir esse problema são propostas duas alternativas:

Na primeira, é possível mudar a restrição (3.20), indexando o índice i de 1 até n da seguinte forma:

$$\sum_{j=1}^{n} x_{ij}^{q-q_i} \; - \; \sum_{l=1}^{n} x_{li}^{q} = 0 \quad i = 1, ..., n \; , q = d_i + d_1, ..., Q \tag{3.25}$$

Outra alternativa é adicionar a seguinte restrição ao modelo original:

$$\sum_{j=2}^{n} x_{1j}^{q} = 1, q = 1, .., Q \tag{3.26}$$

3.5 Uma Meta-heurística GRASP para o Problema de Um Veículo de Entrega

Como foi apresentado no capítulo sobre o Problema de Mínima Latência, não existem muitas heurísticas eficientes para tratar variantes do Problema do Caixeiro Viajante com Dependência de Tempo (PCVDT). Para o PUVE, apenas uma heurística foi encontrada. Como mencionado anteriormente, a heurística apresentada por Bianco *et al.*. [BM$^+$89] encontrou soluções com *gaps* de 9.1% para instâncias de 50 vértices. Nesta seção será apresentada uma heurística eficiente para tratar instâncias do PUVE.

O algoritmo escolhido é uma variação da heurística GRASP apresentada na seção (2.7). O algoritmo, na verdade, possui a mesma Fase de Construção, as mesmas quatro Buscas Locais e o mesmo método de *Path-relinking* apresentados no capítulo anterior, vide seção (2.7). A diferença entre a heurística apresentada no capítulo anterior, definida para o PML, e a heurística definida neste capítulo, que é para o PUVE, está na contabilização dos custos.

3.6 Resultados Computacionais

Os testes foram realizados em uma máquina com um processador Pentium IV de 3.0Ghz com 1.0 Gbyte de memória RAM. O sistema operacional utilizado foi o Linux. Foi utilizado o resolvedor CPLEX 9.1.3 da ILOG.

As instâncias para o PUVE são geradas de acordo com a proposta de Fischetti, Laporte e Martelo [FLM93], da mesma forma que foram criadas as instâncias do capítulo sobre o Problema de Mínima Latência. Para o PUVE, também são geradas 10 instâncias

pseudo-aleatórias para cada classe (tamanho) de problema. Além disso, todas as instâncias são assimétricas e os grafos que representam essas instâncias são todos completos. A única diferença entre as instâncias do PML e as do PUVE é que nessas últimas todos os vértices, salvo a origem, possuem demandas heterogêneas e aleatórias variando entre 1 e o valor de $Max\_Demanda$. No PUVE, a demanda da origem é sempre unitária e representa o operador logístico que precisa retornar ao ponto de partida. Para cada classe de instâncias (dez), duas têm $Max\_Demanda = 2$, duas têm $Max\_Demanda = 3$, duas têm $Max\_Demanda = 4$, duas têm $Max\_Demanda = 5$ e duas têm $Max\_Demanda = 6$.

Com relação aos modelos são apresentados dois conjuntos de resultados. O primeiro é relativo às formulações F1_PUVE e F2_PUVE, isto é, as formulações de fluxo multiproduto apresentadas na seção (3.4.1). Para esses testes foi utilizado o resolvedor CPLEX com os parâmetros padrões, apenas alterando-se o algoritmo de cálculo do limite de programação linear para o Método da Barreira de Newton.

O segundo conjunto está relacionado às formulações baseadas no trabalho de Bianco *et al.* [BM$^+$89]. Para esse caso são apresentadas mais quatro abordagens utilizando o resolvedor CPLEX, de forma monolítica, para resolver as mesmas instâncias do PUVE. A primeira abordagem, chamada de M1_PUVE, é aquela que utiliza a formulação apresentada por Bianco *et al.* [BM$^+$89] apenas substituindo a restrição (3.20) pela restrição (3.25). Na segunda abordagem, chamada de M2_PUVE, apenas é acrescentada a restrição (3.26) ao modelo original (3.19)-(3.24). A terceira abordagem, chamada de M3_PUVE, é a junção das abordagens M1_PUVE e M2_PUVE. Isto é, na abordagem M3_PUVE a restrição (3.20) é substituída pela (3.25) e, além disso, é acrescida a restrição (3.26), assim como na M2_PUVE. A M4_PUVE utiliza a formulação apresentada na abordagem M3_PUVE utilizando como primeiro limite superior do CPLEX a solução encontrada pela heurística GRASP apresentada na seção (3.5). Essa estratégia foi utilizada devido ao sucesso da mesma para o Problema de Mínima Latência do capítulo anterior.

A Tabela (3.1) relativa às formulações F1_PUVE e F2_PUVE apresenta sempre a média dos tempos e dos *gaps* para cada conjunto de 10 instâncias de mesmo tamanho. Nessa tabela tem-se: o tamanho dos problemas, campo *Tamanho*; a média dos *gaps* de relaxação linear, campos *Gap RL F1_PUVE (%)* e *Gap RL F2_PUVE (%)*; e a média dos tempos, em segundos, para calcular o ótimo inteiro, campo *Tempo (s) Ótimo F1_PUVE* e *Tempo (s) Ótimo F2_PUVE*.

Pela Tabela (3.1) é possível verificar o ganho ao se utilizar as restrições redundantes (3.13)-(3.18). Com essas restrições foi possível resolver instâncias maiores com um tempo

Tabela 3.1: Comparação dos modelos de Fluxos para o Problema de Um Veículo de Entrega

Tamanho	Gap RL F1_PUVE (%)	Tempo (s) Ótimo F1_PUVE	Gap RL F2_PUVE (%)	Tempo (s) Ótimo F2_PUVE
10	5.71	1.05	0.34	0.21
12	8.06	4.88	0.11	0.70
15	8.77	25.62	0.78	5.64
20	12.33	2443.83	1.86	62.95
25	*****	*****	1.96	342.21
30	*****	*****	2.69	1632.02

computacional menor. Esse ganho no tempo também está relacionado ao *gap* de relaxação linear que diminui drasticamente quando se acrescentam as restrições redundantes da formulação F2_PUVE.

Na Tabela (3.2) são apresentados tantos os resultados exatos - associados aos modelos M1_PUVE, M2_PUVE, M3_PUVE e M4_PUVE - como os heurísticos. Essa tabela possui os seguintes campos: *Tamanho* que mostra o tamanho das instâncias analisadas; *Gap (%) Relaxação Linear M1_PUVE*, *Gap (%) Relaxação Linear M2_PUVE* e *Gap (%) Relaxação Linear M3_PUVE*, campos que mostram os *gaps* de relaxação linear para as formulações M1_PUVE, M2_PUVE e M3_PUVE. Para a abordagem M4_PUVE, esse *gap* não foi apresentado pois é igual ao da formulação M3_PUVE. *Tempo Ótimo M1_PUVE (s)*, *Tempo Ótimo M2_PUVE (s)*, *Tempo Ótimo M3_PUVE (s)* e *Tempo Ótimo M4_PUVE (s)*, campos que apresentam o tempo, em segundos, para se provar a otimalidade usando-se cada uma das abordagens citadas anteriormente. Além desses, são apresentados resultados relativos à heurística GRASP. O campo *Tempo Grasp (s)* apresenta o tempo, em segundos, para se encontrar o melhor limite superior e, finalmente, o campo *Gap (%) GRASP Ótimo* mostra o *gap*, em porcentagem, entre a melhor solução encontrada pelo GRASP e a solução ótima. Esse *gap* é computado para cada instância pela seguinte fórmula:

$$Gap\ \ (\%)\ \ GRASP\ \ Otimo = 100(\frac{Sol\ GRASP - VO}{Sol\ GRASP}) \tag{3.27}$$

onde *VO* é o valor ótimo e *Sol Grasp* é a solução encontrada pelo algoritmo GRASP. Na Tabela (3.2) são apresentados somente os valores médios para cada conjunto de instâncias.

Para as abordagens M1_PUVE, M2_PUVE, M3_PUVE e M4_PUVE, o CPLEX é exe-

Tabela 3.2: Comparação das Abordagens para o PUVE

Tamanho	Gap (%) Relaxação Linear M1_PUVE	Tempo Ótimo M1_PUVE (s)	Gap (%) Relaxação Linear M2_PUVE	Tempo Ótimo M2_PUVE (s)	Gap (%) Relaxação Linear M3_PUVE	Tempo Ótimo M3_PUVE (s)	Tempo GRASP (s)	Gap (%) GRASP Ótimo	Tempo Ótimo M4_PUVE (s)
10	0.86	0.063	0.86	0.086	0.86	0.072	0.00	0.00	0.063
12	0.77	0.17	0.77	0.181	0.77	0.163	0.00	0.00	0.148
15	0.84	0.409	0.82	0.42	0.82	0.29	0.01	0.00	0.26
20	1.78	4.07	1.96	0.42	1.78	2.76	0.07	0.00	1.26
25	1.91	15.17	1.97	15.27	1.91	12.16	0.93	0.00	5.24
30	2.23	1084.12	2.19	277.73	2.06	107.55	3.21	0.00	23.44
35	1.29	157.97	1.37	196.92	1.29	107.55	9.94	0.02	27.33
40	****	****	****	****	1.99	1826.14	75.03	0.00	340.28
45	****	****	****	****	2.96	****	201.15	0.28	3970.29
50	****	****	****	****	2.26	****	590.95	0.97	6023.10 (4)

cutado duas vezes para cada instância. Na primeira vez a ênfase do CPLEX está em um balanço entre otimalidade e viabilidade inteira (ênfase padrão do CPLEX). Na segunda vez, a ênfase do CPLEX está somente na otimalidade. A Tabela (3.2) apresenta o melhor resultado - em relação ao tempo de computação - dentre as duas ênfases executadas. Os valores entre parênteses apresentam o número de instâncias que não foram resolvidas até a otimalidade em cada tamanho de problemas. Essas instâncias não são utilizadas para calcular as médias dos tempos e dos *gaps*.

3.7 Análise dos Resultados

Neste capítulo são apresentados resultados exatos para instâncias do PUVE de até 50 vértices. Pelo que se sabe, só instâncias de 25 vértices tinham sido resolvidas até a otimalidade. Isso se deve, em parte, ao aumento do poder computacional e à força do resolvedor CPLEX da ILOG. Além disso, é importante destacar o impacto das restrições (3.25) e (3.26) e, principalmente, a utilização do limite superior pré-computado pela heurística GRASP. A utilização desse limite superior possibilitou diminuições drásticas nos tempos computacionais. Por exemplo: na instância I40-2, o tempo para se provar a otimalidade da solução inteira foi mais de 10 vezes mais rápido quando se utilizou o limite superior computado pelo GRASP, vide apêndice B.

As formulações F1_PUVE e F2_PUVE conseguiram resultados satisfatórios mostrando, mais uma vez, a importância de se utilizar as restrições adicionais (3.13)-(3.18). Essa importância está relacionada à diminuição do tempo computacional e dos *gaps* de relaxação

linear. Esses últimos - formulação F2_PUVE - também se mostraram competitivos se comparados com os *gaps* da formulação M3_PUVE, que conseguiu os melhores resultados.

Comparando os limites inferiores calculados pela relaxação linear das formulações M1_PUVE, M2_PUVE e M3_PUVE, chegou-se à seguinte conclusão: os limites gerados pelas formulações M1_PUVE e M2_PUVE, apesar de serem iguais para a maioria das instâncias testadas, diferem em alguns casos uns dos outros. Em algumas instâncias a formulação M1_PUVE é mais justa; em outras, a formulação M2_PUVE é a que gera o melhor limite inferior. A formulação M3_PUVE, por agregar restrições das formulações M1_PUVE e M2_PUVE, possui sempre um limite inferior menor ou igual ao limite computado pelas abordagens M1_PUVE e M2_PUVE.

Pode-se também comparar os tempos de computação das abordagens M1_PUVE, M2_PUVE e M3_PUVE e M4_PUVE. Apesar de as formulações M1_PUVE, M2_PUVE e M3_PUVE possuírem tempos de computação parecidos, os resultados computacionais mostram que a formulação M3_PUVE é, na média, a mais rápida dentre essas três. Isso é verificado para as instâncias de tamanho 12, 15, 25, 30 e 35. Por isso, para as instâncias de tamanho 40 só foram calculados os tempos utilizando-se a formulação M3_PUVE e M4_PUVE. Apesar de a abordagem M3_PUVE ser eficiente, ela não é comparável com a abordagem M4_PUVE, que se mostrou, sem dúvida, ser a mais eficiente. Devido aos experimentos com as instâncias do Problema de Mínima Latência já era esperado que a utilização de um limite superior pré-computado melhorasse o tempo de solução do ótimo inteiro pelo resolvedor CPLEX. Entretanto, essa melhora foi ainda mais significativa no Problema de Um Veículo de Entrega do que no PML. Por exemplo: para as instâncias de 40 vértices (último conjunto em que foram comparadas as formulações M3_PUVE e M4_PUVE) houve um ganho de tempo, em média, de até 81% ao se utilizar a abordagem M4_PUVE. Esses experimentos mostram, outra vez, a vantagem de se passar um limite superior justo para o resolvedor CPLEX.

O limite superior justo utilizado pelo resolvedor CPLEX foi calculado por uma heurística GRASP que se mostrou também muito eficiente para as instâncias do PUVE. Assim como no Problema de Mínima Latência, para a maioria das instâncias testadas, a heurística encontrou o ótimo inteiro. Até 45 vértices foram testadas 90 instâncias. Dentre todas essas, somente em 6 delas não foi encontrada a solução ótima. Além disso, para essas instâncias de até 45 vértices, o maior *gap* encontrado entre a solução ótima e o valor encontrado pela heurística foi de 1.16%. Para as de 50 vértices esses *gaps* foram maiores, mas chegaram a, no máximo, 2.21%. Os tempos de computação da heurística, apesar de não muito altos,

ainda podem ser melhorados.

3.8 Conclusão

Neste capítulo estudou-se o Problema de Um Veículo de Entrega, uma generalização do Problema de Mínima Latência apresentado no capítulo anterior. Para resolver de forma exata o PUVE, utilizaram-se dois tipos de formulações. Primeiramente foram desenvolvidas duas formulações de fluxo multiproduto, baseadas nas formulações da seção (2.4.2). As formulações multifluxo deste capítulo e as do capítulo anterior são as mesmas, salvo que as deste capítulo possuem um parâmetro d_k associado à demanda em cada vértice. A formulação do capítulo do Problema de Mínima Latência é uma simplificação em que todos os d_k são unitários. Além disso, foram desenvolvidas formulações baseadas na formulação de Bianco *et al.*. [BM+89]. Um possível erro na formulação original foi detectado e foram apresentadas duas formas de corrigi-lo. Neste capítulo foram apresentadas três formulações baseadas no trabalho de Bianco *et al.*. [BM+89]. Todas elas apresentaram desempenhos parecidos, apesar das diferenças nos tempos de computação e dos limites de programação linear. Todas as formulações apresentaram limites de programação linear muito justos. Para as instâncias menores a formulação de fluxos apresentou resultados melhores, já para as instâncias maiores, a formulação M3_PUVE aparenta ser a mais adequada.

Um algoritmo especializado, baseado na heurística apresentada no capítulo 2, foi desenvolvido para gerar bons limites superiores para as instâncias criadas para o PUVE. Essa heurística se mostrou eficiente calculando o ótimo inteiro para a grande maioria das instâncias testadas. Para aquelas que o ótimo inteiro não foi encontrado, o *gap* gerado pelo GRASP foi, no máximo, 2.21% do ótimo. Além disso, o uso desses limites superiores como primeira solução para o resolvedor CPLEX tornou-o bem mais eficiente. Os tempos de computação foram diminuídos drasticamente, tornando o CPLEX, em alguns casos, até 10 vezes mais rápido.

Capítulo 4

Problema do Caixeiro Viajante Multiproduto

4.1 Introdução

Com o aumento da velocidade de processamento e capacidade de computação tornou-se possível criar algoritmos exatos para instâncias de problemas clássicos de otimização combinatória que não eram possíveis até alguns anos atrás. Dentre esses problemas se destaca o Problema do Caixeiro Viajante [DFJ54], [NW88], [GL00], [ABCC06], que tem sido objeto de intensa investigação desde que a sua primeira formulação matemática foi proposta por Dantzig, Fulkerson e Johnson em 1954 [DFJ54].

O PCV é importante na solução de diversos problemas de transporte e logística. Dentre eles é possível citar problemas de roteamento de veículos, entrega e/ou coleta de produtos, dentre outros. Por se tratar de um problema muito estudado, houve muito progresso na elaboração de algoritmos combinatórios que solucionassem instâncias do PCV original e de suas variantes clássicas. Por exemplo: foi encontrada recentemente a solução ótima para uma instância do PCV clássico com 85900 pontos de demanda. Esse recorde foi atingido pelo resolvedor CONCORDE que utiliza implementações do algoritmo *Branch-and-Bound* aliado a algoritmos de geração de cortes. Para resolver essa instância de 85900 pontos de demanda, foi necessário utilizar programação paralela [ABCC06].

Apesar do grande avanço na solução exata de instâncias clássicas do PCV, esses algoritmos não lidam com variantes onde uma estrutura de custo mais geral é necessária ou melhor utilizável para aplicações reais. Por exemplo: todos os operadores logísticos deparam com um conflito entre seu próprio custo operacional e a satisfação de seus clientes.

Dessa forma, pode ser um desafio escolher se é melhor apenas minimizar o tempo total de viagem do veículo ou assegurar um dado nível de qualidade de serviço para todos os clientes. Algumas organizações podem ainda desejar priorizar seus clientes mais importantes. Todas essas generalizações não são tratadas pelo clássico PCV e nem mesmo pelo Problema de Roteamento de Veículos [DR59] [LN87] [TV02].

Uma interessante variação do PCV, chamada de Problema do Caixeiro Viajante Multiproduto (PCVM) [Sar03] [SL07b] [SMLM07], é o foco deste capítulo. No PCVM, um único veículo, saindo de uma única origem, e entregando produtos para um dado número de pontos de demanda, deve realizar o percurso com o menor custo. Cada localidade recebe uma quantidade heterogênea de um determinado e diferente tipo de produto. Esse problema tem dois custos para cada arco: o fixo e o variável. O custo fixo é independente dos produtos que são transportados. Já o variável, não só depende da via que o veículo trafega, como da quantidade de produtos transportados e também do tipo do produto e/ou do cliente que está sendo atendido. A estrutura de custo dessa variante do PCV evita soluções ótimas que são insensíveis à qualidade de serviço requerida pelos consumidores. O PCVM tende a priorizar, em termos de ordem de visitação, consumidores com maiores demandas de produtos mais valiosos. Além disso, o PCVM pode considerar o custo e o risco adicional de transportar produtos perecíveis e produtos frágeis, além de priorizar clientes mais importantes.

O Problema do Caixeiro Viajante Multiproduto (PCVM) é um problema que engloba na sua função objetivo o PCV original e uma versão multiproduto e ponderada do Problema de Um Veículo de Entrega (PUVE) [BM$^+$89]. Como o PUVE é uma extensão do Problema de Mínima Latência [BCC$^+$94] (PML), o PCVM também engloba o PML.

Este capítulo é organizado da seguinte forma: a seção (4.2) apresenta formalmente o PCVM, juntamente com uma formulação matemática multiproduto para o mesmo; a seção (4.3) apresenta um exemplo gráfico que mostra a diferença entre o PUVE e o PCVM; a seção (4.4) apresenta uma formulação estendida para o PCVM; a seção (4.5) apresenta uma projeção dessa formulação estendida juntamente com um algoritmo *Cut-and-Branch* baseado no Método de Particionamento de Benders para resolver mais eficientemente instâncias do PCVM; na seção (4.6) é apresentado um algoritmo Lagrangeano e uma heurística para o PCVM, juntamente com todos os resultados computacionais deste capítulo; e, finalmente, a seção (4.7) conclui o capítulo.

4.2 Modelando o Problema do Caixeiro Viajante Multiproduto

O Problema do Caixeiro Viajante Multiproduto [Sar03] [SL07b] [SMLM07] pode ser definido da seguinte forma: considere um grafo completo, dirigido e assimétrico $G = (V, A)$, onde V representa um conjunto de vértices e A um conjunto de arcos. Suponha também que exista uma origem 1 e, para cada vértice $k \in V$, uma demanda d_k de um produto específico k deve ser entregue durante um circuito Hamiltoniano. A demanda da origem é igual a uma unidade e representa o operador logístico responsável pelas entregas que precisa retornar ao ponto de partida. Nesse problema, existem custos fixos relacionados a atravessar um arco $(i, j) \in A$ e também custos variáveis para cada produto diferente que está sendo transportado por arcos diferentes. O objetivo é entregar todas as demandas minimizando a soma dos custos fixos e variáveis.

No PCVM, o operador logístico paga um custo fixo para atravessar um determinado arco (i, j). Esse custo está relacionado à distância e à infra-estrutura de transporte e é pago quando o operador logístico atravessa um arco, mesmo utilizando um veículo sem nenhuma carga. Esse custo fixo pode ser visto como o custo considerado no PCV original. Além do custo fixo, o PCVM também incorpora a cada produto um custo variável que atravessa cada arco. Esse custo variável para cada tipo de produto trafegando em cada arco é uma função dos tipos e das quantidades de produtos transportados por aquele arco. O papel do custo variável é estabelecer o fato de acordo com o qual redes de transporte lidam com diferentes produtos de diferentes valores agregados que estão sendo transportados em quantidades diferentes por rotas também diferentes.

Nesse problema variante do PCV, os tipos e as quantidades de produtos que devem ser entregues e que são transportados através dos arcos influenciam de forma importante o custo total. Isso significa que os consumidores que necessitam de uma demanda maior de produtos mais valiosos ou de alto risco de transporte devem ser atendidos com uma prioridade mais alta. Outra interpretação cabível é que se as tradicionais variantes do PCV são *orientadas ao custo*, o PCVM é também *orientado ao cliente*. Por exemplo: materiais mais sensíveis podem exigir estrutura de transporte especial; produtos perecíveis podem pagar por refrigeração; clientes importantes podem ser atendidos com uma certa prioridade enquanto outros tipos de produtos e/ou clientes não requerem esse grau de atenção.

Para esse problema pode-se definir uma formulação de programação linear inteira mista

M com os seguintes conjuntos de variáveis:

$$x_{ij} = \begin{cases} 1 & \text{se o caixeiro percorre o arco}(i,j), \\ 0 & \text{caso contrário.} \end{cases}$$

f_{ijk}: fração da demanda total do produto k transportada no arco (i,j) destinado ao vértice de demanda k

e o seguintes conjuntos de parâmetros

b_{ij}: custo fixo (estrutural) de percorrer o arco (i,j).

c_{ijk}: custo unitário de transportar o produto k pelo arco (i,j).

d_k: demanda do vértice k.

O modelo permite que os custos variáveis sejam dependentes tanto do produto como do arco.

O modelo matemático M é:

$$\min \sum_{(i,j)\in A} (b_{ij}x_{ij} + \sum_{k\in V} c_{ijk}d_k f_{ijk}) \tag{4.1}$$

sujeito a

$$\sum_{i \in V | i \neq j} x_{ij} = 1 \qquad \forall j \in V \tag{4.2}$$

$$\sum_{j \in V | i \neq j} x_{ij} = 1 \qquad \forall i \in V \tag{4.3}$$

$$f_{ijk} \leq x_{ij} \qquad \forall i, j, k \in V, i \neq j \tag{4.4}$$

$$\sum_{j \in V | j \neq 1} f_{1j1} = 1 \tag{4.5}$$

$$\sum_{j \in V | j \neq 1} (f_{1jk} - f_{j1k}) = 1 \qquad \forall k \in V, k \neq 1 \tag{4.6}$$

$$\sum_{i \in V | i \neq 1} f_{i11} = 1 \tag{4.7}$$

$$\sum_{i \in V | i \neq k} (f_{ikk} - f_{kik}) = 1 \qquad \forall k \in V, k \neq 1 \tag{4.8}$$

$$\sum_{i \in V | i \neq j, i \neq k} (f_{jik} - f_{ijk}) = 0 \qquad \forall k, j \in V, j \neq k, j \neq 1 \tag{4.9}$$

$$\sum_{i,j,k \in V | i \neq j} f_{ijk} = \frac{n(n+1)}{2} \tag{4.10}$$

$$f_{ijk} \geq 0 \qquad \forall i, j, k \in V, i \neq j \tag{4.11}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V, i \neq j \tag{4.12}$$

onde $n = \mid V \mid$.

A função objetivo (4.1) soma os custos para todos os arcos da rede. Cada arco possui dois custos. O primeiro refere-se ao custo fixo de percorrer um arco. Ele é independente dos produtos que estão sendo transportados. O segundo refere-se ao custo associado com a transferência, pelos arcos do caminho, de todos os produtos, do vértice origem ao específico vértice de destino.

As equações (4.2) e (4.3) são as restrições de atribuição introduzidas em 1954 por Dantzig, Fulkerson e Johnson [DFJ54]. Elas garantem que existe somente um arco saindo e um arco chegando a cada vértice. As restrições (4.4) acoplam as variáveis x com as variáveis f. Elas asseguram que nenhum fluxo é permitido em um arco (i, j) a menos que o custo fixo b_{ij} seja pago para percorrer esse arco.

As equações (4.5) e (4.6) garantem que o fluxo total do produto k que é originado do vértice origem 1 é igual à demanda d_k do consumidor localizado no vértice k. A restrição

(4.5) é relativa ao caso especial em que $k = 1$. De outra forma, as restrições (4.7) e (4.8) impõem que a demanda específica d_k do produto k é igual ao fluxo total desse produto que chega ao vértice k. Da mesma forma, a restrição (4.7) é relativa ao caso especial em que $k = 1$. Restrições (4.9) garantem a conservação de fluxo de qualquer produto através dos vértices que não sejam destinos finais para esse produto. O fato de o fluxo de qualquer produto por qualquer arco não ser negativo é garantido pelas equações (4.11). Esse tipo de restrições de fluxo (4.5)-(4.9) é tradicional em termos de problemas de roteamento, como pode ser visto no trabalho de Laporte e Norbert [LN87]. Pelo que se sabe esse tipo de restrição de conservação de fluxo foi originalmente proposta por Garvin *et al.* [GCJS57]. A equação redundante (4.10) assegura que o número de produtos que percorrem todos os arcos é expresso por $\frac{n(n+1)}{2}$, onde $n = \mid V \mid$ (4.10). O fato que todas as variáveis x_{ij} são binárias é assegurado pelas restrições (4.12).

Gavish e Graves [GS79], Finke, Claus e Gunn [FCG84] e Laporte e Norbert [LN87], dentre outros, também mostraram modelos para o PCV e para o Problema de Roteamento de Veículos que utilizavam variáveis de fluxos como variáveis responsáveis pela eliminação de subciclos.

Em relação à tradicional formulação de Dantzig, Fulkerson e Johnson [DFJ54], que é limitada ao espaço das variáveis x_{ij}, a inclusão das variáveis de fluxo f_{ijk} aumenta de forma polinomial o número de variáveis do problema. Ao invés de trabalhar com somente uma variável binária para cada arco (i, j), a formulação (4.1)-(4.12) também opera com $\mid V \mid$ variáveis contínuas para cada arco (i, j).

Como dito anteriormente, o Problema do Caixeiro Viajante Multiproduto (PCVM)(4.1)-(4.12) pode ser reduzido ao clássico Problema do Caixeiro Viajante, ao Problema de Mínima Latência e também ao Problema de Um Veículo de Entrega. O PCVM torna-se o PCV se todos os custos variáveis forem nulos, $c_{ijk} = 0$. O PCVM torna-se o PML quando todas as demandas forem unitárias, $d_k = 1$, todos os custos fixos forem nulos, $b_{ij} = 0$, e os custos variáveis forem os mesmos para cada produto k.

Apesar de uma pequena modificação no que diz respeito à demanda de retorno d_1, observe que o PCVM é redutível ao PUVE quando os custos fixos b_{ij} forem nulos e os custos variáveis forem os mesmos para cada produto k.

4.3 Diferença entre os Problemas

Nesta seção é apresentado um exemplo didático que mostra a diferença entre o PUVE e o PCVM. É utilizado como grafo base o mesmo utilizado no capítulo sobre o Problema de Um Veículo de Entrega, Figura (3.1). Entretanto, para este capítulo, são acrescidos, no grafo original, os custos variáveis de cada produto em cada arco. O objetivo desse acréscimo é que o grafo passe a incorporar os custos relacionados com o PCVM. Para o grafo modificado são resolvidos o PUVE e o PCVM. É obvio que os custos variáveis acrescentados não têm relevância na solução no PUVE, mas são importantes na solução do PCVM. O intuito é mostrar, mesmo que para uma pequena instância, a diferença entre a solução do PUVE e do PCVM. Essa diferença está, como citada anteriormente, nos custos variáveis. É provável que, se para uma pequena instância já existe uma diferença significativa, para instâncias maiores a diferença será ainda maior.

É utilizado como instância padrão um grafo de 6 vértices. Nesse grafo $G = (V, A)$ tem-se o vértice origem (vértice 1), as demandas d_k, os custos fixos e também os custos variáveis. Para cada arco (i, j) tem-se um custo fixo b_{ij} e mais $\mid V \mid -1$ custos variáveis c_{ijk}. A Figura (4.1) mostra o grafo de 6 vértices, todos com demandas. Nota-se que sempre a demanda da origem é unitária. Na Figura (4.1), o número que aparece externamente ao lado de cada vértice indica a demanda de unidades do produto requerida no vértice. Nesse caso, para cada arco (i, j) tem-se um custo fixo e cinco custos variáveis. Cada custo variável representa o custo de se transmitir uma unidade do produto correspondente através do arco (i, j). Os cinco custos variáveis de cada arco são mostrados depois do custo fixo.

Para o PUVE, as demandas e os custos variáveis não influenciam na solução final. A Figura (4.2) mostra a solução do PUVE para o grafo exemplo (4.1) que é 1064 $((12 * 28) + (15 * 20) + (17 * 13) + (21 * 8) + (10 * 3) + (9 * 1))$.

A Figura (4.3) mostra a solução do PCVM para o grafo exemplo. A solução ótima inteira para esse PCVM com $\sum d_k = 28$ é 1800 .

É evidente que a comparação dos valores das soluções ótimas para os dois problemas não têm nenhuma importância, já que estão sendo comparados problemas com estruturas de custos diferentes. O importante nesse caso é a rota ótima de cada problema. Como tanto no PCVM quanto no PUVE as demandas em cada vértice são relevantes, o que caracteriza as diferenças na rota ótima dos dois problemas são os custos variáveis. Por exemplo, na solução do PCVM, o primeiro arco visitado é aquele que possui custos variáveis unitários para todos os produtos. O mesmo não acontece no PUVE, onde esses custos não têm importância.

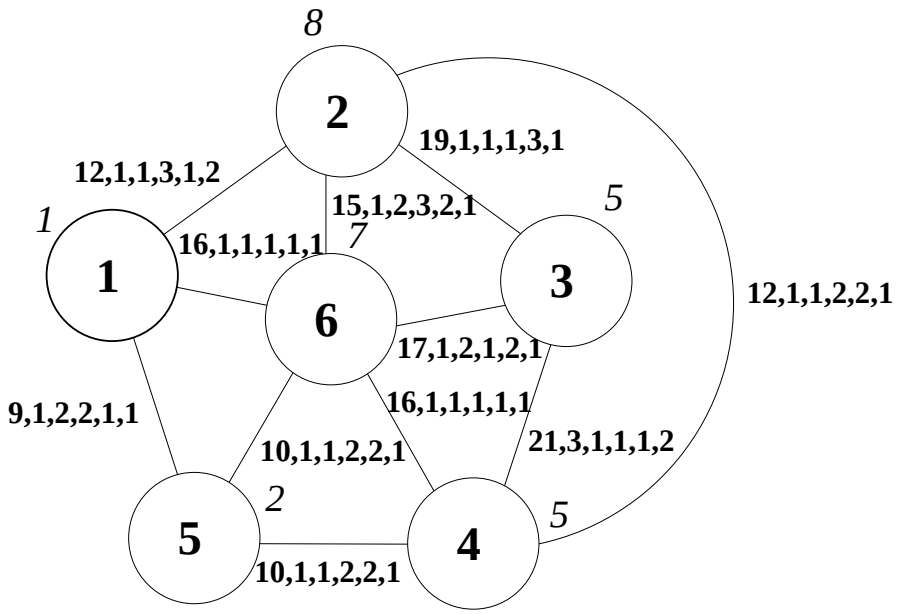


Figura 4.1: Exemplo PCVM.

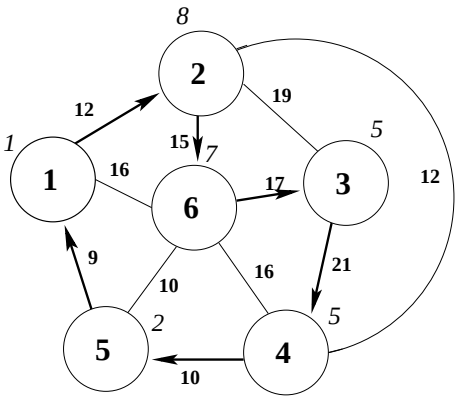


Figura 4.2: Solução PUVE.

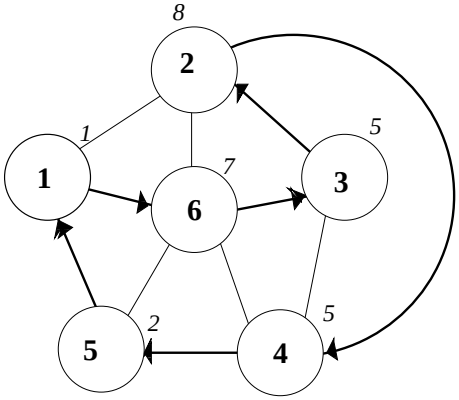


Figura 4.3: Solução PCVM.

4.4 Uma formulação estendida para o PCVM

A formulação (4.1)-(4.12) é suficiente para descrever o PCVM. Por outro lado, dado a associação entre o Problema Quadrático de Atribuição (PQA), do inglês, *Quadratic Assignment Problem* [KB57] e o Problema do Caixeiro Viajante, já apresentada no capítulo relativo ao Problema de Mínima Latência, é possível ter uma idéia de como tornar a formulação anterior mais forte com a ajuda de restrições geralmente associadas ao PQA. O trabalho de Vander Wiel e Sahinidis [WS95] é um exemplo de como conectar o PQA com variantes do PCV.

Quando se usa o PQA para resolver o PCV, é importante lembrar que as restrições de atribuição do PQA não são escritas no espaço dos arcos, isto é, não significam que um dado arco é percorrido somente uma vez em alguma direção. Elas asseguram que um dado vértice é visitado em apenas uma ordem de visitação e que uma dada ordem de visitação é usada por apenas um vértice somente uma vez. Para usar esse tipo de restrição de acoplamento com o modelo M, propõe-se um novo conjunto de variáveis p_{ij} definindo a ordem j de visitação do vértice i. Normalmente, para esse novo conjunto de variáveis, deveria ser criado um novo conjunto de ordens de visitação. Isso não será necessário, pois existem tantas ordens de visitação quantos os vértices de V. Essa observação é a chave para escrever as restrições de atribuição como:

$$\sum_{i \in V} p_{ij} = 1 \quad \forall j \in V \tag{4.13}$$

$$\sum_{j \in V} p_{ij} = 1 \quad \forall i \in V \tag{4.14}$$

$$p_{11} = 1 \tag{4.15}$$

$$p_{ij} \geq 0 \quad \forall i, j \in V \tag{4.16}$$

A restrição (4.15) garante que o primeiro vértice a ser utilizado é o vértice 1, isto é, o vértice origem. As equações de acoplamento (4.17) forçam o fluxo total do produto k a ser associado a uma ordem de visitação do vértice k. As restrições (4.18) garantem que para cada vértice um dado produto será entregue.

$$\sum_{k,j \in V | i \neq j} f_{ijk} = \sum_{l=1}^{n} (n - l + 1) p_{li} \quad \forall i \in V \tag{4.17}$$

$$\sum_{i,j \in V | i \neq j} f_{ijk} = \sum_{l=1}^{n} (l - 1) p_{lk} \quad \forall k \in V, k \neq 1 \tag{4.18}$$

Dada uma solução inteira para as variáveis x, é possível encontrar diretamente, por inspeção, o valor das variáveis p associadas. Entretanto, isso não é verdade se for encontrada a solução fracionária para as variáveis x. Nesse caso, as restrições acima são responsáveis por uma considerável melhora nos *gaps* de relaxação linear, impactando, na mesma maneira, no tempo de solução total do ótimo inteiro. Os experimentos numéricos da seção (4.6) estabelecem a importância dessas restrições adicionais na solução de grandes

instâncias do PCVM. É importante salientar que as variáveis p são escritas em um espaço diferente do espaço das variáveis x.

4.5 Projeção da formulação estendida para o PCVM

Nesta seção, um esforço adicional é realizado para tratar o PCVM. Esse esforço é um algoritmo *Cut-and-Branch* baseado no Método de Particionamento de Benders [Ben62], que tem como intuito gerar inequações válidas e fortes para a formulação (4.1)-(4.12). Essas inequações são o resultado da projeção das variáveis da formulação estendida do PCVM – equações (4.13)-(4.18). Essa revisitação da Decomposição de Benders foi sugerida em alguns trabalhos de Balas *et al.* [Bal79], [BP83], [Bal85]. Também é possível citar o livro de R. K. Martin [Mar99] que descreve o método em detalhes.

Pelo ponto de vista da programação matemática, a projeção do problema (4.1)-(4.18) no espaço das variáveis (x, f) pode ser realizada. Fixando o par $(x, f) = (\bar{x}, \bar{f})$, o seguinte sistema em p precisa ser satisfeito:

$$\sum_{i \in V} p_{ij} = 1 \qquad \forall j \in V \tag{4.19}$$

$$\sum_{j \in V} p_{ij} = 1 \qquad \forall i \in V \tag{4.20}$$

$$p_{11} = 1 \tag{4.21}$$

$$\sum_{l=1}^{n} (n - l + 1) p_{li} = \sum_{k,j \in V | i \neq j} \bar{f}_{ijk} \qquad \forall i \in V \tag{4.22}$$

$$\sum_{l=1}^{n} (l - 1) p_{lk} = \sum_{i,j \in V | i \neq j} \bar{f}_{ijk} \qquad \forall k \in V, k \neq 1 \tag{4.23}$$

$$p_{ij} \geq 0 \qquad \forall i, j \in V \tag{4.24}$$

Para gerar o corte mais profundo no espaço (x, f) a partir da violação desse sistema, é preciso reescrevê-lo como um problema de otimização:

$$\min \; e_1 \; + \; e_2 \; + \; e_3 \; + \; e_4 \tag{4.25}$$

sujeito a

$$\sum_{i \in V} p_{ij} = 1 \qquad \forall j \in V \tag{4.26}$$

$$\sum_{j \in V} p_{ij} = 1 \qquad \forall i \in V \tag{4.27}$$

$$p_{o1} = 1 \tag{4.28}$$

$$\sum_{l=1}^{n} (n-l+1) p_{li} + e_1 \geq \sum_{k,j \in V | i \neq j} \bar{f}_{ijk} \qquad \forall i \in V \tag{4.29}$$

$$-\sum_{l=1}^{n} (n-l+1) p_{li} + e_2 \geq -\sum_{k,j \in V | i \neq j} \bar{f}_{ijk} \qquad \forall i \in V \tag{4.30}$$

$$\sum_{l=1}^{n} (l-1) p_{lk} + e_3 \geq \sum_{i,j \in V | i \neq j} \bar{f}_{ijk} \qquad \forall k \in V, k \neq 1 \tag{4.31}$$

$$-\sum_{l=1}^{n} (l-1) p_{lk} + e_4 \geq -\sum_{i,j \in V | i \neq j} \bar{f}_{ijk} \qquad \forall k \in V, k \neq 1 \tag{4.32}$$

$$p_{ij} \geq 0 \qquad \forall i, j \in V \tag{4.33}$$

$$e_1, e_2, e_3, e_4 \geq 0 \tag{4.34}$$

onde $e_1,\ e_2,\ e_3$ e e_4 são as inviabilidades no espaço p dada a solução fracionária apenas para o par (x, f). Associando variáveis duais v_j, u_i, s, $z_i^1 \geq 0$, $z_i^2 \geq 0$, $w_k^1 \geq 0$ e $w_k^2 \geq 0$, respectivamente para as restrições (4.26), (4.27), (4.28), (4.29), (4.30), (4.31) e (4.32), é possível escrever o dual da formulação (4.25)-(4.34) como:

$$\max \sum_{j \in V} v_j + \sum_{i \in V} u_i + s + \sum_{i,j,k \in V} \bar{f}_{ijk}(z_i^1 - z_i^2) + \sum_{i,j,k \in V} \bar{f}_{ijk}(w_k^1 - w_k^2) \tag{4.35}$$

sujeito a

$$s + u_o + v_1 + n\left[z_1^1 - z_1^2\right] + (o-1)\left[w_1^1 - w_1^2\right] \leq 0 \quad (1,1) \in V^2 \tag{4.36}$$

$$u_i + v_j + (n-i+1)\left[z_j^1 - z_j^2\right] + (i-1)\left[w_j^1 - w_j^2\right] \leq 0$$

$$\forall (i,j) \in V^2, (i,j) \neq (1,1) \tag{4.37}$$

$$\sum_{i \in V} z_i^1 \leq 1 \tag{4.38}$$

$$\sum_{i \in V} z_i^2 \leq 1 \tag{4.39}$$

$$\sum_{k \in V} w_k^1 \leq 1 \tag{4.40}$$

$$\sum_{k \in V} w_k^2 \leq 1 \tag{4.41}$$

$$z_i^1 \geq 0 \quad \forall i \in V \tag{4.42}$$

$$z_i^2 \geq 0 \quad \forall i \in V \tag{4.43}$$

$$w_k^1 \geq 0 \quad \forall k \in V \tag{4.44}$$

$$w_k^2 \geq 0 \quad \forall k \in V \tag{4.45}$$

Como o poliedro definido por (4.36)-(4.45) tem um número finito de raios extremos, indexando esses raios por $h = 1..H$, têm-se inequações válidas para o problema (4.1)-(4.12) através da violação das equações (4.13)-(4.18) na forma de:

$$\sum_{j \in V} v_j^h + \sum_{i \in V} u_i^h + s^h + \sum_{i,j,k \in V} f_{ijk}(z_i^{1^h} - z_i^{2^h}) + \sum_{i,j,k \in V} f_{ijk}(w_k^{1^h} - w_k^{2^h}) \leq 0 \ \ \forall h \in H \tag{4.46}$$

É sabido que quando se usa um algoritmo baseado na decomposição de Benders o maior problema é a ausência de uma estrutura especial no subproblema de Benders. Observando a formulação (4.35)-(4.45) é possível ver que, após uma pequena manipulação, é possível derivar um sistema de violação equivalente com uma estrutura especial. Primeiramente é preciso observar que, no Problema de Atribuição em p, a primeira atribuição já está calculada, uma vez que qualquer circuito começa da origem 1. Isso já é assegurado no problema mestre (problema em (x, f)) que torna a restrição (4.21) redundante e desnecessária. Outra importante constatação é que existem tantos produtos quantos vértices em V e que apenas um produto é entregue a cada vértice. Com a ajuda dessas considerações é possível

reescrever as restrições (4.19)-(4.24) como:

$$\sum_{i \in V} p_{ij} = 1 \qquad \forall j \in V, j \neq 1 \tag{4.47}$$

$$\sum_{i \in V} p_{ji} = 1 \qquad \forall j \in V, j \neq 1 \tag{4.48}$$

$$\sum_{i=1}^{n} (n - i + 1) p_{ij} = \sum_{k, i \in V | i \neq j} \bar{f}_{jik} \qquad \forall j \in V,\ j \neq 1 \tag{4.49}$$

$$\sum_{i=1}^{n} (i - 1) p_{ij} = \sum_{i, l \in V | i \neq l} \bar{f}_{ilj} \qquad \forall j \in V,\ j \neq 1 \tag{4.50}$$

$$p_{ij} \geq 0 \qquad \forall i, j \in V, j \neq 1. \tag{4.51}$$

O sistema violado equivalente permite a solução de um subproblema para cada $j \in V,\ j > 1$. Também é possível calcular a violação de cada subsistema para a solução do problema mestre. O problema de encontrar o corte mais violado para um dado $j \in V,\ j > 1$ é definido como:

$$\min\ e_j^1 \ +\ e_j^2 \ +\ e_j^3 \ +\ e_j^4 \tag{4.52}$$

sujeito a

$$\sum_{i \in V} p_{ij} = \quad 1 \tag{4.53}$$

$$\sum_{i \in V} p_{ji} = \quad 1 \tag{4.54}$$

$$\sum_{i=1}^{n} (n-i+1) p_{ij} + e_j^1 \geq \quad \sum_{k,i \in V | i \neq j} \bar{f}_{jik} \tag{4.55}$$

$$-\sum_{i=1}^{n} (n-i+1) p_{ij} + e_j^2 \geq \quad -\sum_{k,i \in V | i \neq j} \bar{f}_{jik} \tag{4.56}$$

$$\sum_{i=1}^{n} (i-1) p_{ij} + e_j^3 \geq \quad \sum_{i,l \in V | i \neq l} \bar{f}_{ilj} \tag{4.57}$$

$$-\sum_{i=1}^{n} (i-1) p_{ij} + e_j^4 \geq \quad -\sum_{i,l \in V | i \neq l} \bar{f}_{ilj} \tag{4.58}$$

$$p_{ij} \geq \quad 0 \qquad \forall i \in V \tag{4.59}$$

$$e_j^1 \geq \quad 0 \tag{4.60}$$

$$e_j^2 \geq \quad 0 \tag{4.61}$$

$$e_j^3 \geq \quad 0 \tag{4.62}$$

$$e_j^4 \geq \quad 0 \tag{4.63}$$

A correspondente versão dual do programa linear acima é:

$$\max \; v_j + \sum_{i \in V} u_i + \sum_{i,k \in V} \bar{f}_{jik}(z_j^1 - z_j^2) + \sum_{i,k \in V} \bar{f}_{ikj}(w_j^1 - w_j^2) \tag{4.64}$$

sujeito a

$$u_i + v_j + (n-i+1)\left[z_j^1 - z_j^2\right] + (i-1)\left[w_j^1 - w_j^2\right] \leq \quad 0 \qquad \forall i \in V \tag{4.65}$$

$$0 \leq z_j^1 \quad \leq \quad 1 \tag{4.66}$$

$$0 \leq z_j^2 \quad \leq \quad 1 \tag{4.67}$$

$$0 \leq w_j^1 \quad \leq \quad 1 \tag{4.68}$$

$$0 \leq w_j^2 \quad \leq \quad 1 \tag{4.69}$$

Nesse caso, a expressão para a inequação válida é dada por:

$$\sum_{j \in V|j>1} v_j^h + \sum_{j \in V|,j \neq 1} u_j^h + \sum_{i,j,k \in V|j,j \neq 1} f_{jik}(z_j^{1^h} - z_j^{2^h}) + \sum_{i,j,k \in V|j,k \neq 1} f_{ikj}(w_j^{1^h} - w_j^{2^h}) \leq 0 \ \ \forall h \in H \tag{4.70}$$

Além da vantagem óbvia de resolver $n-1$ pequenos subproblemas ao invés de um único grande subproblema, as manipulações anteriores habilitam o uso de uma versão multicorte da Decomposição de Benders. Essa versão foi proposta por Birge, J. R. e Louveaux, F. V. [BL88]. A versão multicorte das inequações válidas é:

$$v_j^h + u_j^h + \sum_{i,k \in V|k \neq 1} f_{jik}(z_j^{1^h} - z_j^{2^h}) + \sum_{i,k \in V|k \neq 1} f_{ikj}(w_j^{1^h} - w_j^{2^h}) \leq 0 \ \ \forall h \in H, \ \forall j \in V, j \neq 1 \tag{4.71}$$

São executados na próxima seção muitos experimentos computacionais para comparar o método *Cut-and-Branch* contra o CPLEX monolítico e os métodos Lagrangeanos da seção (4.6.3).

4.6 Experimentos Computacionais

Os experimentos computacionais foram executados em um computador Pentium IV com um processador de 3.0 Ghz e 1.0 Gbyte de memória RAM. O sistema operacional utilizado é o Linux. O resolvedor ILOG CPLEX 9.1.3 é usado para computar a solução ótima inteira.

Nas instâncias testadas, cada vértice $k \in V$ possui uma demanda entre 1 e o valor do parâmetro *Max Demanda* que varia entre 5 e 20 unidades. A origem tem demanda igual a 1 e todas as demandas são inteiras. Os valores de b_{ij} relativos à distância entre os vértices foram gerados aleatoriamente de acordo com uma distribuição uniforme com valores entre $[1, 100]$. Após a geração desses valores a desigualdade triangular foi garantida através do algoritmo de caminhos mínimos, da mesma forma que Fischetti, Laporte e Martello [FLM93] fizeram. O custo variável c_{ijk} de transportar o produto k pelo arco (i, j) é calculado utilizando-se o custo fixo (b_{ij}) e outros parâmetros adicionais. Cada c_{ijk} é calculado pela seguinte expressão:

$$c_{ijk} = \xi_{ij} \cdot \eta_k$$

O parâmetro ξ_{ij} representa uma relação pseudo-aleatória entre o custo de transporte dos produtos transportados no arco (i, j) e o valor de b_{ij}. Esse parâmetro é igual para todos os produtos que trafegam em um determinado arco, mas pode variar de um arco para outro. Quanto melhores forem as condições de um determinado arco, menor será o valor do parâmetro ξ correspondente. Esse parâmetro varia entre 0.5% e 2% do custo fixo b_{ij}. Cada vértice tem um parâmetro η_k, escolhido aleatoriamente entre 0.5 e 1.5. Para cada vértice um tipo específico de produto deve ser entregue. O valor de η_k representa o valor relativo do produto, em que um η_k maior representa um produto mais valioso ou um produto perecível que precisa ser entregue mais rapidamente no vértice k. No PCVM, o η_k pode diferenciar clientes ao invés de produtos. Mesmo que os produtos associados a dois clientes tenham o mesmo valor agregado, um cliente pode ter uma prioridade maior de entrega impactando no valor do η_k correspondente. Todas as instâncias são assimétricas. Para cada classe de valor de n, dez instâncias pseudo-aleatórias foram criadas.

Primeiramente, para resolver o PCVM, são testadas duas formulações diferentes. A primeira, chamada de formulação M, é constituída apenas das equações (4.1)-(4.12). A segunda formulação, que é chamada de M2, é o modelo M acrescido das inequações válidas (4.13)-(4.18).

A Tabela (4.1) apresenta as médias dos tempos de solução para cada conjunto de dez instâncias. Cada instância foi resolvida duas vezes até a otimalidade. Na primeira vez foi utilizada a formulação M e, na segunda, a formulação M2. Em [SL07b], Sarubbi e Luna apresentaram resultados exatos para o mesmo problema para instâncias de até 65 vértices. Entretanto, aquelas instâncias não estavam associadas a um grafo completo.

A Tabela (4.1) apresenta resultados ótimos para instâncias de até 50 vértices associadas a grafos completos. O modelo M foi capaz de encontrar resultados ótimos para instâncias de até 30 vértices. Já o modelo M2 foi capaz de encontrar resultados ótimos para instâncias de até 50 vértices. Além disso, os *gaps* de relaxação linear do modelo M2 são mais justos que os computados pelo modelo M. A relaxação de programação linear desses dois modelos foi resolvida tanto pelo método da Barreira de Newton como pelo método Dual Simplex. O método da Barreira, entretanto, aparenta ser a melhor escolha devido ao tempo gasto na computação para essas instâncias.

Na Tabela (4.1) todos os campos mostram a média dos valores para cada conjunto

Tabela 4.1: Resultados CPLEX Monolítico para o PCVM

Tamanho	Gap (%) Relax Linear M	Tempo (s) Ótimo Modelo M	Tempo (s) Relax Linear Barreira M	Gap (%) Relax Linear M2	Tempo (s) Ótimo Modelo M2	Tempo (s) Relax Linear Barreira M2	Tempo (s) Relax Linear Dual M2
10	0.67	0.24	0.04	0.37	0.19	0.09	1.54
15	1.04	1.45	0.18	0.24	1.9	0.52	3.02
20	2.33	31.86	1.29	0.41	22.56	2.5	11.34
25	2.56	503.26	3.67	0.66	169.01	6.05	31.01
30	2.56	12827(1)	9.35	0.66	745.41	11.58	138.38
35	5.52	******	17.51	0.74	2395.86	27.16	353.83
40	6.04	******	59.28	0.70	3760.77	52.42	868.56
45	7.12	******	101.2	0.99	37889.49	90.55	1919.48
50	8.31	******	187.87	1.07	86356.09(1)	162.69	******

de 10 instâncias. Ela possui os seguintes campos: *Tamanho*, representa o tamanho das instâncias testadas; *Gap (%) Relax Linear M* campo que apresenta o *gap* de relaxação linear utilizando a formulação M; *Tempo (s) Ótimo Modelo M* campo que relata o tempo, em segundos, para provar a otimalidade das instâncias usando-se a formulação M; *Tempo (s) Relax Linear Modelo M* mostra o tempo, em segundos, para calcular a relaxação linear das instâncias. Esses *gaps* e tempos são também calculados para a formulação M2. Nessa última formulação também é mostrado o tempo para se calcular o limite de programação linear utilizando-se, ao invés do método da Barreira, o método Dual Simplex. Como pode-se ver na Tabela (4.1) o método da Barreira se mostrou a melhor alternativa quando comparado ao modelo M2. Por isso, os valores do campo *Tempo (s) Ótimo Modelo M2* foram calculados utilizando-se como algoritmo de programação linear o método da Barreira. Os valores entre parênteses mostram o número de instâncias que não foram resolvidas até a otimalidade. Esse também foi o critério para deixar de resolver instâncias maiores. Por exemplo: como no modelo M uma instância de tamanho 30 não foi resolvida, não foram testadas instâncias de tamanho 35. O mesmo aconteceu para as instâncias de tamanho 50 usando-se a formulação M2.

4.6.1 Resultados *Cut-and-Branch*

Produziu-se uma implementação do método de decomposição de Benders baseado no algoritmo de *Cut-and-Branch* sugerido na seção (4.5). Para comparar a eficiência do método repetiram-se todos os testes para as mesmas instâncias da seção anterior. A idéia aqui é tornar o procedimento de *Branch-and-Bound* mais rápido se existir um modelo tão justo quanto o modelo M2 (em relação ao limite de programação linear) e com o mesmo número

de variáveis do modelo M. Uma vez que o passo de enumeração é a parte mais cara na solução de um problema inteiro e sabendo-se que qualquer método de *Branch-and-Bound* é sensível ao número de variáveis espera-se uma melhora no tempo de computação através dessa técnica. É bom salientar que não é necessário adicionar todas as inequações violadas (cortes), sendo, em alguns casos, melhor adicionar apenas algumas. Tudo o que se precisa é capturar informação suficiente sobre o sistema violado, impactando assim no procedimento de enumeração.

Nas Tabelas (4.2)-(4.6) mostra-se a comparação entre as três versões do algoritmo de *Cut-and-Branch* e o resolvedor CPLEX. A primeira (P1) é aquela em que foi utilizada a formulação (4.36)-(4.45). Nesta versão não é aproveitada a vantagem da estrutura especial associada a P1. Na segunda, (P2), apenas é utilizada a estrutura especial para resolver os subproblemas, mas adicionando, um corte por iteração, na forma (4.70). A terceira (P3) é a versão multicorte, conforme (4.71).

O campo *Tempo Total(s) CPLEX* apresenta o tempo total, em segundos, gasto pelo CPLEX para resolver a instância, até a otimalidade, usando a formulação (4.1)-(4.18). Também foram tabulados esses tempos de computação considerando-se as três versões do algoritmo *Cut-and-Branch*, juntamente com o tempo de geração de cortes de cada uma delas. Esses últimos tempos estão associados aos campos *Tempo Projeção P1 (s)*, *Tempo Projeção P2 (s)* e *Tempo Projeção P3 (s)*. Na maioria dos casos não se repetiu a geração de cortes até o final. Pode-se parar o procedimento de geração de cortes quando a taxa de melhora do limite inferior começa a decair. Essa decisão é, de alguma forma, heurística, mas, de um modo geral, pode-se parar a geração de cortes se o erro relativo no sistema violado está abaixo dos 5%. Os campos intitulados *Gap(%) Relax Linear* mostram os *gaps* de integralidade do modelo M2. Os campos *Gap Px Final (%)* mostram o *gap* final para cada instância e para cada versão do algoritmo *Cut-and-Branch*, onde x é a versão, depois de serem adicionadas as inequações válidas. Quando o *Gap* Final é o mesmo da coluna *LPR Gap (%)*, o sistema violado foi resolvido até o final, mesmo assegurando apenas 5% de erro relativo. É importante salientar que todas as instâncias foram resolvidas até a otimalidade. Os campos *Gap P1 Final (%)*, *Gap P2 Final (%)* e *Gap P3 Final (%)* apresentam os *gaps*, relativos à solução ótima, após ser executado o procedimento de projeção até o final, isto é, antes de começar o procedimento de enumeração.

É possível verificar que mesmo não se utilizando a estrutura especial, somente gerando cortes para o sistema violado, houve uma redução dos tempos computacionais em relação ao CPLEX monolítico. Na versão P1 essa economia é registrada no procedimento de

Tabela 4.2: Resultados Cut-and-Branch para o PCVM - Instâncias 30

Prob	Gap(%) Relax Linear	Tempo Total(s) CPLEX	Gap P1 Final (%)	Tempo Projeção P1 (s)	Tempo C-&-B P1 (s)	Gap P2 Final (%)	Tempo Projeção P2 (s)	Tempo C-&-B P2 (s)	Gap P3 Final (%)	Tempo Projeção P3 (s)	Tempo C-&-B P3 (s)
30-1	1.21	454.92	1.48	40.85	221.67	1.21	132.46	236.65	1.21	20.46	120.13
30-2	1.46	732.92	1.82	18.89	123.33	1.47	103.79	230.45	1.46	19.96	144.55
30-3	0.31	302.74	0.80	62.3	152.22	0.31	79.22	103.25	0.31	17.76	123.72
30-4	0.78	754.03	1.62	39.27	430.25	0.78	82.03	199.55	0.78	24.00	141.82
30-5	0.78	395.15	2.90	59.66	218.09	0.80	77.31	177.71	0.80	21.10	133.00
30-6	1.10	1653.94	3.48	102.9	551.13	1.10	133.53	535.36	1.10	24.10	390.91
30-7	1.03	459.29	5.53	139.79	429.66	1.15	92.53	191.09	1.13	20.42	172.27
30-8	0.14	80.19	0.92	41.81	175.86	0.14	133.73	253.39	0.14	17.76	76.31
30-9	0.09	287.53	1.10	64.38	153.61	0.10	84.50	188.62	0.09	22.89	289.47
30-10	1.26	955.54	3.00	67.15	285.45	1.27	159.52	955.54	1.28	20.35	231.39

Tabela 4.3: Resultados Cut-and-Branch para o PCVM - Instâncias 35

Prob	Gap(%) Relax Linear	Tempo Total(s) CPLEX	Gap P1 Final (%)	Tempo Projeção P1 (s)	Tempo C-&-B P1 (s)	Gap P2 Final (%)	Tempo Projeção P2 (s)	Tempo C-&-B P2 (s)	Gap P3 Final (%)	Tempo Projeção P3 (s)	Tempo C-&-B P3 (s)
35-1	0.00	25.44	0.01	285.33	436.45	0.00	72.59	73.22	0.00	56.28	1115.84
35-2	0.00	27.34	0.01	216.56	439.34	0.00	80.95	81.84	0.00	38.05	38.58
35-3	0.14	430.93	0.19	217.83	438.54	0.15	201.63	290.36	0.14	47.21	115.88
35-4	1.21	7578.2	1.44	150.17	2763.4	1.24	165.27	1567.37	1.25	43.43	1092.8
35-5	0.87	1411.5	1.10	288.83	827.84	0.89	125.27	453.19	0.87	44.85	819.17
35-6	0.73	1114.09	1.04	320.05	1633.67	0.74	163.83	671.36	0.73	40.46	469.1
35-7	2.40	8197.01	2.82	240.73	1401.19	2.47	162.65	1053.51	2.44	39.75	1617.7
35-8	0.60	1628.26	0.81	198.85	1046.68	0.61	163.74	812.88	0.60	71.88	750.55
35-9	0.43	2004.55	0.54	257.41	764.34	0.44	324.72	661.07	0.54	72.83	711.57
35-10	0.99	1541.29	1.29	286.85	898.46	1.06	134.23	342.59	0.99	50.61	241.85

Tabela 4.4: Resultados Cut-and-Branch para o PCVM - Instâncias 40

Prob	Gap(%) Relax Linear	Tempo Total(s) CPLEX	Gap P1 Final (%)	Tempo Projeção P1 (s)	Tempo C-&-B P1 (s)	Gap P2 Final (%)	Tempo Projeção P2 (s)	Tempo C-&-B P2 (s)	Gap P3 Final (%)	Tempo Projeção P3 (s)	Tempo C-&-B P3 (s)
40-1	0.00	25.44	0.05	243.86	711.1	0.00	344.03	345.85	0.00	67.2	67.98
40-2	1.81	27.34	1.86	679.25	3770.16	1.81	661.38	3355.83	1.84	72.61	4678.16
40-3	0.01	430.93	0.07	753.76	2513.26	0.02	440.71	679.71	0.01	118.5	328.37
40-4	0.00	7578.2	0.23	466.52	858.13	0.00	169.15	170.3	0.00	81.47	82.27
40-5	0.35	1411.5	0.67	541.45	922.44	0.36	335.62	452.7	0.35	86.23	186.08
40-6	0.27	1114.09	0.46	650.14	1134.07	0.28	391.15	575.14	0.29	90.42	262.27
40-7	1.53	8197.01	2.14	701.3	2825.81	1.55	270.98	1325.99	1.54	90.61	2036.49
40-8	0.92	1628.26	1.08	629.6	1210.29	0.92	664.17	1033.76	0.92	73.85	417.08
40-9	0.32	2004.55	0.51	519.59	2328.86	0.32	357.37	874.84	0.32	93.1	631
40-10	1.83	1541.29	2.12	701.43	5367.49	1.84	501.45	3904.27	1.83	93.62	5345.49

Tabela 4.5: Resultados Cut-and-Branch para o PCVM - Instâncias 45

Prob	Gap(%) Relax Linear	Tempo Total(s) CPLEX	Gap P1 Final (%)	Tempo Projeção P1 (s)	Tempo C-&-B P1 (s)	Gap P2 Final (%)	Tempo Projeção P2 (s)	Tempo C-&-B P2 (s)	Gap P3 Final (%)	Tempo Projeção P3 (s)	Tempo C-&-B P3 (s)
45-1	0.00	89.2	0.01	654.94	741.7	0.00	315.53	317.32	0.00	124.44	125.58
45-2	0.52	46178.66	0.58	1674.07	3336.57	0.52	1498.76	4024.13	0.52	141.46	3619.1
45-3	0.01	539.6	0.08	1212.79	1976.64	0.02	725.54	954.36	0.01	128.21	255.88
45-4	1.43	20313.44	1.48	1534.56	10981.67	1.44	624.38	4698.78	****	****	****
45-5	0.51	12033.03	0.64	1524.52	3203.56	0.53	920.01	2093.46	****	****	****
45-6	2.33	82253.25	2.45	1230.38	13086.87	2.33	1070.32	20530.55	2.34	142.97	14881.86
45-7	2.29	167570.17	2.79	1273.7	161989.6	2.34	592.4	24710.12	2.32	147.36	39135.17
45-8	1.26	14150.74	1.33	1443.99	6512.17	1.26	820.65	4217.2	1.28	155.53	3414.1
45-9	0.24	8467.1	0.37	1549.8	4239.09	0.25	767.27	1917.63	0.24	184.07	901.5
45-10	1.28	27299.73	1.45	3292.97	23381.94	1.30	903.44	10155.04	1.29	182.86	8154.3

Tabela 4.6: Resultados Cut-and-Branch para o PCVM - Instâncias 50

Prob	Gap(%) Relax Linear	Tempo Total(s) CPLEX	Gap P1 Final (%)	Tempo Projeção P1 (s)	Tempo C-&-B P1 (s)	Gap P2 Final (%)	Tempo Projeção P2 (s)	Tempo C-&-B P2 (s)	Gap P3 Final (%)	Tempo Projeção P3 (s)	Tempo C-&-B P3 (s)
50-1	0.00	140.24	0.00	3059.73	3285.76	0.00	572.09	574.27	0.00	205.18	206.8
50-2	0.57	25506.38	0.60	2781.04	10123.96	0.57	1238.47	5705.95	0.57	394.12	2983.94
50-3	0.42	13691.08	0.46	3054.28	8909.15	0.42	1533.83	3848.94	0.42	233.97	2098.42
50-4	0.16	4467.81	0.22	2949.9	4895.81	0.17	1981.78	4717.06	0.17	234.32	2328.88
50-5	1.32	56429.14	1.45	2898.9	155446.54	1.33	1354.83	24357.95	1.32	277.2	18881.74
50-6	1.47	166013.09	1.62	3036.81	69224.93	1.48	1371.23	55305.07	1.48	294.86	53866.85
50-7	2.31	******	2.51	2914.27	391809.81	2.32	1262.37	358467.5	2.36	248.65	113124.13
50-8	0.90	287433	0.97	3238.36	20906.17	0.91	1202.61	23194.15	0.91	281.03	23935.89
50-9	1.76	51235.66	1.82	3548.78	19109.27	1.77	1773.44	15212.36	1.76	289.51	10977.69
50-10	1.41	172648.44	1.53	3347.57	122975.61	1.42	1684.94	115744	1.43	314.24	66664.72

enumeração (*Branch-and-Bound*), que opera em um espaço menor de variáveis e restrições.

A contribuição da estrutura especial pode ser vista em dois momentos: primeiro, da redução do tempo de solução do subproblema na versão P2 e, segundo, provendo informação dual agregada através do uso de multicortes na versão P3. A versão P3 é, em geral, a mais agressiva das três testadas, sendo capaz das maiores diminuições no esforço computacional.

Observa-se que os limites gerados por essa técnica se mostraram, até onde foram testados, insensíveis ao tamanho das instâncias. Pesquisas futuras podem estender esses resultados para instâncias maiores. Uma vez que o tempo de computação para instâncias de 50 vértices já são elevados, não se tentou resolver problemas acima desse tamanho. Entretanto, acredita-se que um esforço de computação distribuída pode ser utilizado para lidar com instâncias ainda maiores.

4.6.2 Resultados *Cut-and-Branch* Usando Limites Superiores Pré-computados por GRASP

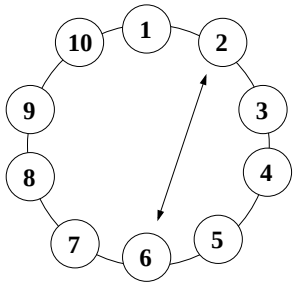
Ao serem resolvidas as instâncias do PCVM utilizando-se o CPLEX monolítico e o algoritmo de *Cut-and-Branch*, percebeu-se que o CPLEX gasta uma quantidade razoável de tempo para encontrar o primeiro limite superior. Percebeu-se também que esse limite não é necessariamente um limite justo. Uma forma de reduzir o esforço computacional para resolver as instâncias do PCVM é prover um limite superior justo para o CPLEX. Esse limite deve ser pré-computado e passado por parâmetro para o CPLEX antes de o mesmo começar o procedimento de *Branch-and-Bound*. Essa idéia já foi descrita e obteve sucesso nos capítulos sobre o Problema de Mínima Latência e sobre o Problema de Um Veículo de Entrega. Optou-se por utilizar a meta-heurística GRASP [FR95], especializada para o PCVM, como algoritmo para determinar limites superiores perto do ótimo. Esse procedimento pode ser executado de forma independente, ou usando-se computação paralela, de tal forma que não acarrete impacto no tempo de solução total.

A meta-heurística GRASP consiste em duas fases: a construção de uma solução viável inicial e um subseqüente procedimento de Busca Local. Essas duas fases são repetidas em cada iteração. Na Fase de Construção uma função gulosa e aleatória é usada para construir uma solução inicial. Essa solução é usada como solução inicial para a Fase de Busca Local. A solução final é simplesmente a melhor solução encontrada após executadas todas as iterações. O algoritmo (4.1) mostra as idéias principais do algoritmo GRASP implementado.

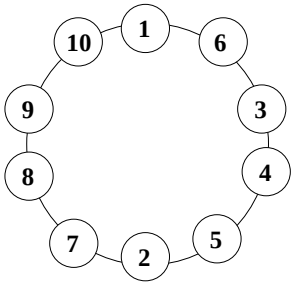
Algoritmo 4.1: GRASP

```
Fase_Construção();
Busca_Local();
if Solução_Melhor() then
    Atualize_Melhor_Solução();
    Busca_Local();
end
```

Na Fase de Construção, uma técnica gulosa e aleatória gera soluções viáveis incorporando tanto características gulosas quanto aleatórias. A solução viável é iterativamente construída, um elemento por vez. Escolheu-se algoritmo que leva em consideração a demanda d_k em cada vértice para ser inserido na solução parcial. Ao invés de escolher-se sempre o melhor elemento, isto é, o elemento com maior demanda, construiu-se uma lista restrita de candidatos (LRC) de bons elementos e escolheu-se um desses bons elementos contidos na lista. Dessa maneira, não se escolheu sempre o melhor elemento, caracterizando-se assim a aleatoriedade do algoritmo. Um parâmetro associado à lista restrita de candidatos, α_PCVM, determina o nível de aleatoriedade e gulosidade do algoritmo na Fase de Construção. Quando α_PCVM é zero tem-se uma solução totalmente gulosa. De outra forma, quando α_PCVM é um, gera-se uma solução totalmente aleatória.

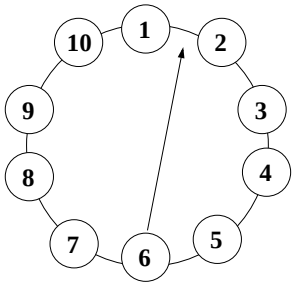


(a) Antes

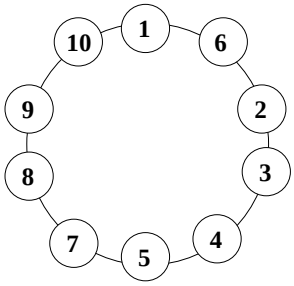


(b) Depois

Figura 4.4: Busca Local Troca 2 Vértices



(a) Antes



(b) Depois

Figura 4.5: Busca Local de Inserção

Ao invés de ser usado um algoritmo de Busca Local único, propõe-se o uso de dois métodos simples de Busca Local.

Primeiramente, um método de Busca Local baseado em trocas de posições, como mostrado na Figura (4.4), da mesma forma que foi usado no capítulo sobre o Problema de Mínima Latência, é aplicado para melhorar a solução da Fase de Construção. O segundo método de Busca Local é um algoritmo de inserção. Dada uma solução parcial oriunda da

Tabela 4.7: Resultados PCVM usando Limites Superiores (GRASP) - Instâncias 30

Problema	Tempo (s) CPLEX	Tempo (s) C-&-B	Tempo (s) Usando Limite Superior GRASP	Gap (%) Limite Superior
30-1	454.92	120.13	124.49	5.57
30-2	732.92	144.55	110.72	0.92
30-3	302.74	123.72	76.89	4.21
30-4	754.03	141.82	132.75	0.05
30-5	395.15	133.00	88.78	2.84
30-6	1653.94	390.91	394.12	0.00
30-7	459.29	172.27	175.30	0.88
30-8	80.19	76.31	59.63	5.23
30-9	287.53	289.47	84.51	0.63
30-10	955.54	231.39	134.91	0.85

Fase de Construção, essa última Busca Local funciona da seguinte maneira: cada elemento é inserido em todas as possíveis posições anteriores. Esse procedimento é repetido para todos os outros elementos da solução parcial. Um exemplo é mostrado na Figura (4.5), quando essa mesma busca local foi utilizada no algoritmo GRASP referente ao problema de Mínima Latência.

Nas Tabelas (4.7)-(4.11), o campo *Tempo (s) CPLEX* exibe o tempo total, em segundos, gasto pelo CPLEX para resolver o problema na otimalidade usando-se a formulação (4.1)-(4.18). Apresenta-se também o melhor tempo entre as três versões do Algoritmo de *Cut-and-Branch*, (*Time C-&-B (s)*). Além disso, mostra-se o tempo gasto pelo algoritmo *Cut-and-Branch* quando o mesmo utiliza os limites superiores gerados anteriormente pelo algoritmo GRASP. Esse tempo está registrado nos campos (*Tempo Total (s) usando LS GRASP*). É também apresentado o *gap* entre a melhor solução encontrada pelo algoritmo GRASP e a solução ótima, campo (*Gap Limite Superior(%)*). Todos os *gaps* são relativos ao limite superior.

Como se constata nas Tabelas (4.7)-(4.11), pré-computar limites superiores nem sempre é a melhor alternativa. Entretanto, em geral, prover limites superiores justos para o CPLEX é uma boa estratégia para melhorar o tempo de solução nessa classe de problemas. O impacto dessa estratégia no tempo de solução total parece aumentar com o tamanho do problema, assim como aconteceu nas instâncias do Problema de Mínima Latência.

Tabela 4.8: Resultados PCVM usando Limites Superiores (GRASP) - Instâncias 35

Problema	Tempo (s) CPLEX	Tempo (s) C-&-B	Tempo (s) Usando Limite Superior GRASP	Gap (%) Limite Superior
35-1	25.44	1115.84	592.61	2.01
35-2	27.34	38.58	57.59	0.00
35-3	430.93	115.88	125.12	3.02
35-4	7578.2	1092.8	628.55	0.20
35-5	1411.5	819.17	364.65	0.00
35-6	1114.09	469.1	477.69	3.55
35-7	8197.01	1617.7	736.04	0.01
35-8	1628.26	647.1	438.36	0.44
35-9	2004.55	331.92	173.94	0.00
35-10	1541.29	241.85	251.21	0.00

Tabela 4.9: Resultados PCVM usando Limites Superiores (GRASP) - Instâncias 40

Problema	Tempo (s) CPLEX	Tempo (s) C-&-B	Tempo (s) Usando Limite Superior GRASP	Gap (%) Limite Superior
40-1	67.56	67.98	96.16	0.26
40-2	9849.78	4678.16	1134.61	5.64
40-3	1505.76	328.37	334.67	0.47
40-4	49.38	82.27	188.53	1.60
40-5	975.81	186.08	184.41	1.68
40-6	1508.14	262.27	282.58	3.07
40-7	5915.15	2036.49	1610.55	3.49
40-8	3762.59	417.08	450.18	2.31
40-9	6403.87	631	612.75	5.04
40-10	7569.61	5345.49	3037.93	5.40

Tabela 4.10: Resultados PCVM usando Limites Superiores (GRASP) - Instâncias 45

Problema	Tempo (s) CPLEX	Tempo (s) C-&-B	Tempo (s) Usando Limite Superior GRASP	Gap (%) Limite Superior
45-1	89.2	125.58	193.92	0.00
45-2	46178.66	3619.1	1459.83	0.24
45-3	539.6	255.88	280.92	0.83
45-4	20313.44	4698.78	4694.31	5.65
45-5	12033.03	2093.46	1884.29	0.00
45-6	82253.25	14881.86	9604.35	3.45
45-7	167570.17	39135.17	23754.96	1.23
45-8	14150.74	3414.1	3622.23	3.92
45-9	8467.1	901.5	980.08	3.36
45-10	27299.73	8154.3	7501.77	0.86

Tabela 4.11: Resultados PCVM usando Limites Superiores (GRASP) - Instâncias 50

Problema	Tempo (s) CPLEX	Tempo (s) C-&-B	Tempo (s) Usando Limite Superior GRASP	Gap (%) Limite Superior
50-1	140.24	206.80	257.96	5.57
50-2	25506.38	2983.94	4199.94	0.58
50-3	13691.08	2098.42	1829.66	0.16
50-4	4467.81	2328.88	826.16	0.00
50-5	56429.14	18881.74	11567.00	0.77
50-6	166013.09	53866.85	25366.56	1.25
50-7	******	113124.13	113116.20	1.72
50-8	287433.00	23935.89	5780.45	0.16
50-9	51235.66	10977.69	7493.83	2.29
50-10	172648.44	66664.72	32673.69	1.41

4.6.3 Um Modo Alternativo de Computar Limites Inferiores para Instâncias Muito Grandes

Quando se lida com instâncias muito grandes, tipicamente acima de 50 vértices, o resolvedor CPLEX encontra problemas na manipulação dos dados. Esse problema é devido ao uso excessivo de memória. Essa limitação, entretanto, é dependente da máquina. Acima de 80 vértices, não é mais possível, com a máquina utilizada, usar o método da Barreira de Newton para computar os limites inferiores por programação linear. Ainda é possível utilizar o método Dual Simplex, mas esse é muito lento. Por exemplo: tentou-se computar o limite de programação linear usando o método Dual Simplex para uma instância de 100 vértices. O CPLEX gastou mais de dois dias de máquina e não conseguiu nem ao menos encontrar o limite de programação linear.

Dada a dificuldade de encontrar soluções ótimas para instâncias acima de 50 vértices, optou-se por procurar uma maneira de encontrar pelo menos *gaps* de otimalidade para essas instâncias. Para lidar com esse tipo de instâncias, resolveu-se utilizar uma estratégia de relaxação Lagrangeana. Essa escolha deve-se ao fato de essa técnica conseguir tanto limites inferiores quanto superiores para as instâncias. Uma vez que os limites inferiores de programação linear da formulação (4.2)-(4.18) são justos, tudo que se precisa é uma forma de calculá-los mais rapidamente que o resolvedor CPLEX. O algoritmo Lagrangeano proposto é um método para se obter um limite inferior justo para o problema linear inteiro misto M2, descrito na seção 4.6. A estratégia adotada faz uso da técnica de subgradiente [HWC74] [Ree93] para sucessivamente aumentar os limites inferiores obtidos pela relaxação Lagrangeana.

Para computar limites superiores justos é usado, como heurística Lagrangeana, uma versão adaptada da heurística GRASP apresentada anteriormente. A combinação dessas duas estratégias permite determinar bons *gaps* de otimalidade em um tempo razoável.

A técnica de relaxação Lagrangeana [Fis85] [Ree93] tem sido aplicada com sucesso na solução de problemas de programação linear inteira mista de grande porte. Essa técnica consiste basicamente em relaxar um conjunto de restrições complicantes e penalizar sua violação. A função objetivo é alterada adicionando-se as restrições dualizadas juntamente com seus respectivos multiplicadores de Lagrange, criando um problema chamado de Problema Lagrangeano (PLAG). Usualmente, as restrições relaxadas podem ser decompostas em muitos subproblemas fáceis de resolver. A solução dos problemas relaxados gera um limite inferior, e o objetivo é encontrar valores para os multiplicadores que gerem o maior limite inferior possível. Esse problema, que tenta maximizar os limites inferiores, é chamado de problema Lagrangeano Dual (LD).

Seja α e γ os multiplicadores associados às restrições (4.5) e (4.7). Seja $\beta_k, \forall k \in V$ e $\delta_k, \forall k \in V$ o conjunto de multiplicadores associados às restrições (4.6) e (4.8), respectivamente. Seja também $\omega_{jk}, \forall j \in V, \forall k \in V \mid j \neq k$ o conjunto de multiplicadores associados às restrições (4.9), θ os multiplicadores associados às restrições (4.10), e $\sigma_i, \forall i \in V$ e $\rho_k, \forall k \in V$ o conjunto de multiplicadores associados às restrições (4.17) e (4.18), respectivamente, o problema Lagrangeano pode ser formulado como:

$Z_{PLAG}(\alpha, \beta, \gamma, \delta, \omega, \theta, \sigma, \rho) =$

$$\min \sum_{(i,j)\in A}(b_{ij}x_{ij}+\sum_{k\in V}c_{ijk}d_k f_{ijk})+\alpha(\sum_{j\in V|j\neq o}f_{ojo}-1)$$
$$+\sum_{k\in V}\beta_k(\sum_{j\in V|j\neq o}(f_{ojk}-f_{jok})-1)+\gamma(\sum_{i\in V|i\neq o}f_{ioo}-1)$$
$$+\sum_{k\in V}\delta_k(\sum_{i\in V|i\neq k}(f_{ikk}-f_{kik})-1)$$
$$+\sum_{j\in V}\sum_{k\in V|j\neq k}\omega_{jk}(\sum_{i\in V|i\neq j,j\neq k,j\neq o}(f_{jik}-f_{ijk}))$$
$$+\theta(\sum_{i\in V,j\in V,k\in V|i\neq j}f_{ijk}-\sum_{t=1}^{n}t)$$
$$+\sum_{i\in V}\sigma_i(\sum_{k\in V,j\in V|i\neq j}f_{ijk}-\sum_{l=1}^{n}(n-l+1)p_{li})$$
$$+\sum_{k\in V}\rho_k(\sum_{i\in V,j\in V|i\neq j}f_{ijk}-\sum_{l=1}^{n}(l-1)p_{lk}) \quad (4.72)$$

sujeito a: (4.2),(4.3), (4.4), (4.11), (4.12), (4.13), (4.14) e (4.16).

Para qualquer conjunto de multiplicadores dados α, β, γ, δ, ω, θ, σ, e ρ, o problema Lagrangeano pode ser decomposto em três subproblemas de forma que: $Z_{PLAG}(\alpha,\beta,\gamma,\delta,\omega,\theta,\sigma,\rho) = Z_{PLAG_x}(\alpha,\beta,\gamma,\delta,\omega,\theta,\sigma,\rho) + Z_{PLAG_f}(\alpha,\beta,\gamma,\delta,\omega,\theta,\sigma,\rho) + Z_{PLAG_p}$

4.6.3.1 Subproblema $PLAG_x$

O subproblema $PLAG_x$ é simplesmente um Problema de Atribuição cuja função objetivo é dada por:

$$\min \sum_{(i,j)\in A}B_{ij}x_{ij} \quad (4.73)$$

sujeito a (4.2),(4.3), (4.12).

O parâmetro B_{ij} pode ser calculado pelo seguinte algoritmo:

Algoritmo 4.2: Compute B_{ij}

Data: $b_{ij}, c_{ijk}, d_k, n, G = (V, A)$

Compute todos os novos coeficientes f_{ijk} usando os valores dos multiplicadores e salve em D_{ijk};

Compute $B_{ij} = b_{ij} + \sum_{k \in V} min(0, D_{ijk})$;

É preciso salientar que os problemas de atribuição são totalmente unimodulares e que existem métodos específicos de programação linear para computar seu valor ótimo, como, por exemplo, o bem conhecido Método Húngaro ou, mais recentemente, o procedimento de Jönker e Volgenant [JV87] [JV99].

4.6.3.2 Subproblema $PLAG_f$

O subproblema $PLAG_f$ pode ser computado por inspeção. Cada variável f_{ijk} terá valor um somente se a variável correspondente $x_{ij} = 1$ e se $D_{ijk} \leq 0$. Caso contrário, f_{ijk} é igual a zero.

4.6.3.3 Subproblema $PLAG_p$

O subproblema $PLAG_p$ também é um Problema de Atribuição. Nesse caso a função objetivo é definida por:

$$\min \sum_{(i,j) \in A} CP_{ij} p_{ij} \tag{4.74}$$

sujeito a (4.13),(4.14) e (4.16).

Os valores CP_{ij} podem ser obtidos considerando-se todos os coeficientes de p_{ij} relacionados aos multiplicadores Lagrangeanos σ e ρ.

4.6.3.4 O Problema Lagrangeano Dual

O problema Lagrangeano PLAG calcula um limite inferior para o problema original M2, para qualquer conjunto de multiplicadores α, β, γ, δ, ω, θ, σ, e ρ.

Por essa razão, quanto maior for a solução do problema PLAG, melhor será o limite inferior para o problema M2. O problema de encontrar um conjunto de multiplicadores que maximizem o PLAG é chamado de Lagrangeano Dual (LD), e pode ser definido como:

$Z_{LD} = \max Z_{RL}(\alpha, \beta, \gamma, \delta, \omega, \theta, \sigma, \rho)$.

O método usado para resolver o problema LD é o método clássico do subgradiente [Fis85]. Esse método consiste em iterativamente atualizar os valores dos multiplicadores de Lagrange tal que na enésima iteração os multiplicadores sejam atualizados pelas seguintes equações:

$$\alpha^{n+1} = \alpha^n + p^n g_\alpha^n(f^n) \tag{4.75}$$
$$\beta^{n+1} = \beta^n + p^n g_\beta^n(f^n) \tag{4.76}$$
$$\gamma^{n+1} = \gamma^n + p^n g_\gamma^n(f^n) \tag{4.77}$$
$$\delta^{n+1} = \delta^n + p^n g_\delta^n(f^n) \tag{4.78}$$
$$\omega^{n+1} = \omega^n + p^n g_\omega^n(f^n) \tag{4.79}$$
$$\theta^{n+1} = \theta^n + p^n g_\theta^n(f^n) \tag{4.80}$$
$$\sigma^{n+1} = \sigma^n + p^n g_\sigma^n(f^n, p^n) \tag{4.81}$$
$$\rho^{n+1} = \rho^n + p^n g_\rho^n(f^n, p^n) \tag{4.82}$$

onde:

g_α^n, g_β^n, g_γ^n, g_δ^n, g_ω^n, g_θ^n, g_σ^n e g_ρ^n são os vetores subgradientes da função $Z_{RL}(\alpha, \beta, \gamma, \delta, \omega, \theta, \sigma, \rho)$ na enésima iteração com respeito aos multiplicadores $\alpha, \beta, \gamma, \delta, \omega, \theta, \sigma$ e ρ, respectivamente.

Os vetores subgradientes são definidos por:

$$g^n_{\alpha}(f^n) = \sum_{j \in V | j \neq o} f_{ojo} - 1 \tag{4.83}$$

$$g^n_{\beta_k}(f^n) = \sum_{j \in V | j \neq o} (f_{ojk} - f_{jok}) - 1 \qquad , \forall k \in V \tag{4.84}$$

$$g^n_{\gamma}(f^n) = \sum_{i \in V | i \neq o} f_{ioo} - 1 \tag{4.85}$$

$$g^n_{\delta_k}(f^n) = \sum_{i \in V | i \neq k} (f_{ikk} - f_{kik}) - 1 \qquad , \forall k \in V \tag{4.86}$$

$$g^n_{\omega_{kj}}(f^n) = \sum_{i \in V | i \neq j} (f_{jik} - f_{ijk}) \qquad , \forall k, j \in V \ , j \neq k, j \neq o \tag{4.87}$$

$$g^n_{\theta}(f^n) = \sum_{i,j,k \in V | i \neq j} f_{ijk} - \sum_{t=1}^{n} t \tag{4.88}$$

$$g^n_{\sigma_i}(f^n, p^n) = \sum_{k,j \in V | i \neq j} f_{ijk} - \sum_{l=1}^{n} (n - l + 1) p_{li} \quad , \forall i \in V \tag{4.89}$$

$$g^n_{\rho_k}(f^n, p^n) = \sum_{i,j \in V | i \neq j} f_{ijk} - \sum_{l=1}^{n} (l - 1) p_{lk} \qquad , \forall k \in V \tag{4.90}$$

O passo do método subgradiente é definido por:

$$p^n = \pi \frac{1.05LS - Z_{LR}(\alpha, \beta, \gamma, \delta, \omega, \theta, \sigma, \rho)}{|| \ g^n_{\alpha} \ ||^2 + || \ g^n_{\beta} \ ||^2 + || \ g^n_{\gamma} \ ||^2 + || \ g^n_{\delta} \ ||^2 + || \ g^n_{\omega} \ ||^2 + || \ g^n_{\theta} \ ||^2 + || \ g^n_{\sigma} \ ||^2 + || \ g^n_{\rho} \ ||^2} \tag{4.91}$$

onde π é um escalar definido por $0 < \pi \leq 2$. O escalar π é usado para controlar a velocidade e a convergência do método, uma vez que ele controla o tamanho do passo. O parâmetro LS é qualquer limite superior para o problema M2. Esse limite superior pode ser uma solução viável conhecida ou o máximo valor possível para a função objetivo do problema dada pela equação (4.1). O limite superior é multiplicado por 1.05 com o intuito de acelerar a convergência perto do ótimo. A função matemática $|| \ . \ ||$ define a norma Euclidiana do vetor dado. Essa equação de passo é padrão no método do subgradiente clássico [Fis85].

Os multiplicadores de Lagrange podem receber, inicialmente, o valor zero. Uma vez que a solução do problema Lagrangeano PLAG é conhecida para um conjunto de multiplicadores, os subgradientes, dados pelas equações (4.83) - (4.90) e pelo passo (4.91), podem ser estimados. O método iterage até a convergência para a solução ótima do problema Lagrangeano Dual ou até que um dado critério de parada é alcançado. O parâmetro π é iniciado com 2 e dividido por 2 a cada 200 iterações consecutivas sem melhora no limite inferior.

4.6.3.5 A Heurística Lagrangeana Baseada no método GRASP

Limites superiores para o problema M2 são estimados por uma heurística Lagrangeana desenvolvida especialmente para o problema original. A idéia principal dessa heurística consiste em tornar a solução do PLAG viável. No caso, a solução do PLAG gera uma atribuição e o objetivo é encontrar um ciclo a partir dessa atribuição. A heurística Lagrangeana proposta faz uso de uma estratégia adaptada da meta-heurística GRASP [FR95] para encontrar esse ciclo. Utilizou-se como heurística Lagrangeana quase o mesmo algoritmo GRASP apresentado na seção (4.6.2) com apenas algumas diferenças.

1. Na Fase de Construção, para cada elemento tentou-se inseri-lo usando a solução dada pelo algoritmo de subgradiente. Se o dado vértice não pode ser inserido - o elemento gera um ciclo ou o elemento já foi inserido na solução parcial - escolhe-se aleatoriamente um dos elementos com maiores demandas que ainda não foram inseridos.

2. O algoritmo GRASP somente foi chamado quando o limite inferior é melhorado ou após 40 iterações do subgradiente.

4.6.3.6 Algoritmo Lagrangeano x CPLEX Monolítico

A Tabela (4.12) mostra os resultados para o algoritmo Lagrangeano. Em todos os casos, até 50 vértices, os limites inferiores calculados pelo resolvedor CPLEX são melhores que os encontrados pelo algoritmo de relaxação Lagrangeana. Isso já era esperado pois a relaxação Lagrangeana proposta possui a propriedade da integralidade. O campo *Gap (%) Limite Superior Ótimo* mostra os *gaps* entre a solução ótima e o valor encontrado pela heurística Lagrangeana apresentada na seção (4.6.3.5). Os limites superiores não estão mais que 3% do ótimo mesmo para as instâncias maiores.

A Tabela (4.13) exibe os resultados para instâncias ainda maiores; instâncias acima de 90 vértices. O campo *Gap (%) Lagrangeano* mostra os *gaps*, em porcentagem, do limite inferior calculado pelo algoritmo Lagrangeano sobre o limite superior calculado pela Heurística Lagrangeana. O campo *Limite Inferior Lagrangeano X Limite Inferior CPLEX (%)* mostra a diferença relativa entre os limites inferiores calculados pelo algoritmo Lagrangeano e aqueles computados pelo resolvedor CPLEX utilizando o algoritmo Dual Simplex, ambos depois de 4 horas de CPU. É possível verificar que, depois de 4 horas de CPU, os limites inferiores de Lagrange são superiores aos limites inferiores do Dual Simplex. Outro ponto importante é que o Algoritmo Lagrangeano também torna disponíveis alguns limites

Tabela 4.12: Resultados da Relaxação Lagrangeana para o PCVM - Instâncias até 50

Tamanho	Gap (%) CPLEX	Gap (%) Relaxação Lagrangeana	Tempo (s) Relaxação Lagrangeana	Gap (%) Limite Inferior Ótimo	Gap (%) Limite Superior Ótimo
10	0.37	0.39	6.44	0.39	0.00
15	0.24	0.69	17.21	0.65	0.05
20	0.41	2.06	40.41	1.37	0.72
25	0.66	2.85	98.66	1.65	1.25
30	0.66	5.12	200.73	3.29	1.90
35	0.74	3.58	364.23	1.80	1.82
40	0.70	4.95	692.85	2.57	2.47
45	0.99	5.27	1174.12	2.83	2.54
50	1.07	4.50	1510.33	2.54	2.01

Tabela 4.13: Resultados Relaxação Lagrangeana (PCVM) - Instâncias Maiores que 90

Tamanho	Gap (%) Lagrangeano	Limite Inferior Lagrangeano X Limite Inferior CPLEX (%)
90	9.13	3.23
100	9.67	1.01
110	8.87	2.44
120	10.12	***
150	13.91	***

superiores que não são disponibilizados pelo resolvedor CPLEX. Acima de 120 vértices, o CPLEX trava pela falta de memória, enquanto o algoritmo Lagrangeano ainda consegue calcular limites.

Comparou-se também a evolução, no tempo, dos limites inferiores calculados por ambas as estratégias. Como se pode ver nas Figuras (4.6), (4.7), (4.8),(4.9), (4.10) e (4.11) os limites inferiores de Lagrange são superiores aos calculados pelo CPLEX para todas as instâncias testadas. Em poucos segundos, o algoritmo consegue limites inferiores que, em geral, não são obtidos pelo CPLEX, mesmo após 4 horas de computação.

4.7 Conclusão

Neste capítulo foi apresentado o Problema do Caixeiro Viajante Multiproduto (PCVM). Mostrou-se que o PCVM é uma variante mais geral do clássico Problema do Caixeiro

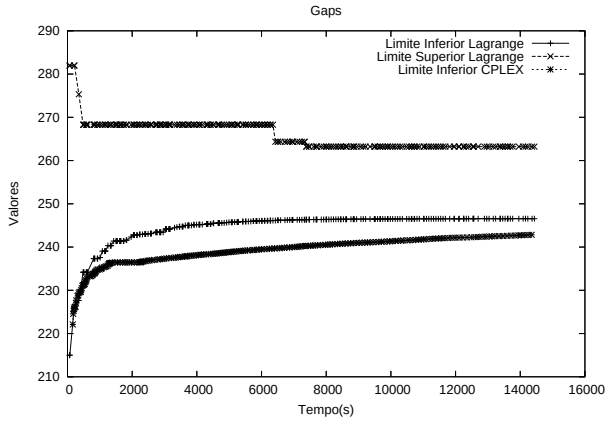


Figura 4.6: 100-1 - CPLEX x Lagrange

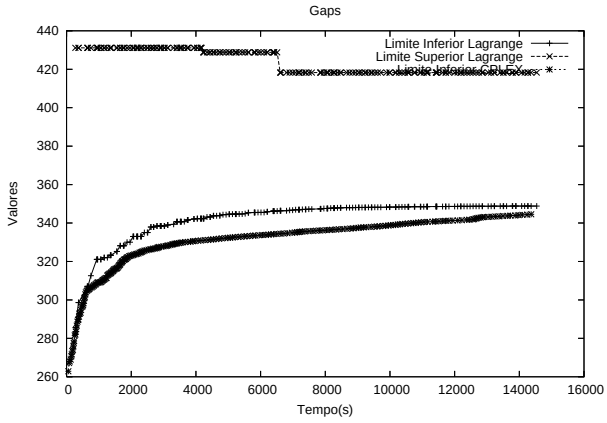


Figura 4.7: 100-2 - CPLEX x Lagrange

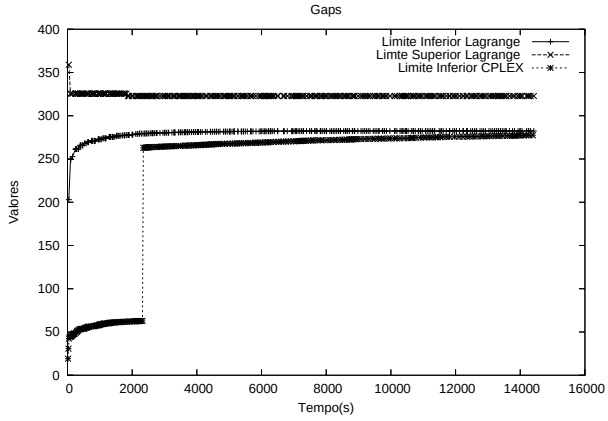


Figura 4.8: 100-3 - CPLEX x Lagrange

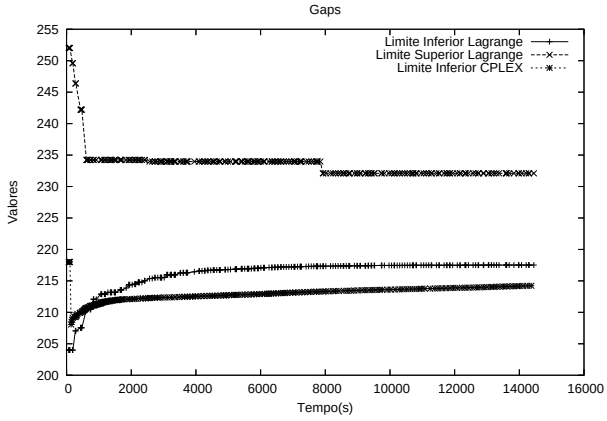


Figura 4.9: 110-1 - CPLEX x Lagrange

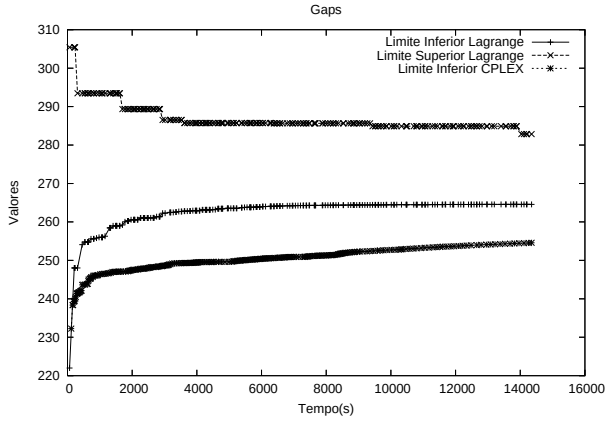


Figura 4.10: 110-2 - CPLEX x Lagrange

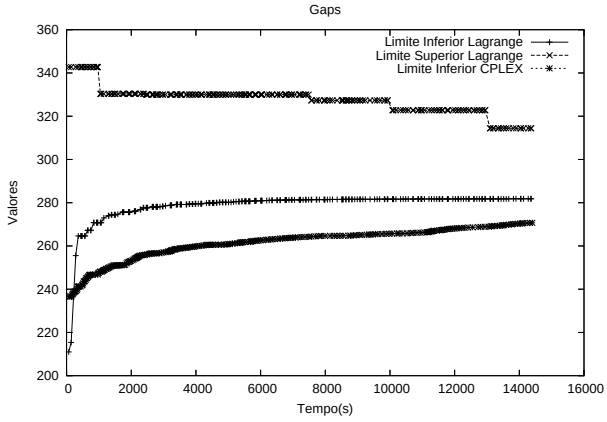


Figura 4.11: 110-3 - CPLEX x Lagrange

Viajante, que além dos custos fixos relacionados ao PCV também possui custos variáveis relativos à quantidade e qualidade dos produtos transportados. No PCV original a solução é sensível somente ao operador logístico. No PCVM ela também é sensível ao cliente. É possível relacionar o PCVM com um problema mais antigo chamado de Problema do Caixeiro Viajante Dependente do Tempo (PCVDT). Em ambos os problemas existe um custo que está relacionado à ordem de visitação dos vértices. Por esse aspecto, o PCVDT é uma generalização do PCVM, pois esse último possui uma dependência de tempo bem específica. No PCVM os arcos que são visitados posteriormente têm uma influência menor no custo devido ao total de produtos que diminuem à medida que os arcos são visitados. Já no PCVDT não existe nenhuma regra predefinida entre a ordem de visitação e o custo. Por outro lado, pode-se dizer que o PCVM é uma generalização do PCVDT. Isto é evidente quando se observa que o PCVM tem demandas heterogêneas nos vértices e essas demandas influenciam no custo total. Pelo que se sabe, essas demandas heterogêneas não acontecem em nenhuma variante do PCV, nem mesmo nessa variante mais geral PCVDT.

Para resolver o PCVM foram mostradas as semelhanças entre esse problema e o Problema de Um Veículo de Entrega. Uma formulação de programação linear inteira mista é apresentada, juntamente com um método *Cut-and-Branch* baseado na Decomposição de Benders. Esse método é utilizado para resolver, de forma mais eficiente, as instâncias geradas para o problema. Três variações do método são desenvolvidas: a primeira utiliza a forma padrão; a segunda utiliza a estrutura especial da formulação como apresentado na seção (4.5); e a terceira utiliza a versão multicorte também apresentada na seção (4.5). Pelos experimentos computacionais mostrou-se que, na maioria dos casos, a formulação multicorte é a mais agressiva, seguida da formulação que se utiliza da estrutura especial. Com essas abordagens foi possível resolver, no ótimo, instâncias de 50 vértices mais rapidamente que se utilizando a formulação original sem o método de *Cut-and-Branch*.

Para as instâncias grandes, aquelas que o CPLEX trava quando tenta resolvê-las, foi desenvolvido um método baseado na técnica de Relaxação Lagrangeana. Com esse método conseguiu-se gerar limites inferiores e superiores para instâncias de até 150 vértices. Para se ter uma idéia, com 80 vértices o CPLEX trava quando utiliza o método da Barreira para calcular o primeiro limite inferior. Com mais de 120 vértices, o CPLEX trava mesmo utilizando o método Dual Simplex que consome menos memória que o método da Barreira. Para essas grandes instâncias, é apresentada uma comparação da evolução dos limites no tempo. Pelos gráficos apresentados, até 4 horas de CPU, para as instâncias maiores que 90 vértices, é mais vantajoso utilizar o método baseado na Relaxação Lagrangeana.

Capítulo 5

Problema do Caixeiro Viajante Multiproduto Congestionado

5.1 Introdução

O Problema do Caixeiro Viajante Multiproduto (PCVM), apresentado no capítulo anterior, pode, em certos casos, ser mais aderente à realidade do que o Problema do Caixeiro Viajante (PCV) [DFJ54] original. O PCVM possui, além dos custos fixos associados à distância entre dois vértices (o mesmo presente no PCV), um custo variável que está relacionado à quantidade e à qualidade dos produtos transportados. Por exemplo: um veículo mais pesado gera mais desgaste, diminuindo a vida útil do mesmo. Esse custo variável também pode ser atribuído à segurança, em função da qual é preferível entregar os produtos mais caros primeiro para que se houver qualquer incidente os prejuízos sejam menores. Outra visão cabível e inerente ao PCVM é atribuir esse custo variável a alguma prioridade que os clientes possam ter.

Apesar do PCVM parecer ser mais aplicável para problemas reais de roteamento, esse não avalia problemas de congestionamento que redes de transporte de produtos possam ter. Em aplicações reais, esse congestionamento pode ser medido tanto no arco como no vértice (ponto de demanda).

Neste capítulo é apresentado um novo problema ainda mais genérico do que o Problema do Caixeiro Viajante Multiproduto (PCVM). Esse novo problema, chamado de Problema do Caixeiro Viajante Multiproduto Congestionado (PCVMC), possui, além dos custos fixos e variáveis do PCVM, um custo de congestionamento não-linear e capacidades nos arcos. No Problema do Caixeiro Viajante Multiproduto Congestionado (PCVMC), o custo de

carga, isto é, o custo variável, é dividido em duas partes: a primeira é o custo de carga do PCVM, já apresentado anteriormente, que é linear; a segunda é o custo relacionado ao atraso, ao tempo que é gasto a mais no processo de entrega devido ao peso do veículo. Esse custo, existente no PCVMC, é modelado por uma função não-linear. Isso se deve ao fato que esses custos aumentam de uma forma mais acentuada que os custos lineares.

Neste trabalho, o custo não-linear associado ao atraso na entrega pode ser considerado em dois lugares distintos. O primeiro é o atraso que acontece em cada arco do grafo. Esse atraso está relacionado à velocidade do veículo nas vias de transporte. Essa velocidade é função tanto do peso do veículo como da qualidade da via de transporte. Nesse caso, a capacidade do arco está relacionada à qualidade da via de transporte. Por exemplo, um veículo terrestre, mais vazio, percorrendo uma rodovia bem conservada, bem asfaltada e bem sinalizada, fluiu com maior rapidez. O segundo atraso é o que acontece em cada vértice do grafo. Esse atraso está relacionado à capacidade, em termos de mão-de-obra, de cada vértice, no processo de descarga dos produtos do veículo. A quantidade de produtos entregues, o peso desses produtos, a disposição desses produtos no veículo e o número de operadores que fazem a descarga dos produtos nos diversos pontos de demanda são os responsáveis pelo atraso nessa operação. Para esse atraso a capacidade é originalmente associada a cada vértice. Entretanto, para efeitos de simplificação, a capacidade relacionada ao segundo atraso será associada a cada arco. Esse procedimento pode ser realizado sem perda de informação. Isto é, no PCVMC, a capacidade de cada arco está associada tanto à qualidade de transporte do arco como à capacidade de processamento do vértice incidente nesse arco. Sendo assim, tem-se um problema ainda mais complexo que o PCVM. Esse problema incorpora todos os custos do problema original e, também, acrescenta um outro custo variável associado às características não-lineares que tanto o transporte como a entrega de produtos possuem.

Randazzo e Luna [RL01] desenvolveram uma formulação de programação linear para um problema de projeto de redes de acesso local. Nesse modelo, eles trabalharam com uma estrutura em árvore e com dois custos distintos que deviam ser minimizados: o custo fixo (estrutural) de instalação dos arcos e um custo variável (operacional) relativo ao fluxo dos produtos. Nesse problema nem todos os nós deveriam ser visitados. Em 2004, Miranda [Mir04] trabalhou em uma extensão do problema anterior. Nesse novo problema foi acrescentado um terceiro custo à função objetivo. Esse custo estava relacionado ao congestionamento da rede e foi utilizada uma função não-linear para representá-lo. O problema trabalhava com expansão de capacidades, da mesma forma que na tese de Ferreira [Fer03].

Em [Mir04] um modelo de fluxos não-linear e um algoritmo baseado no método de Benders foram apresentados.

Neste capítulo é estudada uma extensão do trabalho de [SMLM07] que acrescenta um custo não-linear, da mesma forma que Miranda [Mir04] fez no trabalho de Randazzo e Luna [RL01]. A diferença entre o PCVMC e o problema de Miranda [Mir04] é que esse último trabalhou com um problema de redes em árvore e com expansão de capacidades. Já o PCVMC trabalha com um circuito Hamiltoniano sem expansão de capacidades nos arcos. Além disso, como em [Sar03] [SMLM07], no PCVMC todos os vértices possuem demandas.

Este capítulo é dividido da seguinte forma: a seção (5.2) descreve formalmente o PCVMC e apresenta um modelo de programação não-linear inteira mista para esse problema; a seção (5.3) apresenta uma versão do Método de Benders Generalizado que é usado para resolver, de forma exata, instâncias do PCVMC; a seção (5.4) apresenta os resultados computacionais; a seção (5.5) apresenta uma nálise dos resultados; e, finalmente, a seção (5.6) conclui o capítulo.

5.2 Problema do Caixeiro Viajante Multiproduto Congestionado

O Problema do Caixeiro Viajante Multiproduto Congestionado (PCVMC) pode ser definido da seguinte forma: considere um grafo completo, dirigido e assimétrico $G = (V, A)$, onde V representa um conjunto de vértices e A um conjunto de arcos. Suponha também que exista uma origem 1 e, para cada vértice $k \in V$, uma demanda d_k de um produto específico k deve ser entregue durante um circuito Hamiltoniano. A demanda da origem é igual a uma unidade e representa o operador responsável pelas entregas que precisa retornar ao ponto de partida. Cada arco ainda possui uma capacidade de transporte. Nesse problema, existem custos fixos relacionados a atravessar um arco $(i, j) \in A$ e, também, custos variáveis relacionados tanto ao custo de cada produto heterogêneo a ser transportado por arcos diferentes como ao atraso na entrega devido a problemas de congestionamento. Esse congestionamento, como citado anteriormente, pode ser medido tanto no transporte como na entrega dos produtos. O objetivo do PCVMC é entregar todas as demandas minimizando a soma dos custos fixos e variáveis (lineares e não-lineares). No PCVMC são exploradas características de projeto de topologia, roteamento e capacidade dos arcos assim como [BF77] [Gav83] [MLS00].

No PCVMC, o operador logístico paga um custo fixo para atravessar um determinado arco (i, j). Esse custo está relacionado à distância e à infra-estrutura de transporte e é pago quando o operador logístico atravessa um arco, mesmo utilizando um veículo sem nenhuma carga. Esse custo fixo é o mesmo custo considerado no PCV original e no PCVM. Além do custo fixo, o PCVMC também incorpora um custo variável para cada produto que atravessa cada arco. Esse custo variável está dividido em um custo linear e um custo não-linear. O custo variável linear está associado a cada tipo de produto trafegando em cada arco, e é tanto uma função do custo fixo como também é uma função dos tipos e das quantidades de produtos transportados por aquele arco. O papel desse custo variável é estabelecer o fato que redes de transporte lidam com diferentes produtos de diferentes valores agregados que são transportados em quantidades diferentes. O segundo custo variável, o não-linear, está associado ao atraso na entrega relacionado ao peso do veículo, à qualidade da via de transporte e à capacidade de descarga em cada ponto de entrega desses produtos.

Nesse problema variante do PCV, os tipos e as quantidades de produtos que devem ser entregues e que são transportados através dos arcos influenciam de forma importante o custo total. Isso significa que os consumidores que necessitam de uma demanda maior de produtos mais valiosos ou de alto risco de transporte devem ser atendidos com uma prioridade mais alta. Outra interpretação cabível é que se as tradicionais variantes do PCV são *orientadas ao custo*, o PCVMC é também *orientado ao cliente, à capacidade da via de transporte e à capacidade de descarga dos produtos.* Por exemplo: materiais mais sensíveis podem exigir estrutura de transporte especial; produtos perecíveis podem pagar por refrigeração; clientes importantes podem ser atendidos com uma certa prioridade enquanto outros tipos de produtos e/ou clientes não requerem esse grau de atenção; vias de transporte bem sinalizadas, bem conservadas, sem congestionamento, diminuem o atraso na entrega; empresas que possuem uma boa quantidade de operadores para a descarga dos produtos tornam esse processo mais rápido, ajudando tanto as próprias empresas que necessitam logo do produto, como influenciam na entrega desses para as localidades subseqüentes.

Esse problema pode ser visto como uma generalização do PCVM. Se forem retirados o custo variável não-linear e a capacidade dos arcos, o PCVMC torna-se o PCVM. Como o PCVM engloba tanto o Problema de Um Veículo de Entrega como o Problema de Mínima Latência, é possível dizer que o PCVMC é o problema mais geral desta tese, englobando todos os outros. Dessa forma, o modelo (5.2)-(5.15), apresentado para o PCVMC, pode ser usado para todos os problemas anteriormente apresentados neste trabalho.

Para esse problema pode-se definir uma formulação de programação não-linear inteira mista com o seguinte conjunto de variáveis:

$$x_{ij} = \begin{cases} 1 & \text{se o caixeiro percorre o arco}(i,j), \\ 0 & \text{caso contrário.} \end{cases}$$

f_{ijk}: Fluxo do produto k através do arco (i,j) destinado ao vértice de demanda k.

g_{ij}: Fluxo total que passa através do arco (i,j).

e o seguinte conjunto de parâmetros

b_{ij}: custo fixo (estrutural) de percorrer o arco (i,j). Supõe-se que $b_{ij} = \delta\vartheta_{ij}$, onde ϑ_{ij} é a distância (em metros) entre i e j, e δ é o custo, por metro, para se fazer a ligação entre os vértices i e j.

c_{ijk}: custo unitário de transportar o produto k pelo arco (i,j).

C_{ij}: Capacidade do arco (i,j).

d_k: Demanda do vértice k

O modelo permite que os custos variáveis sejam dependentes tanto por produto como por arco. Se o custo variável é independente por produto tem-se $\gamma^k = \gamma \ \ \forall k \in V$. É também assumido que, para cada arco (i,j), existe uma função crescente $\tau_{ij}(g_{ij})$ do total de fluxo que passa pelo arco. A função de congestionamento é assumida como sendo separável com respeito aos arcos, e cada parcela $\tau_{ij}(g_{ij})$ é usada para avaliar a qualidade de serviço em termos de custo. Qualidade de serviço é tipicamente uma função crescente que mede o congestionamento no arco onde serviços são requisitados por usuários que competem por um determinado recurso.

Neste trabalho é usada a função não-linear sugerida por Kleinrock [Kle64] nos primórdios das redes de comunicação de computadores, vide Figura (5.1). Esse função aumenta exponencialmente à medida que o fluxo global no arco se aproxima da capacidade do mesmo e pode ser expressa, para cada arco, pela seguinte expressão:

$$f(g_{ij}) = \frac{g_{ij}}{C_{ij} - g_{ij}} \tag{5.1}$$

Obviamente, como a solução do PCVMC é um circuito Hamiltoniano, o primeiro arco selecionado para pertencer à solução, o arco que deixa o vértice de origem, deve possuir uma capacidade maior a $\sum_{k \in V} d_k$. À medida que os produtos passam pelo circuito, o custo de congestionamento vai se tornando menor. Para aumentar as características combinatórias do problema todas as capacidades são escolhidas aleatoriamente entre $0.7 \sum_{k \in V} d_k$ e $1.2 \sum_{k \in V} d_k$.

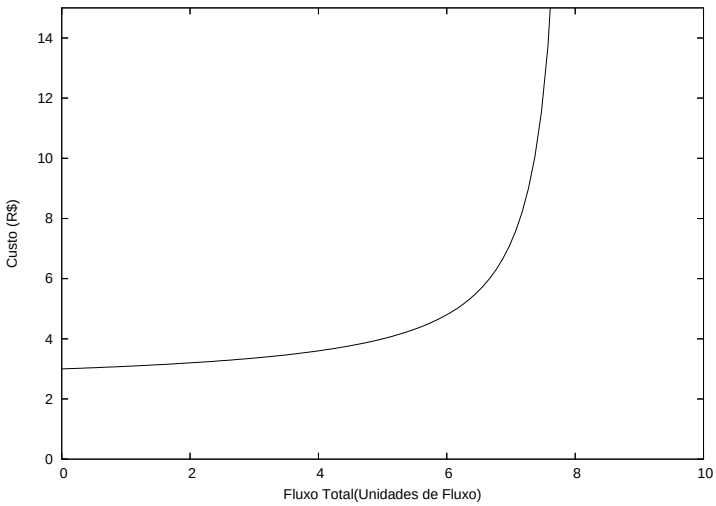


Figura 5.1: Função Kleinrock

Dessa forma, o modelo de programação não-linear inteira mista proposto para o PCVMC é:

$$\min \sum_{(i,j)\in A} [b_{ij}x_{ij} + \tau_{ij}(g_{ij}) + \sum_{k\in V} c_{ijk}f_{ijk}] \tag{5.2}$$

sujeito a

$$\sum_{i \in V | i \neq j} x_{ij} = 1 \qquad \forall j \in V \tag{5.3}$$

$$\sum_{j \in V | i \neq j} x_{ij} = 1 \qquad \forall i \in V \tag{5.4}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in V, i \neq j \tag{5.5}$$

$$f_{ijk} \leq d_k x_{ij} \qquad \forall i,j,k \in V, i \neq j \tag{5.6}$$

$$-\sum_{j \in V | j \neq 1} f_{1j1} = -1 \tag{5.7}$$

$$\sum_{j \in V | j \neq 1} (-f_{1jk} + f_{j1k}) = -d_k \qquad \forall k \in V, k \neq 1 \tag{5.8}$$

$$\sum_{i \in V | i \neq 1} f_{i11} = 1 \tag{5.9}$$

$$\sum_{i \in V | i \neq k} (f_{ikk} - f_{kik}) = d_k \qquad \forall k \in V, k \neq 1 \tag{5.10}$$

$$\sum_{i \in V | i \neq j} (f_{jik} - f_{ijk}) = 0 \qquad \forall k,j \in V, j \neq k, j \neq 1 \tag{5.11}$$

$$f_{ijk} \geq 0 \qquad \forall i,j,k \in V, i \neq j \tag{5.12}$$

$$\sum_{k \in V} f_{ijk} - g_{ij} \leq 0 \qquad \forall i,j \in V, i \neq j \tag{5.13}$$

$$g_{ij} \leq C_{ij} x_{ij} \qquad \forall i,j \in V, i \neq j \tag{5.14}$$

$$g_{ij} \geq 0 \qquad \forall i,j \in V, i \neq j \tag{5.15}$$

A função objetivo (5.2) soma os custos para todos os arcos da rede. Cada arco possui três custos. O primeiro custo refere-se ao custo fixo de percorrer um arco, independente dos produtos trafegados por esse arco. O segundo custo refere-se ao custo de congestionamento não-linear associado ao fluxo total que passa pelo arco. O terceiro custo refere-se ao custo associado com a transferência dos produtos pelos arcos do caminho, de todos os produtos, do vértice origem ao específico vértice de destino. O fato de a função objetivo ser separada por arco e por produto é a chave para a estratégia de decomposição que é usada para resolver o PCVMC.

As restrições (5.3) e (5.4) são aquelas de atribuição introduzidas em 1954 por Dantzig, Fulkerson e Johnson [DFJ54]. Essas restrições garantem que existe somente um arco saindo e um arco chegando a cada vértice. O fato que todas as variáveis x_{ij} são binárias é

assegurado pelas restrições (5.5).

As restrições que acoplam as variáveis x e f (5.6) asseguram que não existe fluxo no arco (i, j) a menos que seja pago o custo fixo b_{ij} para instalar esse arco. Essas restrições são redundantes mas são usadas na estratégia de decomposição que será mostrada posteriormente.

As equações (5.7) e (5.8) garantem que o fluxo total do produto k que é originado do vértice origem 1 é igual à demanda d_k do consumidor localizado no vértice k. A restrição (5.7) é relativa ao caso especial onde $k = 1$. De outra forma, as restrições (5.9) e (5.10) impõem que a demanda específica d_k do produto k é igual ao fluxo total desse produto que chega ao vértice k. Da mesma forma, a restrição (5.9) é relativa ao caso especial onde $k = 1$. Restrições (5.11) garantem a conservação de fluxo de qualquer produto através dos vértices que não sejam destinos finais para esse produto. O fato de o fluxo de qualquer produto por qualquer arco não ser negativo é garantido pelas equações (5.12).

As restrições (5.13) asseguram que as variáveis g_{ij} representam o fluxo total que passa pelo arco (i, j). As restrições (5.14) asseguram que a restrição de capacidade de cada arco será respeitada. E finalmente, as restrições (5.15) forçam a não-negatividade das variáveis g_{ij}.

Observe que as restrições (5.6)- (5.12) possuem dois objetivos. O primeiro é armazenar o fluxo em cada arco. Esse fluxo é usado para calcular o custo variável linear e o custo não-linear de congestionamento. O segundo objetivo é evitar subciclos ilegais. Essas restrições eliminam a necessidade de se incluir um número exponencial de restrições de ciclos presentes no modelo original do PCV [DFJ54].

Em relação à tradicional formulação de Dantzig, Fulkerson e Johnson [DFJ54], que é limitada ao espaço das variáveis x_{ij}, a inclusão das variáveis de fluxo f_{ijk} e g_{ij} aumenta de forma polinomial o número de variáveis do problema. Ao invés de trabalhar com somente uma variável binária para cada arco (i, j), a formulação (5.2)- (5.15) também opera com $\mid V + 1 \mid$ variáveis contínuas para cada arco (i, j).

5.3 Algoritmo de Benders

O método de particionamento de Benders [Ben62], também chamado método de decomposição de Benders, foi publicado em 1962 e foi inicialmente desenvolvido para resolver problemas de programação inteira mista. O sucesso computacional do método para resolver modelos de projetos de sistema de distribuição multiproduto de grande escala tem sido

confirmado desde o artigo pioneiro de Geoffrion e Graves [GG74]. Florian *et al.*. [FGB76] usaram decomposição de Benders para solucionar problemas de escalonamento de locomotivas e Richardson [Ric76] aplicou o método para resolver problemas de roteamento de rotas aéreas. Fisher e Jaikumar [FJ78] discutiram sobre as vantagens da utilização do algoritmo para roteamento de veículos. Randazzo [RL01] cita que em [LRM98] o método de particionamento de Benders foi empregado para resolver o problema de *Local Access Network Design* e teve desempenho melhor que os algoritmos de *Branch-and-Cut* e *Branch-and-Bound*. Randazzo e Luna também usaram Benders em [RL01]. Sarubbi usou Benders em [Sar03] e Miranda o usou em [Mir04]. Camargo [Cam07] utilizou esse método para encontrar, com sucesso, soluções exatas para instâncias do problema de Eixo-Raio com Atribuição Múltipla. Neste trabalho, é apresentada uma versão do método de Decomposição de Benders Generalizado especializado para o modelo (5.2)-(5.15).

5.3.1 Manipulação do Problema

O método de particionamento de Benders usa projeção como estratégia básica de manipulação do problema que é, então, seguida pelas estratégias de dualização, linearização e relaxação [Ran01]. Do ponto de vista de programação matemática, é possível conceber uma projeção do modelo (5.2)-(5.15) no espaço das variáveis topológicas x, que são as variáveis binárias do problema. A projeção resulta no seguinte problema implícito a ser resolvido no nível superior:

$$\min_{x \in X} \sum_{(i,j) \in A} b_{ij} x_{ij} + v(x) \tag{5.16}$$

onde $X = \{x \mid$ para um x fixado existem fluxos viáveis satisfazendo (5.6)-(5.12)$\}$ e onde $v(x)$ é calculado pelo seguinte problema a ser resolvido no nível inferior:

$$v(x) = \min_{f,g \in G} \sum_{(i,j) \in A} [\tau_{ij}(g_{ij}) + \sum_{k \in V} c_{ijk} f_{ijk}] \text{ sujeito a (5.6) para um } x \text{ fixado.} \tag{5.17}$$

onde $G = \{(f,g) |\ f \geq 0$,e $g \geq 0$ satisfazendo $(5.7) - (5.11)$ e $(5.13)\}$

A exigência da viabilidade dos fluxos relacionados com a variável topológica $x \in X$ implica que o conjunto de arcos para os quais $x_{ij} = 1$ forma um circuito com origem no vértice 1 e destinado a todo vértice de demanda $k \in V$. Além disso, não existe necessidade de outras restrições viáveis no domínio da projeção (5.16), e a existência de um mínimo no

subproblema (5.17) é assegurada uma vez que se está minimizando uma função convexa em um espaço não-vazio.

Uma vez que o subproblema tem uma função objetivo diferenciável e convexa e restrições lineares, as condições de Karush-Kunh-Tucker são necessárias e suficientes para garantir a otimalidade, e técnicas de dualização podem ser usadas para esse problema [Geo72]. Com um vetor de variáveis duais associado $\alpha \geq 0$, a idéia é dualizar o subproblema com respeito às restrições de acoplamento (5.6). Uma vez que não existe *gap* de dualidade, para nenhum $x \in X$, o valor ótimo do subproblema (5.17) pode ser dado por:

$$v(x) = \max_{\alpha \geq 0}[\min_{f,g \in G} \sum_{(i,j) \in A} [\tau_{ij}(g_{ij}) + \sum_{k \in V} c_{ijk} f_{ijk}] + \sum_{k \in V} \alpha_{ijk}(f_{ijk} - d_k x_{ij})] \tag{5.18}$$

ou

$$v(x) = \max_{\alpha \geq 0}[\sum_{(i,j) \in A} \sum_{k \in V} -\alpha_{ijk} d_k x_{ij} + \min_{f,g \in G} \sum_{(i,j) \in A} [\tau_{ij}(g_{ij}) + \sum_{k \in V} c_{ijk} f_{ijk}]] \tag{5.19}$$

O problema completo (5.16) é então equivalente a:

$$\min_{x \in X} \sum_{(i,j) \in A} b_{ij} x_{ij} + \max_{\alpha \geq 0}[\sum_{(i,j) \in A} \sum_{k \in V} -\alpha_{ijk} d_k x_{ij} + \min_{f,g \in G} \sum_{(i,j) \in A} [\tau_{ij}(g_{ij}) + \sum_{k \in V} (c_{ijk} + \alpha_{ijk}) f_{ijk}]] \tag{5.20}$$

ou, usando o fato que o supremo é o menor limite superior, o problema (5.2)-(5.15) é equivalente ao problema mestre:

$$\min_{x,t \in X} \sum_{(i,j) \in A} b_{ij} + t \tag{5.21}$$

sujeito a:

$$t \geq \sum_{(i,j) \in A} \sum_{k \in V} -\alpha_{ijk} d_k x_{ij} + \min_{f,g \in G} \sum_{(i,j) \in A} [\tau_{ij}(g_{ij}) + \sum_{k \in V} (c_{ijk} + \alpha_{ijk}) f_{ijk}] \ \ \forall \ \ \alpha \geq 0 \tag{5.22}$$

O método de Decomposição de Benders Generalizado resolve o problema (5.21)-(5.22) pela estratégia de relaxação, isto é, adotando algumas poucas restrições do tipo (5.22). Se em um certo circuito h, o subproblema foi resolvido por um dado circuito x^h e um vetor

ótimo de multiplicadores α^h foi recuperado, então, de (5.20), o valor ótimo $v(x^h)$ ocorre para $\alpha = \alpha^h$ e é dado por:

$$v(x^h) = -\sum_{(i,j)\in A}\sum_{k\in V}\alpha^h_{ijk}d_k x^h_{ij} + \min_{f,g\in G}\sum_{(i,j)\in A}[\tau_{ij}(g_{ij}) + \sum_{k\in V}(c_{ijk} + \alpha^h_{ijk})f_{ijk}] \tag{5.23}$$

De (5.22) segue que, associado com α^h, existe a restrição:

$$t \geq v(x^h) + \sum_{(i,j)\in A}\sum_{k\in V}\alpha^h_{ijk}d_k(x^h_{ij} - x_{ij}). \tag{5.24}$$

No nível inferior, o lado direito do problema primal (5.17) é dependente dos valores de x, mas o conjunto de soluções viáveis do problema dual correspondente é sempre o mesmo para qualquer x fixado. Usando os pontos extremos gerados desse conjunto dual é possível, no nível superior de cada ciclo de Benders, ter uma melhor subestimativa do custo operacional relacionado ao circuito gerado pelas variáveis x [Ran01]. A idéia é escolher para cada ciclo h uma solução x^h que minimize a soma do custo fixo conhecido $\sum_{(i,j)\in A} b_{ij}x_{ij}$ mais a melhor subestimativa conhecida do custo operacional relativo à topologia x^h. Será analisado agora o subproblema para prover mais detalhes das características de implementação.

5.3.2 Subproblemas

Para um circuito fixo C^h, associado com o vetor x^h, a computação do fluxo de custo mínimo $t(x^h)$ pode ser separada em uma série de problemas triviais de fluxos em redes. Seja P^h_{ok} o conjunto de arcos pertencentes ao caminho do vértice origem até o vértice de demanda k, que foi definido pelo problema mestre do ciclo h na iteração h. Para cada produto k é preciso resolver o seguinte subproblema:

$$\min_{g\geq 0, f\geq 0}\sum_{(i,j)\in A}[\tau_{ij}(g_{ij}) + \sum_{k\in V}c_{ijk}f_{ijk}] \tag{5.25}$$

sujeito às restrições de acoplamento do fluxo nos arcos (5.13) que são:

$$\sum_{k\in V} f_{ijk} - g_{ij} \leq 0 \;\; \forall (i,j) \in A$$

e as restrições (5.6)- (5.12) para um vetor binário fixado $x = x^h$.

Uma vez que $\tau_{ij}(g_{ij})$ é uma função crescente, as restrições (5.13) são satisfeitas na igualdade na solução ótima. Um fluxo único $f_{ijk} = d_k$ pode ser associado para cada arco pertencente ao caminho P^h_{1k} do vértice origem 1 até o vértice de demanda k, resultando em uma única solução ótima (f^h, g^h) associada ao circuito x^h. A construção de um vetor de multiplicadores ótimos associados pode ser realizada dualizando-se o subproblema com respeito às restrições de acoplamento (5.13). Com as variáveis duais correspondentes $\beta \geq 0$, o valor ótimo $v(x^h)$ pode ser computado como:

$$v(x^h) = \max_{\beta \geq 0} d(\beta) \tag{5.26}$$

onde a função dual $d(\beta)$ é avaliada induzindo a separabilidade natural de cada fluxo de produtos,

$$d(\beta) = \min_{g \geq 0, f \geq 0} \sum_{(i,j) \in A} [\tau_{ij}(g_{ij}) + \sum_{k \in V} c_{ijk} f_{ijk}] + \sum_{(i,j) \in A} \beta(\sum_{k \in V} f_{ijk} - g_{ij}) \tag{5.27}$$

De forma que:

$$d(\beta) = \sum_{k \in V} \min_{f_k \geq 0} \sum_{(i,j) \in A} (c_{ijk} + \beta_{ij}) f_{ijk} + \sum_{(i,j) \in A} \min_{g_{ij} \geq 0} (\tau_{ij}(g_{ij}) - \beta_{ij} g_{ij}) \tag{5.28}$$

onde cada f_k refere-se ao vetor de produtos k que é viável nas restrições correspondentes (5.6) - (5.12) para $x = x^h$.

Observa-se que, para uma solução ótima (f^h, g^h) do problema primal (5.25), uma solução ótima associada β^h para o problema dual (5.26) deve minimizar, para cada $g_{ij} \in A$, a correspondente parcela da função Lagrangeana em (5.28), o que implica:

$$\beta^h_{ij} = \tau'_{ij}(g_{ij}) \tag{5.29}$$

Como uma conseqüência de fixar esse único vetor ótimo β^h, é possível agora determinar em detalhes o par primal-dual para ser resolvido separadamente por cada produto $k \in V$ para qualquer x^h dado.

5.3.2.1 Subproblema primal para o produto k quando x=x^h

$$\min \sum_{(i,j)\in A} (c_{ijk} + \beta_{ij}) f^h_{ijk} \tag{5.30}$$

$$\text{sujeito a } f^h_{ijk} \leq d_k x^h_{ij} \quad \forall i,j,k \in V, i \neq j \tag{5.31}$$

$$-\sum_{j\in V|j\neq 1} f^h_{1j1} = -d_1 \tag{5.32}$$

$$\sum_{j\in V|j\neq 1} (-f^h_{1jk} + f^h_{j1k}) = -d_k \quad \forall k \in V,\ k \neq 1 \tag{5.33}$$

$$\sum_{i\in V|i\neq 1} f^h_{i11} = d_1 \tag{5.34}$$

$$\sum_{i,k\in V|i\neq k} (f^h_{ikk} - f^h_{kik}) = d_k \quad \forall k \in V,\ k \neq 1 \tag{5.35}$$

$$\sum_{i\in V|i\neq j} (f^h_{ijk} - f^h_{jik}) = 0 \quad \forall j,k \in V, j \neq k, k \neq 1 \tag{5.36}$$

$$f^h_{ijk} \geq 0 \quad \forall i,j,k \in V, i \neq j \tag{5.37}$$

A única e trivial solução para esse problema é:

$$f^h_{ijk} = \begin{cases} d_k & \text{se } (i,j) \in P^h_{1k} \subseteq C^h, \\ 0 & \text{caso contrário.} \end{cases}$$

5.3.2.2 Subproblema dual para o produto k quando x=x^h

O problema dual associado ao subproblema dado pela função objetivo (5.30) e pelas restrições (5.31)-(5.37) é:

$$\max_{\rho^h,\alpha^h\geq 0} d_k(\rho^h_{kk} - \rho^h_{1k} - \sum_{(i,j)\in A} x^h_{ij}\alpha^h_{ijk}) \tag{5.38}$$

$$\text{sujeito a } \rho^h_{jk} - \rho^h_{ik} - \alpha^h_{ijk} \leq c_{ijk} + \beta^h_{ij}, \ \forall k \in V \tag{5.39}$$

O problema dual tem muitas soluções viáveis, ao contrário do problema primal que tem somente uma única solução trivial. Desde que $f^h_{ijk} = d_k > 0, \forall (i,j) \in P^h_{1k} \subseteq C^h$, tem-se pelo teorema da complementaridade de folga que:

$$\rho^h_{jk} - \rho^h_{ik} - \alpha^h_{ijk} \;=\; c_{ijk} + \beta^h_{ij},\;\; \forall (i,j) \in P^h_{1k} \subset C^h \tag{5.40}$$

de uma forma que é possível construir, associada com a solução primal x^h, a seguinte solução dual viável.

$$\begin{aligned}
\rho^h_{1k} &= 0,\;\; \forall k \in V, \quad \text{para o vértice origem 1,} && (5.41)\\
\rho^h_{jk} &= \rho^h_{ik} + c_{ijk} + \beta^h_{ij},\;\; \forall (i,j) \in P^h_{1k} \subset C^h, && (5.42)\\
\alpha^h_{ijk} &= 0,\;\; \forall (i,j) \in P^h_{1k} \subset C^h, && (5.43)\\
\alpha^h_{ijk} &= \rho^h_{jk} - \rho^h_{ik} - c_{ijk} - \beta^h_{ij},\;\; \forall (i,j) \in A \backslash C^h \;\mid \rho^h_{jk} - \rho^h_{ik} > c_{ijk} + \beta^h_{ij}, && (5.44)\\
\alpha^h_{ijk} &= 0,\;\; \forall (i,j) \in A \backslash C^h \;\mid \rho^h_{jk} - \rho^h_{ik} \leq c_{ijk} + \beta^h_{ij}, && (5.45)
\end{aligned}$$

A avaliação sistemática das variáveis duais com valores que tenham sentido econômico é a chave para uma implementação eficiente. As duas séries de variáveis duais podem ser interpretadas como informação de preço. Cada variável ρ^h_{ik} representa o preço de se estabelecer a ligação $k (k \in V)$ do vértice origem 1 até o vértice i $(i \in V)$ na iteração h $(h = 1, ..., H)$. Por outro lado, cada variável α^h_{ijk} informa para cada produto k o valor de uma unidade adicional de capacidade no arco $(i,j) \in A$. A variável dual α^h_{ijk} avalia para o produto k a máxima redução no custo operacional que pode ser obtida com a inclusão do arco (i,j) na solução. No caso de sistemas de transporte a variável α^h_{ijk} também pode ser entendida como a taxa a ser paga com o uso do arco (i,j) com o intuito de manter os agentes de distribuição sem lucro. Observe que a solução dual em (5.39) representa preços espaciais nos quais não existe lucro para nenhum dos agentes que pagam o custo c_{ijk} para transportar o produto k através do arco(i,j) [RL01].

5.3.3 Problema Mestre

O objetivo do Problema Mestre é gerar um circuito C^h a cada iteração h. Na primeira iteração nenhum corte de Benders está disponível e a solução do problema mestre é a solução do PCV para esse problema.

Como citado anteriormente, as variáveis f_{ijk} presentes no modelo são responsáveis por construir o custo variável e também para evitar subciclos ilegais. Essas variáveis são usadas no subproblema, mas como inexistem no problema mestre, é preciso outra forma

para evitar subciclos ilegais. Para isso são adicionadas as variáveis y_{ij} ao problema Mestre. Essas variáveis, assim como as variáveis g_{ij} no modelo original, representam o fluxo global que passa através de um arco (i, j).

O modelo matemático do problema mestre é constituído pela seguinte função objetivo.

$$\min_{x \in X} \sum_{(i,j) \in A} b_{ij} x_{ij} + t \tag{5.46}$$

sujeito às restrições (5.3),(5.4), (5.5) e pelas restrições

$$-\sum_{j \in V | j \neq 1} y_{1j} = -\sum_{k \in V} d_k \quad \forall k \in V \tag{5.47}$$

$$\sum_{i \in V | i \neq k} y_{ik} - \sum_{k,j \in V | k \neq j} y_{kj} = 1 \quad \forall k \in V \tag{5.48}$$

$$y_{ij} \leq C_{ij} x_{ij} \quad \forall i, j \in V, i \neq j \tag{5.49}$$

$$y_{ij} \geq 0 \quad \forall i, j \in V, i \neq j \tag{5.50}$$

e pelo corte de Benders:

$$t \geq v(x^h) + \sum_{i,j \in V | i \neq j} \sum_{k \in V} (\alpha^h_{ijk} d_k x_{ij}) \;\; h = 1, .., H \tag{5.51}$$

O parâmetro h é um contador de ciclos que indica o número de cortes de Benders que foram usados até o momento. Para um dado par h e k, o valor correspondente do lado direito das restrições (5.51) provê um limite inferior no custo do fluxo que deixa o vértice de origem para o vértice de demanda k. A variável t que aparece na função objetivo (5.46) é o melhor limite inferior conhecido do custo operacional.

Devido ao sucesso das variáveis p_{ij}, definindo a ordem j de visitação do vértice i, vide seção (4.4), é possível adicionar o seguinte conjunto de restrições ao modelo original para o PCVMC:

$$\sum_{i \in V} p_{ij} = 1 \qquad \forall j \in V \tag{5.52}$$

$$\sum_{j \in V} p_{ij} = 1 \qquad \forall i \in V \tag{5.53}$$

$$p_{11} = 1 \tag{5.54}$$

$$p_{ij} \geq 0 \qquad \forall i, j \in V \tag{5.55}$$

$$\sum_{k,j \in V | i \neq j} f_{ijk} = \sum_{l=1}^{n} (n - l + 1) p_{li} \qquad \forall i \in V \tag{5.56}$$

$$\sum_{i,j \in V | i \neq j} f_{ijk} = \sum_{l=1}^{n} (l - 1) p_{lk} \qquad \forall k \in V, k \neq 1 \tag{5.57}$$

Mesmo se as restrições acima fossem adicionadas ao modelo do PCVMC (5.2)-(5.15), elas somente seriam usadas no Problema Mestre de Benders. Esse é o motivo pelo qual elas são apresentadas nesta seção. Testaram-se essas restrições acrescentando-as ao Problema Mestre e, ao contrário do que era esperado, elas tornaram o Problema Mestre ainda mais difícil de ser resolvido pelo CPLEX. A dificuldade está associada às restrições (5.49), pois quando essas foram retiradas, o Problema Mestre foi resolvido mais rapidamente, apesar de gerar circuitos Hamiltonianos sem respeitar a capacidade dos arcos. Por esse motivo, isto é, pela necessidade de incluir as restrições (5.49) no Problema Mestre e a dificuldade de resolvê-lo quando as restrições (5.52)-(5.57) são adicionadas no Problema Mestre, optou-se por não utilizá-las.

5.3.4 Algoritmo

A seguir é apresentada a implementação do método de decomposição de Benders. Os passos do algoritmo são os seguintes.

1. Encontre a solução de relaxação linear para o modelo (5.2)-(5.15) atribuindo zero ao custo de congestionamento. Armazene essa solução em SL.

2. Inicialize o Limite Inferior $LI = +\infty$. Resolva o problema inicial composto pela função objetivo (5.46) e pelas restrições (5.3)-(5.5) e (5.47)-(5.50). A resposta representa a solução do PCV para o problema. Esse valor, adicionado com o valor da solução do subproblema representado pela função objetivo (5.30) e pelas restrições (5.31)-(5.37) e calculado por um algoritmo em C++, gera o primeiro Limite Superior

LS para o problema. Se $LS - LI < \epsilon$ então **pare**. Caso contrário, calcule as variáveis duais usando as expressões (5.41)-(5.45). Adicione o corte de Benders (5.51) no problema mestre. Inicialize o contador de ciclos com zero e, se $LI < SL$, então adicione um novo corte

$$t \geq SL - LI \tag{5.58}$$

3. Resolva o problema mestre, pelo resolvedor CPLEX, composto pela função objetivo (5.46), pelas restrições (5.3)-(5.5) e (5.47)-(5.50), pelo corte de Benders (5.51) e, caso exista, pelo corte (5.58). Este último só pode ser adicionado na primeira iteração do algoritmo. A solução desse modelo modificado gera um novo Limite Inferior LI para o problema.

4. Resolva os subproblemas pelo algoritmo encontrando todos os fluxos. Um novo valor do limite superior LS é calculado. Esse LS é a adição do limite inferior com a soma de todos os custos dos fluxos calculados nos subproblemas. Se esse valor é menor que o limite superior corrente, então o limite superior é atualizado. Se $LS - LI < \epsilon$, então **pare**. Calcule as variáveis duais usando as expressões (5.41)-(5.45). Adicione o corte de Benders (5.51) ao problema mestre. Adicione uma unidade ao contador de iterações h. Vá para o passo 3.

5.4 Resultados Computacionais

Os testes foram executados em um Pentium IV com 2.4 GHz e 1 Gbyte de memória RAM. Foi utilizado o sistema operacional Linux. O algoritmo de decomposição de Benders Generalizado foi implementado em C++ usando-se a biblioteca *Concert Technology 2.0* do CPLEX 9.0. O limite de tempo total do algoritmo foi de 7200 segundos.

O campo $\mid V \mid$ da Tabela (5.1) representa o número de vértices do grafo usado como base. O campo *Max Custo Fixo* é um número inteiro que representa o máximo custo b_{ij} para a respectiva instância. O custo fixo mínimo para essa instância é igual a $(Max\ Custo\ Fixo/2)+1$. Isto é, os valores de b_{ij} relacionados ao custo de se estabelecer a ligação entre os pares de vértices estão compreendidos entre $((Max\ Custo\ Fixo/2)+1)$ e *Max Custo Fixo* assegurando assim a satisfação da propriedade de desigualdade triangular. Cada vértice $k \in V$ possui uma demanda entre 1 e o valor do parâmetro *Max Demanda* que varia entre 5 e 20 unidades. A origem tem demanda igual a 1 representando o operador

logístico que deve retornar à origem. Todas as demandas são inteiras. O custo variável linear c_{ijk} de transportar o produto k pelo arco (i, j) é calculado utilizando-se o custo fixo e outros parâmetros adicionais. Cada c_{ijk} é calculado pela seguinte expressão:

$$c_{ijk} = \xi_{ij} \cdot \eta_k$$

O parâmetro ξ_{ij} representa uma relação pseudo-aleatória entre o custo de transporte dos produtos transportados no arco (i, j) e o valor de b_{ij}. Esse parâmetro é igual para todos os produtos trafegando em um determinado arco, mas pode variar de um arco para outro. Quanto melhores forem as condições de um determinado arco, menor será o valor do parâmetro ξ correspondente. Esse parâmetro varia entre 0.5% e 2% do custo fixo b_{ij}. Cada vértice tem um parâmetro η_k, escolhido aleatoriamente entre 0.5 e 1.5. Para cada vértice um tipo específico de produtos deve ser entregue. O valor de η_k representa o valor relativo do produto, onde um η_k maior representa um produto mais valioso ou um produto perecível que precisa ser entregue mais rapidamente no vértice k. No PCVMC, o η_k pode diferenciar clientes ao invés de produtos. Mesmo que os produtos associados a dois clientes tenham o mesmo valor agregado, um cliente pode ter uma prioridade maior de entrega impactando no valor do η_k correspondente. Todas as instâncias estão associadas a grafos completos e são assimétricas. Para cada classe de valor de n, dez instâncias pseudo-aleatórias são criadas.

A principal dificuldade nessa fase do trabalho é que não estão disponíveis métodos populares para se resolver problemas de programação não-linear mista. Essa característica faz com que métodos de comparação sejam difíceis de se obter. Decidiu-se então analisar as versões linear e não-linear para efeitos de comparação. A versão linear foi obtida forçando-se o custo de congestionamento para zero. Através do seguinte exercício é possível mostrar as diferenças entre as soluções linear e não-linear.

$$Gap\ Nlinear\ geral = 100(\frac{Otimo_{naolinear} - Otimo_{linear}}{Otimo_{naolinear}}) \tag{5.59}$$

Os resultados são mostrados na Tabela (5.2). O campo *Gap Não-Linear(%)* mostra o valor calculado na expressão (5.59), isto é, o valor do *gap* não-linear. Outra medida de comparação é representada pelo campo L/NL. Essa relação mostra, para a solução ótima, quanto representa a parte linear e quanto representa a parte não-linear da solução. Essa relação é calculada por:

Tabela 5.1: Instâncias geradas para o PCVMC

Problema	$\|V\|$	Max Custo Fixo	Max Demanda	ξ
P5A	5	100	100	6
P8A	8	500	500	20
P10A	10	500	10	5
P10B	10	500	10	10
P10C	10	1000	10	5
P10D	10	1000	10	10
P10E	10	500	15	6
P10F	10	1000	15	15
P12A	12	10000	10	5
P12B	12	5000	10	5
P12C	12	1000	10	5
P12D	12	1000	10	8
P12E	12	1000	8	8
P12F	12	1000	15	5
P14A	14	10000	10	5
P14B	14	10000	7	7
P14C	14	10000	7	12
P14D	14	10000	7	10
P14E	14	10000	10	7
P16A	16	10000	5	5
P16B	16	10000	8	5
P20A	20	10000	3	5

Tabela 5.2: Resultados PCVMC

Problema	L/NL (%)	Gap Não-Linear (%)	CF/CV	Benders Ciclos	Benders Tempo (s)	Benders Gap (%)	Ciclo Ótimo	Ciclo Ótimo Tempo (s)
P5A	64.52	64.52	0.49	4	0.04	0	1	0.001
P8A	15.42	20.60	4.27	45	29.22	0	17	4.10
P10A	9.65	13.61	8.66	78	286.64	0	70	197.23
P10B	12.20	13.49	5.85	99	520.94	0	63	156.23
P10C	6.44	7.75	13.49	30	23.84	0	4	0.45
P10D	5.77	11.87	11.49	237	7237.95	0.38	-	-
P10E	10.36	17.28	7.15	98	894.01	0	32	75.79
P10F	4.79	6.64	10.13	134	366.81	0	84	107.85
P12A	0.87	0.87	46.73	6	1.97	0	1	0.15
P12B	2.76	3.68	25.77	36	94.07	0	2	0.34
P12C	6.78	6.78	12.44	46	113.91	0	1	0.06
P12D	5.54	10.53	11.95	129	7248.38	2.51	-	-
P12E	12.39	16.39	6.34	116	7336.90	4.68	-	-
P12F	6.35	11.31	11.67	224	7303.71	0.53	-	-
P14A	1.34	2.20	32.92	25	101.10	0	8	2.24
P14B	0.84	1.31	53.06	10	5.29	0	2	0.34
P14C	2.35	2.35	19.83	86	1131.70	0	1	0.84
P14D	0.76	1.40	30.26	112	1843.32	0	12	22.41
P14E	2.20	2.85	21.62	95	787.54	0	7	2.10
P16A	5.67	6.81	14.75	128	7353.18	2.31	-	-
P16B	1.92	1.92	30.34	96	1955.14	0	1	1.05
P20A	1.24	1.78	48.34	27	65.14	0	2	0.56

$$L/NL = 100(\frac{Otimo_{naolinear} - Parcela_{linear}}{Otimo_{naolinear}}) \tag{5.60}$$

onde $Parcela_{linear}$ corresponde à soma de todos os custos lineares.

Outro parâmetro de comparação na Tabela (5.2) é o campo *CF/CV*. *CF* representa o custo fixo e *CV* representa o custo variável. O custo variável é a soma do custo de se transportar os produtos pelos arcos e do custo não-linear. O parâmetro mostra a relação entre o *CF* e o *CV*, isto é, quantas vezes o custo fixo é maior que o custo variável.

Os campos *Ciclos Benders*, *Benders tempo (s)* e *Benders Gap (%)* mostram, respectivamente, o número dos ciclos de Benders, o tempo (segundos) e o *gap* do algoritmo de Benders.

O *Gap* do algoritmo de Benders é calculado por:

$$Benders\ Gap = 100(\frac{LS}{LI}) \tag{5.61}$$

O campo *Ciclo Ótimo* mostra o número de ciclos de Benders que o algoritmo usa para encontrar a solução ótima. E, finalmente, o campo *Ciclo Ótimo Tempo (s)* mostra o tempo, em segundos, que o algoritmo necessita para encontrar a solução ótima.

5.5 Análise dos Resultados

Foram realizados 22 experimentos com grafos completos e resolvidos problemas de até 20 vértices. Em 5 dessas instâncias não foi encontrada a solução ótima no tempo estipulado, mas em todos os casos é sabido que a solução ótima não está mais de 5% da solução encontrada.

Para problemas de mesmo tamanho e mesmo valor ξ, como P12A, P12B e P12C. notou-se que quanto maior é o campo *Max fixed cost* mais fácil se torna para o algoritmo de Benders encontrar a solução ótima.

Outra análise foi feita no campo *CF/CV*. Quanto maior for essa relação entre o custo fixo e os custos variáveis, mais facilmente o algoritmo resolve a instância. Isso acontece pois se o custo variável for eliminado tem-se o Problema do Caixeiro Viajante. Como o PCV é mais fácil que o Problema de Mínima Latência e que o Problema de Um Veículo de Entrega (veja seção (3.2)), quanto mais próxima for a instância do PCV, mais fácil será para resolvê-la. Por outro lado, quanto menor for a relação *CF/CV*, mais difícil será para resolvê-la. Nas 22 instâncias presentes neste capítulo, somente na primeira, a menor, o custo variável é maior que o custo fixo. No maior problema, P20A, o custo fixo é 48.34 vezes superior à soma dos custos variáveis. Foi notado que para instâncias maiores é difícil resolvê-las se o *CF/CV* for pequeno. Isso explica o porquê de serem criados problemas de dificuldades diferentes para instâncias de tamanhos diferentes. Quanto menores forem os campos *Max Demand* e ξ, a instância vai se tornando mais fácil. Isso acontece porque em cada arco uma parcela do custo variável é multiplicada pelo número de produtos que trafegam através desse arco.

Outra análise que foi realizada diz respeito ao campo *Ciclo Ótimo*. Na maioria das instâncias, Benders encontrou a solução ótima em poucos ciclos, mas na sua maioria não foi possível provar essa otimalidade rapidamente. Em 5 instâncias Benders encontrou a solução ótima no primeiro ciclo de Benders. Nesse caso, a solução do problema é a solução do PCV. Apesar de os custos serem diferentes dos custos do PCV, os arcos escolhidos na solução são os mesmos que seriam escolhidos caso fosse resolvido o PCV clássico para essa instância.

Outros pontos de comparação são os campos *L/NL* e *Gap Não-linear*. Na Tabela (5.2) é possível ver que na maioria das instâncias a parcela não-linear, *L/LN*, não é muito significativa. Além disso, quando o *L/LN* e o *Gap Não-linear* são iguais, isso significa que o custo variável (linear e não-linear) não é importante. Nesse caso a solução ótima foi encontrada no primeiro ciclo de Benders e é a solução do PCV.

5.6 Conclusão

Neste capítulo é introduzido um problema baseado no PCVM, chamado de Problema do Caixeiro Viajante Multiproduto Congestionado. Para esse problema um modelo e um algoritmo exato foram apresentados. O objetivo desse novo problema é encontrar um circuito Hamiltoniano em uma rede com três custos diferentes. O primeiro custo é o custo fixo de percorrer um arco com o veículo vazio. O segundo custo, o custo variável linear, está relacionado à quantidade e ao tipo dos produtos que passam pelo arco. Cada tipo de produto pode ter um custo variável diferente. E o terceiro custo é o custo não-linear que mede o congestionamento na rede. Esse custo é medido usando-se como base a função de Kleinrock através das seguintes informações: a capacidade do arco e o fluxo total que passa por esse arco.

Também foi mostrado que o PCVMC é um problema mais geral que engloba tanto o Problema do Caixeiro Viajante Multiproduto, O Problema de Um Veículo de Entrega, o Problema de Mínima Latência e, também, o Problema do Caixeiro Viajante clássico. Também foram mostradas as similiaridades entre o PCVMC e outros problemas de projeto não-lineares.

Para o PCVMC foram resolvidos, de forma ótima, problemas de 20 vértices usando-se um algoritmo baseado no Método de Benders Generalizado. Foi notado que quanto maior o custo fixo mais fácil torna-se para resolver o problema usando-se o algoritmo de Benders. Além disso, foi realizada uma análise dos *gaps* e mostrada a influência dos custos na solução ótima.

Capítulo 6

Conclusão

O principal objetivo deste trabalho foi apresentar e resolver, de forma exata, variações mais abrangentes do Problema do Caixeiro Viajante (PCV) clássico [DFJ54] [NW88] [ABCC06]. O PCV é um problema muito estudado mas possui uma estrutura de custo que privilegia somente o operador logístico responsável pelas entregas e, conseqüentemente, o dono do veículo ou da empresa transportadora. Nesta tese foram resolvidos, de forma exata, quatro problemas variantes do PCV. Esses problemas apresentam uma estrutura de custo mais geral e mais bem utilizável para aplicações reais. Neles, o objetivo não é somente minimizar o custo de um único operador logístico, mas, também, privilegiar os clientes com demandas de produtos mais valiosos ou clientes mais importantes, tentando maximizar a satisfação dos mesmos. Esses objetivos conflitantes de minimizar o custo operacional e maximizar a satisfação dos clientes são modelados através de problemas em que o custo da carga transportada também é um fator importante no cálculo do custo total, influenciando diretamente na ordem de visitação dos vértices.

O primeiro problema estudado foi o Problema de Mínima Latência (PML). Para ele foram apresentados três conjuntos de formulações. O primeiro, baseado em conservação de fluxo, o segundo, baseado no Problema Quadrático de Atribuição (PQA), e o terceiro, baseado no Problema do Caminho Mínimo. Com esses modelos foi possível resolver instâncias aleatórias e assimétricas de até 80 vértices, algo nunca apresentado antes na literatura. Foi possível também, resolver, de forma exata, instâncias no PML baseadas em instâncias do PQA de até 150 vértices. Resultados exatos para instâncias do PML de 150 vértices, pelo que se sabe, nunca tinham sido apresentados anteriormente. Essas instâncias baseadas no PQA foram resolvidas, no ótimo, utilizando-se a combinação de limites de programação linear muito fortes e de uma heurística GRASP especializada para o PML. Os limites de

programação linear das formulações apresentadas se mostraram extremamente fortes. Eles estiveram aproximadamente a 1.5% do ótimo até para instâncias de 80 vértices. Além disso, esses limites foram resolvidos em poucos segundos pelo método da Barreira implementado no resolvedor CPLEX. A heurística GRASP, apresentada na seção (2.7), também ofereceu resultados muito expressivos. Com ela conseguiu-se achar o ótimo inteiro para a maioria das instâncias testadas. Além disso, quando esse ótimo não foi encontrado, a solução encontrada pelo algoritmo GRASP ficou a menos de 2% do ótimo até para instâncias de 60 vértices. Esse resultado é melhor que todos os resultados heurísticos já apresentados para o PML. O limite superior calculado pela heurística GRASP também auxiliou o CPLEX a encontrar a solução ótima mais rapidamente. Notou-se que nem sempre é melhor utilizar o limite oriundo do algoritmo GRASP como primeiro limite superior para o CPLEX. Quando o primeiro limite superior calculado pelo CPLEX é melhor que o limite calculado pelo algoritmo GRASP, utilizar o limite superior calculado por esse algoritmo torna o CPLEX mais lento. É notória também a importância de se utilizar parâmetros do CPLEX adequados para resolver cada tipo de instância e para cada formulação.

O segundo problema estudado é uma generalização do PML e é conhecido como o Problema de Um Veículo de Entrega (PUVE). Para o PUVE, foram apresentados dois conjuntos de formulações. O primeiro baseado em conservação de fluxo e o segundo baseado no modelo de Bianco *et al.* [BM$^+$89]. Encontrou-se, também, um possível erro no modelo apresentado no trabalho de Bianco *et al.* [BM$^+$89]. Esse erro foi resolvido adicionando-se restrições ao modelo original que tornou o limite de programação do novo modelo mais justo para algumas instâncias. Os *gaps* de programação linear ficaram, em média, sempre menores que 3%. A heurística apresentada no capítulo sobre o PML foi alterada para resolver instâncias do PUVE e também apresentou resultados muito expressivos, chegando, no máximo, a 2.21% do ótimo para instâncias de 50 vértices. Esse resultado é mais significativo que o de Bianco *et al.* [BM$^+$89] que conseguiram resultados a 9.1% do ótimo, em média, para instâncias de 50 vértices. Da mesma forma que no PML, quando esse limite superior, encontrado pela heurística, foi passado como primeiro limite superior para o CPLEX, esse se tornou ainda mais rápido. Além disso, no PUVE, a melhora foi ainda mais significativa que a melhora no PML. As considerações sobre quando usar o limite superior oriundo do GRASP e a importância de se adequar os parâmetros do resolvedor CPLEX, válidas para o PML, são também válidas para o PUVE.

O terceiro problema estudado é uma generalização do PUVE e foi chamado de Problema do Caixeiro Viajante Multiproduto (PCVM). Para esse problema foi apresentada

uma formulação de fluxos M, e nessa formulação foram adicionadas variáveis e restrições, criando-se a formulação M2, com o intuito de melhorar o tempo de computação e também melhorar os *gaps* de relaxação linear. No primeiro modelo, o modelo M, esses *gaps* estavam perto dos 8% e chegaram, com o acrescido de restrições adicionais, modelo M2, perto de 1%. Foram propostas três versões de um algoritmo *Cut-and-Branch* especializado para o PCVM. Esse algoritmo de cortes é baseado na Decomposição de Benders. O intuito desse algoritmo é utilizar as restrições adicionais - as que melhoram o limite de programação linear - para criar cortes que tornam a formulação M tão forte quanto a formulação M2, mas com os números de variáveis e restrições próximos ao do modelo M. As três versões do algoritmo *Cut-and-Branch* se mostraram muito competitivas em comparação com o CPLEX monolítico. A última versão, a multicorte, conseguiu diminuições drásticas no esforço computacional. Para todas as versões do algoritmo *Cut-and-Branch*, a fase de projeção, isto é, a fase de geração de cortes, mostrou-se muito cara. Em compensação, o esforço na fase seguinte, isto é, o tempo computacional da enumeração, foi menor quando comparado com a versão monolítica do CPLEX. Para o PCVM também foi proposto um algoritmo Lagrangeano que apresentou resultados muito satisfatórios para instâncias acima de 90 vértices. Para essas instâncias nem o CPLEX, nem o algoritmo Lagrangeano conseguiram provar a otimalidade de nenhuma instância testada. Entretanto, o algoritmo Lagrangeano conseguiu limites inferiores e superiores que o resolvedor CPLEX não conseguiu. Para as instâncias de mais de 120 vértices, o CPLEX abortou por falta de memória e o algoritmo Lagrangeano conseguiu encontrar limites razoáveis.

O quarto e último problema estudado nesta tese é o Problema do Caixeiro Viajante Multiproduto Congestionado (PCVMC). Esse problema é uma generalização do PCVM que acrescenta custos de congestionamento não-lineares nos arcos. Para o PCVMC foi apresentada uma formulação de programação não-linear inteira mista. Essa formulação devidamente parametrizada pode resolver, de forma exata, quaisquer problemas dentre aqueles que foram apresentados nesta tese. Dessa forma, tem-se um problema mais geral que engloba tanto o PCVM, quanto o PUVE, quanto o PML. Para resolver o PCVMC foi proposto um algoritmo baseado no método de particionamento de Benders [Ben62]. Esse algoritmo conseguiu resultados exatos para instâncias de até 20 vértices. Para esse problema mostrou-se a relevância dos custos variáveis, linear e não-linear, no tempo de solução do ótimo inteiro pelo algoritmo de Benders. Instâncias do PCVMC, com um custo fixo maior, isto é, mais próximas do PCV, conseguiram resultados mais rápidos quando comparadas às instâncias com um custo variável maior, mais próximas do PUVE.

Como trabalhos futuros é possível sugerir:

- Implementação de um algoritmo *Cut-and-Branch*, baseado na formulação (2.23)-(2.40) para o PML e na formulação (2.23)-(2.40) para o PUVE, da mesma forma que foi apresentado o algoritmo *Cut-and-Branch* para o Problema do Caixeiro Viajante Multiproduto, vide seção (4.5).

- Implementação de um algoritmo *Branch-and-Cut* para o PCVM. Nesse algoritmo, os cortes gerados pela projeção das variáveis p_{ij} poderiam ser colocados e/ou retirados de todos os nós da árvore de *Branch-and-Bound* a qualquer momento, como sugerido por Balas [BB83].

- Estudar o impacto de cada Busca Local e do método de *Path-Relinking* no resultado obtido pela heurística e no tempo de computação do GRASP. Esse estudo pode ser feito tanto no algoritmo GRASP do problema de Mínima Latência, como no algoritmo GRASP presente no capítulo do Problema de Um Veículo de Entrega.

- Analisar o melhor valor para o conjunto de parâmetros $\alpha_{inicial}$ e α_{final} para a heurística GRASP.

- Implementação de um GRASP Reativo baseado no algoritmo GRASP já implementado.

- Analisar o desempenho do CPLEX, para as formulações associadas ao PML e ao PUVE quando é passado para os mesmos um limite superior não tão justo. Isto é, será que diminuir o tempo de execução da heurística GRASP, gerando um limite superior não tão perto do ótimo, irá prejudicar o procedimento de enumeração do CPLEX? Até que ponto é importante perder tempo com a execução da heurística em prol do método de *Branch-and-Bound* do CPLEX?

- Apresentar uma versão multicorte do algoritmo de Benders para o PCVMC.

- Mudar o modelo do PCVMC para que este trabalhe com um circuito bidimencional com expansão de capacidades, da mesma forma que [Mir04].

Referências Bibliográficas

[ABCC06] David L. Applegate, Robert E. Bixby, Vasek Chvátal, and William J. Cook. *The Traveling Salesman Problem:A Computational Study*. Princeton University Press, 2006.

[ACP+86] F. Afrati, S. Cosmadakis, C. H. Papadimitriou, G. Papageorgiou, and N. Papakostantinou. The complexity of the traveling repairman problem. *ITA - Informatique Theorique et Applications*, 20(1):79–87, 1986.

[AJ94] H.P. Adams and T. Johnson. Improved linear programming bounds for the quadratic assignment problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, (16):43–77, 1994.

[AK03] S. Arora and G. Karakostas. Aproximation schemes for minimum latency problems. *SIAM Journal on Computing*, 32(5):1317–1337, 2003.

[AW03] A. Archer and D.P. Williamson. Faster approximation algorithms for the minimum latency problem. pages 88–96. 14th (SODA)ACM-SIAM Symposion on Discrete Algorithms, 2003.

[Bal79] E. Balas. Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.

[Bal85] E. Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimizations problems. *SIAM Journal on Algebraic and Discrete Methods*, 6:466–486, 1985.

[BB83] E. Balas and C. Bergthaller. Benders method revisition. *Journal of Computational and Applied Mathemathics*, 9(1):3–12, 1983.

[BCC+94] A. Blum, P. Chalasani, D. Coppersmish, B. Pulleyblank, P. Raguavan, and M. Sudan. The minumum latency problem. *Proc 26th Annual ACM Symposium on Theory of Computting*, 1994. Montreal.

[Ben62] J. F. Benders. Partitioning algorithm for solving mixed integer variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[BF77] R.R. Boorstyn and H. Frank. Large scale network topological optimization. *IEEE Transactions on Communications*, COM-25:29–47, 1977.

[BL88] J.R. Birge and F.V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operations Research*, 34:384–392, 1988.

[BM+89] L. Bianco, , A. Mingozzi, S. Ricciardelli, and M. Spadoni. A new algorithm for the single vehicle delivery problem. In *Logistique- Production, Distribution, Transport*, pages 313–320. Conference on the Practice and Theory of Operations Management et 4^{es} Journées Francophones sur la Logistique et le Transportes, Dezembro 1989.

[BMR93] L. Bianco, A. Mingozzi, and S. Ricciardelli. The traveling salesman problem with cumulative costs. *Networks*, 23(2):81–91, 1993.

[BP83] E. Balas and W. Pulleyblank. The perfectly matchable subgraph polytope of a bipartite graph. *Networks*, 13:495–516, 1983.

[Cam07] R. S. Camargo. *Sistemas Eixo-Raio de Múltipla Atribuição: Modelos e Algoritmos*. PhD thesis, Universidade Federal de Minas Gerais-Departamento de Ciência da Computação, 2007.

[CE69] N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Operations Research Quaterly*, 20:309–318, 1969.

[CGRT03] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar. Paths, trees and minimum latency tours. IEEE Symposium on Foundations of Computer Science-FOCS, 2003.

[Chr79] N. Christofides. *The Traveling Salesman Problem*. Wiley Chichester, 1979.

[CMM67] R. Conway, W. Maxwell, and L. Miller. *Theory of Scheduling*. Addison-Wesley, 1967.

[CW64] G. Clark and J.W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[DFJ54] R. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410, 1954.

[DR59] G. Dantzig and J. Ramser. The truck dispaching problem. *Operations Research*, 12:81–91, 1959.

[dSDR03] M.C. de Souza, C. Duhamel, and C.C. Ribeiro. A grasp heuristic for the capacitated minimum spanning tree problem using a memory-based local search strategy. *Metaheuristics: Computer Decision-Making*, pages 627–657, 2003.

[Eij95] C. Eijl. A polyhedral approch to the delivery man problem. Memorandum COSOT 95 19, Eindhoven University of Tecnology, New York University, 1995.

[FCG84] G. Finke, A. Claus, and E. Gunn. A two-commodity network flow approach to the traveling salesman problem. *Congress Numerantium*, 41:167–178, May 1984.

[Fer03] R. P. M. Ferreira. *Modelos e Algoritmos para Problemas de Atribuição de Capacidade e Roteamento em Redes de Comunicação*. PhD thesis, Universidade Federal de Minas Gerais-Departamento de Ciência da Computação, 2003.

[FGB76] M. G. Florian, G. Guerin, and G. Bushel. The engine scheduling problem in a railway network. *INFOR Journal*, 14:121–138, 1976.

[FGG79] K.R. Fox, B. Gavish, and S. C. Graves. The time dependent traveling salesman problem and extensions. Technical report, Boeing Commercial Airplane Company, University of Rochester e Massachusetts Institute of Tecnology, 1979.

[FGG80] K. Fox, B. Gavish, and S. Graves. An n-constraint formulation of the (time-dependent) traveling salesman problem. *Operations Research*, 28:1018–1021, 1980.

[Fis85] M. L. Fisher. An applications oriented guide to Lagrangian relaxation. *Interfaces*, 15(2):10–21, 1985.

[FJ78] M. L. Fisher and R. Jaikumer. A decomposition algorithm for large scale vehicle routing. Technical report, Department of Decision Sciences, University of Pennsylvania, 1978.

[FLM93] M. Fischetti, G. Laporte, and S. Martelo. The delivery man problem and cumulative methods. *Operations Research*, 6:1055–1064, 1993.

[Fox73] K. Fox. *Production scheduling on parallel lines with dependencies*. PhD thesis, Johns Hopkins University, June 1973.

[FR95] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–135, 1995.

[Gas67] T. Gaskel. Beses for the vehicle fleet scheduling. *Operations Research Quaterly*, 18:281–295, 1967.

[Gav83] B. Gavish. Topological design of centralized computer networks. *Networks*, 12:355–377, 1983.

[GCJS57] W.M Garvin, H.W. Crandal, J.B. John, and R.A. Spellman. Aplications of linear programing in the oil industry. *Management Science*, 3:407–430, 1957.

[Geo72] A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, (10):237–260, 1972.

[GG74] A. M. Geoffrion and G. W. Graves. Multicomodity distribution system design by Benders decomposition. *Management Science*, 20:822–844, 1974.

[GG76] A.M. Geoffrion and G.W. Graves. Schedulling paralell production lines with changeover costs: Practical application of a quadratic assignment/LP approach. *Operations Research*, 24(4):595–610, 1976.

[GJT02] A. Garcia, P. Jodrá, and J. Tejel. A note on the traveling repairman problem. *Networks*, 40(1):27–31, 2002.

[GK98] M. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82:114–124, 1998.

[GL00] M. Goldbarg and H. P. L. Luna. *Otimização Combinatória e Programação Linear*. Campus, 2000.

[Glo96] Fred W Glover. Tabu search and adaptive memory programming - Advances. *Interfaces in Computer Science and Operations Research*, pages 1–75, 1996. R S Barr, R V Helgason, and J L Kennington, eds., Kluwer Academic Publishers.

[GS79] B. Gavish and S.Graves. The travelling salesman problem and related problems. Technical Report Working Paper 7906, University of Rochester and Massachusetts Institute of Technology, 1979.

[Hay67] R. Hays. The delivery problem: Rapport technique 106. Technical report, Carnegie Institute of Technology, Departament of Management Science, 1967.

[HHJ+01] P. M. Hahn, W. L. Hightower, T. A. Johnson, M. Guignard-Spielberg, and C. Roucairol. A lower bound for the quadratic assignment problem based on a level-2 reformulation-linearization technique. Technical report, The University of Pennsylvania- USA, 2001.

[HWC74] M. Held, P. Wolfe, and H. P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6:62–88, 1974.

[JM97] D. S. Johnson and L. A. McGeoch. *Local Search in Combinatorial Optimization*, chapter The Traveling Salesman Problem: a case study. John Wiley & Sons Ltd., 1997.

[JV87] R. Jönker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987.

[JV99] R. Jönker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *European Journal of Operations Research*, 116:233–234, 1999.

[KB57] T.C. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, (25):53–76, 1957.

[Kle64] L. Kleinrock. *Communications Nets: Stochastic Message Flow and Delay.* McGraw-Hill,New York, 1964.

[Law63] E.L. Lawler. The quadratic assignment problem. *Management Science*, (9):586–599, 1963.

[LM99] M. Laguna and R. Martí. GRASP and path-relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52, 1999.

[LN87] G. Laporte and Y. Nobert. Exact algorithms for the vehicle routing problem. *Annals of Discrete Mathemathics*, 31:147–184, 1987.

[LRM98] H. P. L. Luna, C. D. Randazzo, and G. R. Mateus. Experiments on optimal methods for local access network design. pages 576–583. In 6th International Conference on Telecommunication Systems: Modelling and Analysis, Vanderbilt University, Nashville, USA, 1998.

[Luc90] A. Lucena. Time-dependent traveling salesman problem - the deliveryman case. *Networks*, 20:753–763, 1990.

[Mar99] R. K. Martin. *Large Scale Linear and Integer Optimization: A Unified Approach*. Kluwer, 1999.

[MDZL07] Isabel Méndez-Dias, Paula Zabala, and Abilio Lucena. A new formulation for the traveling deliveryman problem. *Discrete Applied Mathemathics*, 2007. Aceito para publicação.

[Mir04] G. Miranda. *Localização de Servidores e Projeto de Redes com Custos de Congestionamento e Interdependência*. PhD thesis, Universidade Federal de Minas Gerais-Departamento de Ciência da Computação, 2004.

[MLS00] G.R. Mateus, H.P. Luna, and A.B. Sirihal. Heuristics for distribution network design in telecommunication. *Journal of Heuristics*, pages 131–148, 2000.

[MTZ60] C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulation of traveling salesman problems. *ACM*, 7(4):326–329, 1960.

[NW88] G. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley Interscience Series in Discrete Mathematics and Optimization, 1988.

[PQ78] J. C. Picard and M. Queyranne. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, 26:86–110, 1978.

[Ran01] C. D. Randazzo. *Algoritmos para problemas de planejamento de redes multi-níveis*. PhD thesis, Universidade Federal de Minas Gerais, Departamento de Ciência da Computação, 2001.

[Rav07] M. Gómes Ravetti. *Algorithms for a scheduling problem with parallel machines and sequence dependent setups*. PhD thesis, Universidade Federal de Minas Gerais-Departamento de Ciência da Computação, 2007.

[Ree93] C. R. Reeves. *Modern Heuristics Techniques for Combinatorial Problems*. John Wiley & Sons, 1993.

[Ric76] R. Richardson. An optimization approach to routing aircraft. *Transportation Science*, 10:52–71, 1976.

[RL01] C. D. Randazzo and H. P. L. Luna. A comparison of optimal methods for local access uncapacitated network design. *Annals of Operations Research*, 106:263–286, 2001.

[RR03] M.G.C. Resende and C.C. Ribeiro. A GRASP with path-relinking for private virtual circuit routing. *Networks*, 41(1):104–114, 2003.

[RR05] M.G.C. Resende and C.C. Ribeiro. GRASP with path-relinking: recent advantages and applications. In T. Ibaraki, K. Nonobi, and M. Yagiura, editors, *Metaheuristics: progress in real time solvers*, pages 29–63, 2005.

[Sar03] J.F.M. Sarubbi. Um modelo linear para o problema do caixeiro viajante com demandas heterogêneas. Master's thesis, Universidade Federal de Minas Gerais, Departamento de Ciência da Computação, Março 2003.

[SG76] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the Association for Computing Machinery*, 23(3):555–565, 1976.

[SL07a] J.F.M. Sarubbi. and H.P.L. Luna. A flow formulation for the minimum latency problem. INOC - Internacional Network Optimization Conference, April 2007. http://www.poms.ucl.ac.be/inoc2007/Papers/author.31/paper/paper.31.pdf.

[SL07b] J.F.M. Sarubbi and H.P.L. Luna. The multicommodity traveling salesman problem. INOC - Internacional Network Optimization Conference, April 2007. http://www.poms.ucl.ac.be/inoc2007/Papers/author.68/paper/paper.68.pdf.

[SLB91] D. Simchi-Levi and O. Berman. Minimize the total flow time of n jobs on a network. *IIE Transactions*, 23:236–244, 1991.

[SMLC07] J.F.M. Sarubbi, G. Miranda, H.P.L. Luna, and R.S. Camargo. Computing sharp lower and upper bounds for the minimum latency problem. volume 0, pages 71–77. HIS - 7th International Conference on Hybrid Intelligent Systems, Proceedings of the 7th International Conference on Hybrid Intelligent Systems. Washington, DC, USA : IEEE Computer, September 2007. 10.1109/ichis.2007.4344030.

[SMLM07] J.F.M. Sarubbi, G.R. Mateus, H.P.L. Luna, and G. Miranda. Model and algorithms for the multicommodity traveling salesman problem. volume 0, pages 113–119. HIS - 7th International Conference on Hybrid Intelligent Systems, Proceedings of the 7th International Conference on Hybrid Intelligent Systems. Washington, DC, USA : IEEE Computer, September 2007. http://dx.doi.org/10.1109/his.2007.57.

[Tsi92] J. N. Tsitsiklis. Special cases of travelling salesman and repairman problems with time windows. *Networks*, 22:263–282, 1992.

[TV02] Paolo Toth and Daniele Vigo. Models, relaxations and exacts approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123:487–512, 2002.

[Wag69] H. M. Wagner. *Principles of Operations Research - With Applications to Managerial Decisions*. Prentice-Hall, 1969.

[WHZ04] Bang Ye Wu, Zheng-Nan Huang, and Fu-Jie Zhan. Exact algorithms for the minimum latency problem. *Information Processing Letters*, 92(6):303–309, 2004.

[WS95] Russ J. Vander Wiel and Nikolaos V. Sahinidis. Heuristic bounds and test problem generation for the time-dependent traveling salesman problem. *Transportation Science*, 29(2):167–183, 1995.

Apêndice A

Problema de Mínima Latência

A.1 Introdução

Este apêndice tem como objetivo mostrar de uma forma mais detalhada os resultados das instâncias do Problema de Mínima Latência (PML) apresentadas anteriormente. No capítulo 2 foram apresentadas sempre as médias para as dez instâncias de cada conjunto. Neste apêndice são apresentados os tempos e os *gaps* para cada instância, usando cada uma das formulações analisadas.

A.2 Formulações Fluxo

Nesta seção são apresentados os resultados comparando as 4 formulações de fluxo apresentadas na seção (2.4). A formulação F1_PML é a formulação uniproduto sem as variáveis p, (2.9)-(2.17). A formulação F2_PML é a formulação uniproduto com as variáveis p, (2.9)-(2.22). A formulação F3_PML é a formulação multiproduto sem as variáveis p, (2.23)-(2.34); e a formulação F4_PML é a formulação multiproduto com as variáveis p, (2.23)-(2.40).

As Tabelas (A.1)-(A.6) possuem os seguintes campos: *Problema*, apresenta o nome da instância; campos *Gap RL F1_PML (%)*, *Gap RL F2_PML (%)*, *Gap RL F3_PML (%)*, *Gap RL F4_PML (%)*, apresentam o *gap* de relaxação linear de cada instância e para cada formulação apresentada; campos *Tempo (s) Ótimo F1_PML*, *Tempo (s) Ótimo F2_PML*, *Tempo (s) Ótimo F3_PML*, *Tempo (s) Ótimo F4_PML*, apresentam o tempo, em segundos, que cada formulação levou para provar a otimalidade da instância.

Tabela A.1: Comparação de Modelos do PML Formulações Uniproduto e Multiproduto-Instâncias 10

Problema	Gap RL F1_PML (%)	Tempo (s) Ótimo F1_PML	Gap RL F2_PML (%)	Tempo (s) Ótimo F2_PML	Gap RL F3_PML (%)	Tempo (s) Ótimo F3_PML	Gap RL F4_PML (%)	Tempo (s) Ótimo F4_PML
P10-1	10.83	0.07	3.36	0.11	0.00	0.08	0.00	0.09
P10-2	12.65	0.1	10.50	0.14	1.90	0.26	0.00	0.08
P10-3	27.82	0.36	19.43	0.43	8.55	1.47	0.00	0.11
P10-4	19.08	0.09	2.38	0.08	2.51	0.56	0.00	0.08
P10-5	29.16	0.51	14.04	0.29	7.55	1.26	0.00	0.1
P10-6	28.84	0.62	19.55	0.46	15.03	1.62	3.03	1.38
P10-7	23.43	0.46	14.40	0.29	3.44	1.09	0.00	0.09
P10-8	18.53	0.35	10.91	0.38	3.75	1.14	0.00	0.1
P10-9	25.87	0.21	14.15	0.17	4.51	1.12	0.00	0.09
P10-10	13.41	0.14	7.32	0.12	0.51	0.13	0.00	0.08
Média	20.96	0.29	11.60	0.24	4.77	0.87	0.30	0.22

Tabela A.2: Comparação de Modelos do PML Formulações Uniproduto e Multiproduto-Instâncias 12

Problema	Gap RL F1_PML (%)	Tempo (s) Ótimo F1_PML	Gap RL F2_PML (%)	Tempo (s) Ótimo F2_PML	Gap RL F3_PML (%)	Tempo (s) Ótimo F3_PML	Gap RL F4_PML (%)	Tempo (s) Ótimo F4_PML
P12-1	12.90	0.48	16.04	0.51	6.18	3.06	0.00	0.2
P12-2	14.91	0.54	6.87	0.32	2.54	2.04	0.00	0.16
P12-3	17.49	0.73	12.86	0.66	1.60	1.43	0.00	0.18
P12-4	26.77	1.02	13.38	1.1	11.32	4.98	0.78	0.5
P12-5	17.91	0.71	9.20	0.6	7.22	4.4	0.00	0.18
P12-6	13.39	0.62	10.05	0.44	3.21	3.21	0.00	0.19
P12-7	8.45	0.26	5.45	0.18	0.56	0.77	0.00	0.18
P12-8	22.32	0.44	16.83	0.59	5.93	2.03	0.00	0.21
P12-9	14.24	0.29	9.55	0.47	5.56	2.47	0.00	0.21
P12-10	21.19	0.34	13.01	0.37	7.34	2.18	0.00	0.21
Média	16.96	0.54	11.32	0.52	5.14	2.67	0.08	0.22

Tabela A.3: Comparação de Modelos do PML Formulações Uniproduto e Multiproduto-Instâncias 15

Problema	Gap RL F1_PML (%)	Tempo (s) Ótimo F1_PML	Gap RL F2_PML (%)	Tempo (s) Ótimo F2_PML	Gap RL F3_PML (%)	Tempo (s) Ótimo F3_PML	Gap RL F4_PML (%)	Tempo (s) Ótimo F4_PML
P15-1	21.67	5.07	12.22	1.10	6.14	11.02	0.00	0.49
P15-2	27.22	8.54	18.19	1.23	10.65	23.75	0.00	0.51
P15-3	33.97	25.53	13.28	1.91	13.43	65.12	0.00	0.45
P15-4	25.10	15.75	9.29	0.97	9.61	17.05	0.00	0.49
P15-5	14.81	5.98	8.17	0.69	2.49	7.56	0.00	0.47
P15-6	12.15	6.41	7.10	2.44	4.84	11.04	0.00	0.5
P15-7	18.11	12.87	6.30	0.51	3.34	5.82	0.00	0.5
P15-8	18.93	16.51	14.49	3.65	9.19	23.88	0.31	6.75
P15-9	20.20	17.82	14.39	1.43	6.44	20.41	1.03	9.29
P15-10	26.58	19.85	14.72	1.67	14.54	53.15	0.98	10.89
Média	21.87	13.43	11.82	1.56	8.07	23.28	0.23	3.04

Tabela A.4: Comparação de Modelos do PML Formulações Uniproduto e Multiproduto-Instâncias 20

Problema	Gap RL F1_PML (%)	Tempo (s) Ótimo F1_PML	Gap RL F2_PML (%)	Tempo (s) Ótimo F2_PML	Gap RL F3_PML (%)	Tempo (s) Ótimo F3_PML	Gap RL F4_PML (%)	Tempo (s) Ótimo F4_PML
P20-1	22.75	2515.48	15.48	8.68	12.27	440.54	2.25	72.99
P20-2	29.82	3794.96	18.73	12.82	14.65	1234.38	0.00	2.46
P20-3	15.17	122.44	10.81	12.66	9.22	354.25	0.00	2.19
P20-4	18.70	683.84	15.38	11.32	13.31	3026.31	0.00	2.21
P20-5	25.44	18589.03	16.56	20.31	9.45	471.9	0.77	34.82
P20-6	20.08	117.06	10.64	7.13	7.20	345.96	1.01	43.8
P20-7	13.71	318.5	10.26	21.74	12.45	1062.6	1.12	46.43
P20-8	25.05	628.05	16.13	9.31	12.68	1553.82	0.00	2.59
P20-9	25.38	1679.29	14.17	15.71	3.54	20.05	0.00	1.91
P20-10	14.88	3.44	7.79	2.45	1.85	22.00	0.00	2.17
Média	21.79	3160.96	13.60	12.21	9.66	853.18	0.52	21.15

Tabela A.5: Comparação de Modelos do PML Formulações Uniproduto e Multiproduto-Instâncias 25

Problema	Gap RL F1_PML (%)	Tempo (s) Ótimo F1_PML	Gap RL F2_PML (%)	Tempo (s) Ótimo F2_PML	Gap RL F3_PML (%)	Tempo (s) Ótimo F3_PML	Gap RL F4_PML (%)	Tempo (s) Ótimo F4_PML
P20-1	*****	*****	*****	*****	*****	*****	0.00	4.95
P20-2	*****	*****	*****	*****	*****	*****	1.35	78.64
P20-3	*****	*****	*****	*****	*****	*****	2.52	247.3
P20-4	*****	*****	*****	*****	*****	*****	0.00	6.61
P20-5	*****	*****	*****	*****	*****	*****	2.74	157.48
P20-6	*****	*****	*****	*****	*****	*****	3.92	2026.83
P20-7	*****	*****	*****	*****	*****	*****	0.00	5.75
P20-8	*****	*****	*****	*****	*****	*****	1.28	165.26
P20-9	*****	*****	*****	*****	*****	*****	2.80	356.01
P20-10	*****	*****	*****	*****	*****	*****	0.00	5.77
Média	*****	*****	*****	*****	*****	*****	1.46	305.46

Tabela A.6: Comparação de Modelos do PML Formulações Uniproduto e Multiproduto-Instâncias 30

Problema	Gap RL F1_PML (%)	Tempo (s) Ótimo F1_PML	Gap RL F2_PML (%)	Tempo (s) Ótimo F2_PML	Gap RL F3_PML (%)	Tempo (s) Ótimo F3_PML	Gap RL F4_PML (%)	Tempo (s) Ótimo F4_PML
P20-1	*****	*****	*****	*****	*****	*****	1.23	524.44
P20-2	*****	*****	*****	*****	*****	*****	2.08	1253.55
P20-3	*****	*****	*****	*****	*****	*****	4.14	1218.94
P20-4	*****	*****	*****	*****	*****	*****	3.07	2112.32
P20-5	*****	*****	*****	*****	*****	*****	3.73	3674.33
P20-6	*****	*****	*****	*****	*****	*****	1.60	1002.88
P20-7	*****	*****	*****	*****	*****	*****	1.06	1183.59
P20-8	*****	*****	*****	*****	*****	*****	1.36	1194.57
P20-9	*****	*****	*****	*****	*****	*****	2.04	824.24
P20-10	*****	*****	*****	*****	*****	*****	1.11	562.45
Média	*****	*****	*****	*****	*****	*****	2.14	1355.13

Tabela A.7: Comparação de Modelos do PML Baseados no PQA- Instâncias 10

Problema	Ótimo	Limite Inferior M1_PML	Gap RL M1_PML (%)	Tempo (s) RL M1_PML	Tempo (s) Ótimo M1_PML	Limite Inferior M2_PML	Gap RL M2_PML (%)	Tempo (s) RL M2_PML	Tempo (s) Ótimo M2_PML
P10-1	803	803	0.00	0.02	0.02	803	0.00	0.01	0.02
P10-2	332	332	0.00	0.02	0.02	332	0.00	0.01	0.01
P10-3	595	595	0.00	0.02	0.03	555	6.72	0.01	0.11
P10-4	767	767	0.00	0.02	0.02	767	0.00	0.01	0.02
P10-5	535	535	0.00	0.02	0.03	535	0.00	0.01	0.01
P10-6	561	553	1.43	0.02	0.09	553	1.43	0.01	0.03
P10-7	811	811	0.00	0.02	0.02	791	2.47	0.01	0.15
P10-8	694	694	0.00	0.02	0.03	694	0.00	0.01	0.02
P10-9	636	636	0.00	0.02	0.03	636	0.00	0.01	0.01
P10-10	410	410	0.00	0.02	0.02	410	0.00	0.01	0.01
Média			0.14	0.02	0.04		1.06	0.01	0.04

A.3 Formulações para o PML Baseadas no Problema Quadrático de Atribuição

Nesta seção são apresentados os resultados comparando as 2 formulações de fluxo baseadas no Problema Quadrático de Atribuição (PQA) e apresentadas na seção (2.5). A formulação M1_PML é a formulação baseada no PQA que apresenta o melhor *gap* de relaxação linear, (2.60)-(2.65). A formulação M2_PML é a *Formulação_Relaxada*, a formulação de três índices apresentada na seção (2.5.4.1) e composta pelas equações (2.66)-(2.74).

As Tabelas (A.7)-(A.18) possuem os seguintes campos: *Problema*, apresenta o nome da instância; campo *Ótimo*, apresenta o valor ótimo da instância; campos *Limite Inferior M1_PML* e *Limite Inferior M2_PML* apresentam o limite inferior de cada instância calculado pela relaxação linear de cada modelo; campos *Gap RL M1_PML (%)* e *Gap RL M2_PML (%)*apresentam o *gap* de relaxação linear de cada instância e para cada formulação apresentada; campos *Tempo (s) RL M1_PML*, *Tempo (s) RL M2_PML*, apresentam o tempo, em segundos, que cada formulação levou para encontrar o limite de programação linear usando cada um dos modelos; campos *Tempo (s) Ótimo M1_PML* e *Tempo (s) Ótimo M2_PML*, apresentam o tempo, em segundos, que cada formulação levou para provar a otimalidade da instância usando cada um dos modelos.

Tabela A.8: Comparação de Modelos do PML Baseados no PQA - Instâncias 12

Problema	Ótimo	Limite Inferior M1_PML	Gap RL M1_PML (%)	Tempo (s) RL M1_PML	Tempo (s) Ótimo M1_PML	Limite Inferior M2_PML	Gap RL M2_PML (%)	Tempo (s) RL M2_PML	Tempo (s) Ótimo M2_PML
P12-1	1116	1086.75	2.62	0.04	0.95	1086.75	2.62	0.02	0.08
P12-2	1140	1140	0.00	0.05	0.06	1140	0.00	0.01	0.04
P12-3	915	915	0.00	0.04	0.05	896.12	2.06	0.02	0.06
P12-4	592	592	0.00	0.04	0.06	591.5	0.08	0.02	0.05
P12-5	1178	1178	0.00	0.04	0.06	1178	0.00	0.02	0.04
P12-6	1031	1031	0.00	0.04	0.05	1031	0.00	0.01	0.04
P12-7	982	982	0.00	0.02	0.04	982	0.00	0.01	0.04
P12-8	575	554.28	3.60	0.06	1.21	541	5.91	0.02	0.06
P12-9	1032	1032	0.00	0.03	0.06	1032	0.00	0.01	0.04
P12-10	899	890	1.00	0.04	0.20	890	1.00	0.02	0.05
Média			0.72	0.04	0.27		1.17	0.02	0.05

Tabela A.9: Comparação de Modelos do PML Baseados no PQA - Instâncias 15

Problema	Ótimo	Limite Inferior M1_PML	Gap RL M1_PML (%)	Tempo (s) RL M1_PML	Tempo (s) Ótimo M1_PML	Limite Inferior M2_PML	Gap RL M2_PML (%)	Tempo (s) RL M2_PML	Tempo (s) Ótimo M2_PML
P15-1	1032	1232	0.00	0.19	0.22	1232	0.00	0.04	0.06
P15-2	643	643	0.00	0.19	0.21	642.66	0.05	0.05	0.10
P15-3	685	685	0.00	0.10	0.14	685	0.00	0.05	0.06
P15-4	781	781	0.00	0.17	0.19	781	0.00	0.05	0.06
P15-5	945	945	0.00	0.11	0.13	945	0.00	0.04	0.05
P15-6	1121	1121	0.00	0.13	0.15	1121	0.00	0.04	0.05
P15-7	944	944	0.00	0.12	0.14	944	0.00	0.04	0.06
P15-8	1273	1273	0.00	0.19	0.22	1272	0.00	0.05	0.06
P15-9	1032	1019	1.26	0.13	2.39	1009	2.23	0.05	0.31
P15-10	1170	1170	0.00	0.23	0.26	1169.33	0.06	0.05	0.11
Média			0.13	0.16	0.41		0.23	0.05	0.09

Tabela A.10: Comparação de Modelos do PML Baseados no PQA - Instâncias 20

Problema	Ótimo	Limite Inferior M1_PML	Gap RL M1_PML (%)	Tempo (s) RL M1_PML	Tempo (s) Ótimo M1_PML	Limite Inferior M2_PML	Gap RL M2_PML (%)	Tempo (s) RL M2_PML	Tempo (s) Ótimo M2_PML
P20-1	1345	1301.5	3.23	1.10	21.84	1301.5	3.23	0.14	2.49
P20-2	1083	1083	0.00	0.15	0.48	1083	0.00	0.13	0.16
P20-3	1504	1504	0.00	0.73	24.29	1502.5	0.10	0.12	0.44
P20-4	1537	1537	0.00	0.15	0.67	1537	0.00	0.13	0.16
P20-5	1379	1379	0.00	0.20	0.86	1379	0.00	0.17	0.19
P20-6	1848	1830.5	0.95	0.87	0.87	1830.5	0.95	0.14	2.64
P20-7	1018	990.6	2.69	0.73	16.4	990.1	2.74	0.14	1.04
P20-8	1789	1787	0.11	0.73	21.8	1787	0.11	0.14	0.55
P20-9	921	921	0.00	0.20	1.76	921	0.00	0.17	0.18
P20-10	1584	1584	0.00	0.53	0.54	1583.66	0.02	0.14	0.29
Média			0.70	0.54	8.95		0.72	0.14	0.81

Tabela A.11: Comparação de Modelos do PML Baseados no PQA - Instâncias 25

Problema	Ótimo	Limite Inferior M1_PML	Gap RL M1_PML (%)	Tempo (s) RL M1_PML	Tempo (s) Ótimo M1_PML	Limite Inferior M2_PML	Gap RL M2_PML (%)	Tempo (s) RL M2_PML	Tempo (s) Ótimo M2_PML
P25-1	1207	1207	0.00	2.21	2.34	1207	0.00	0.14	0.34
P25-2	1255	1248.73	0.50	3.16	40.82	1245.81	0.73	0.13	1.76
P25-3	1987	1979.5	0.38	3.50	36.02	1979.5	0.38	0.12	0.93
P25-4	1482	1482	0.00	3.33	3.47	1482	0.00	0.13	0.36
P25-5	2295	2268.75	1.14	3.07	78.63	2224.43	3.07	0.17	9.83
P25-6	2349	2251.05	4.17	4.19	228.63	2250.62	4.19	0.14	10.6
P25-7	1845	1845	0.00	2.39	2.52	1845	0.00	0.14	0.38
P25-8	1998	1993.73	0.21	4.16	34.2	1970	1.40	0.14	0.95
P25-9	1687	1685.31	1.70	4.21	108.18	1658.01	1.72	0.17	4.46
P25-10	1830	1821.5	0.46	3.50	25.87	1809.5	1.12	0.14	0.76
Média			0.86	3.37	56.09		1.26	0.14	3.04

Tabela A.12: Comparação de Modelos do PML Baseados no PQA - Instâncias 30

Problema	Ótimo	Limite Inferior M1_PML	Gap RL M1_PML (%)	Tempo (s) RL M1_PML	Tempo (s) Ótimo M1_PML	Limite Inferior M2_PML	Gap RL M2_PML (%)	Tempo (s) RL M2_PML	Tempo (s) Ótimo M2_PML
P30-1	2119	2116.04	0.14	9	65.95	2088.55	1.44	0.71	5.67
P30-2	1737	1715	1.27	9.54	178.13	1715	1.27	0.72	3.25
P30-3	2098	2037.66	2.88	11.01	191	2037.28	2.89	0.73	7.68
P30-4	2259	2255.13	0.17	12.85	62.47	2181.28	3.44	0.76	38.35
P30-5	2414	2370.56	1.80	9.97	309.43	2370.56	1.80	0.68	18.4
P30-6	3333	3302.59	0.91	12.20	147.30	3285.73	1.42	0.70	18.47
P30-7	1847	1847	0.00	8.17	10.85	1830.7	0.88	0.68	3.72
P30-8	2082	2064.11	0.86	10.59	129.12	2017.20	3.11	0.72	7.14
P30-9	1807	1802.75	1.24	10.61	110.02	1802.75	0.78	0.66	4.57
P30-10	2389	2358.25	1.29	10.23	158.17	2352.86	1.51	0.77	6.10
Média			0.95	10.42	136.24		1.85	0.71	11.34

Tabela A.13: Comparação de Modelos do PML Baseados no PQA - Instâncias 35

Problema	Ótimo	Limite Inferior M1_PML	Gap RL M1_PML (%)	Tempo (s) RL M1_PML	Tempo (s) Ótimo M1_PML	Limite Inferior M2_PML	Gap RL M2_PML (%)	Tempo (s) RL M2_PML	Tempo (s) Ótimo M2_PML
P35-1	2525	2485.99	1.44	29.55	410.64	2485.99	1.54	1.50	30.26
P35-2	2287	2223.90	2.76	37.09	4129.67	2223.90	2.76	1.51	71.90
P35-3	2291	2271	0.65	22.79	221.03	2271	0.87	1.44	4.21
P35-4	2401	2401	0.00	32.91	33.3	2401	0.00	1.33	1.49
P35-5	1994	1982	0.49	38.39	181.56	2102.51	0.60	1.53	3.25
P35-6	2554	2472.14	3.15	38.67	669.28	1982	3.21	1.54	44.42
P35-7	2168	2144.58	1.01	29.59	348.08	2472.14	1.08	1.49	34.82
P35-8	2204	2154.39	1.90	24.54	336.17	2144.58	2.25	1.40	16.81
P35-9	2435	2309.80	0.68	43.33	385.01	2154.39	5.14	1.63	152.10
P35-10	2670	2632.05	1.08	32.86	395.96	2632.05	1.42	1.37	18.86
Média			1.32	32.57	711.07		1.89	1.47	37.77

Tabela A.14: Comparação de Modelos do PML Baseados no PQA- Instâncias 40

Problema	Ótimo	Limite Inferior M1_PML	Gap RL M1_PML (%)	Tempo (s) RL M1_PML	Tempo (s) Ótimo M1_PML	Limite Inferior M2_PML	Gap RL M2_PML (%)	Tempo (s) RL M2_PML	Tempo (s) Ótimo M2_PML
P40-1	3027	2952.16	2.47	97.45	****	2946.89	2.65	2.51	151.15
P40-2	2467	2444.68	0.90	89.95	****	2421.98	1.82	2.29	32.01
P40-3	3168	3137.41	0.97	108.21	****	3105.94	1.96	2.24	200.93
P40-4	2197	2190.85	0.28	76.91	****	2187.96	0.41	2.42	19.63
P40-5	3745	3643.81	2.70	67.47	****	3639.84	2.81	2.41	108.16
P40-6	2757	2728.05	1.05	65.94	****	2689.34	2.45	2.41	110.23
P40-7	4209	4148.69	1.43	94.88	****	4146.91	1.48	2.41	82.41
P40-8	3185	3184.33	0.02	74.6	****	3184.33	0.02	2.05	3.82
P40-9	2685	2645.5	1.47	79.23	****	2645.5	1.47	2.34	72.77
P40-10	3201	3163.9	1.16	99.25	****	3152.78	1.51	2.37	62.74
Média			1.25	85.39	****		1.66	2.35	84.38

Tabela A.15: Comparação de Modelos do PML Baseados no PQA- Instâncias 45

Problema	Ótimo	Limite Inferior M1_PML	Gap RL M1_PML (%)	Tempo (s) RL M1_PML	Tempo (s) Ótimo M1_PML	Limite Inferior M2_PML	Gap RL M2_PML (%)	Tempo (s) RL M2_PML	Tempo (s) Ótimo M2_PML
P45-1	3382	3300.03	2.42	133.11	****	3283.86	2.91	3.52	602.79
P45-2	2608	2599.88	0.31	106.31	****	2575.4	1.25	3.63	51.91
P45-3	3009	3009	0.00	125.84	****	2990.9	0.60	4.35	21.46
P45-4	2885	2860.23	0.86	134.37	****	2843.25	1.45	4.03	319.76
P45-5	3535	3494.36	1.15	149.21	****	3437.71	2.75	3.49	665.85
P45-6	2841	2797.61	1.53	119.43	****	2793.32	1.68	4.06	149.1
P45-7	3406	3400	0.18	132.98	****	3372.54	0.98	3.92	252.89
P45-8	3155	3127.46	0.87	130.20	****	3122.38	1.03	4.25	79.16
P45-9	3484	3415.8	1.96	137.83	****	3399.53	2.42	4.08	330.5
P45-10	3522	3467.5	1.55	132.66	****	3467.5	1.55	3.71	159.38
Média			1.10	130.19	****		1.66	3.90	263.28

Tabela A.16: Comparação de Modelos do PML Baseados no PQA- Instâncias 50

Problema	Ótimo	Limite Inferior M1_PML	Gap RL M1_PML (%)	Tempo (s) RL M1_PML	Tempo (s) Ótimo M1_PML	Limite Inferior M2_PML	Gap RL M2_PML (%)	Tempo (s) RL M2_PML	Tempo (s) Ótimo M2_PML
P50-1	3141	3131.89	0.29	204.25	****	3131.63	0.30	5.58	30.98
P50-2	3501	3430.55	2.01	271.22	****	3420.25	2.31	5.73	5468.17
P50-3	3957	3907.22	1.26	230.54	****	3892.25	1.64	5.47	2443.27
P50-4	3781	3733.43	1.26	216.93	****	3647.04	3.54	5.67	5107.16
P50-5	4490	4432.84	1.27	249.82	****	4432.21	1.29	6.17	630.12
P50-6	3447	3424.08	0.66	206.55	****	3424.08	0.66	5.6	83.73
P50-7	3658	3626.58	0.86	255.78	****	3607.39	1.38	5.29	722.75
P50-8	3769	3746.5	0.60	213.58	****	3746.5	0.60	6.64	30.91
P50-9	3024	2994.75	0.97	167.82	****	2994.75	0.97	5.34	233.51
P50-10	3718	3619.41	2.65	202.72	****	3605.7	3.02	5.37	2175.7
Média			1.18	221.92	****		1.57	5.68	1692.63

Tabela A.17: Comparação de Modelos do PML Baseados no PQA- Instâncias 55

Problema	Ótimo	Limite Inferior M1_PML	Gap RL M1_PML (%)	Tempo (s) RL M1_PML	Tempo (s) Ótimo M1_PML	Limite Inferior M2_PML	Gap RL M2_PML (%)	Tempo (s) RL M2_PML	Tempo (s) Ótimo M2_PML
P55-1	4229	4137.48	2.16	364.02	****	4134.95	2.22	7.77	2566.86
P55-2	4159	4152.60	0.15	351.54	****	4067.88	2.19	8.47	783.17
P55-3	3412	3340.97	2.08	383.77	****	3340.59	2.09	7.64	4146.93
P55-4	3823	3722.98	2.62	398.61	****	3713.94	2.85	7.46	5468.22
P55-5	4308	4178.87	3.00	388.22	****	4129.61	4.14	7.98	32591.09
P55-6	3826	3767.35	1.53	388.22	****	3763.01	1.65	9.04	1982.98
P55-7	3796	3772.42	0.62	380.41	****	3772.1	0.63	9.13	129.54
P55-8	3720	3648.82	1.91	377.78	****	3648.51	1.92	7.96	790.6
P55-9	4011	3972.04	0.97	331.58	****	3970.88	1.00	8.65	310.45
P55-10	4520	4423.54	2.13	306.95	****	4422.81	2.15	9.32	760.62
Média			1.72	367.11	****		2.08	8.34	4953.04

Tabela A.18: Comparação de Modelos do PML Baseados no PQA - Instâncias 60

Problema	Ótimo	Limite Inferior M1_PML	Gap RL M1_PML (%)	Tempo (s) RL M1_PML	Tempo (s) Ótimo M1_PML	Limite Inferior M2_PML	Gap RL M2_PML (%)	Tempo (s) RL M2_PML	Tempo (s) Ótimo M2_PML
P60-1	4833	4826.63	0.13	590.14	****	4790.18	0.89	11.20	1176.75
P60-2	4945	4914.98	0.61	621.28	****	4853.57	1.85	11.37	17790.2
P60-3	3851	3818.27	0.85	595.74	****	3793.03	1.51	11.56	13653.13
P60-4	4811	4793.23	0.37	680.38	****	4758.31	1.10	12.05	3637.69
P60-5	4483	4424.15	1.31	547.33	****	4418.84	1.43	11.53	10787.5
P60-6	4564	4493.25	1.55	592.9	****	4462.51	2.22	11.06	59682.29
P60-7	4251	4207.64	1.02	620.29	****	4207.41	1.03	12.30	1342.98
P60-8	4704	4626.79	1.64	656.08	****	4620.48	1.78	11.57	32543.5
P60-9	4712	4617.68	2.00	557.8	****	4612.28	2.12	12.20	29608.56
P60-10	4376	4333.58	0.97	682.36	****	4332.29	1.00	11.66	5657.83
Média			1.05	614.43	****		1.49	11.65	17558.04

A.4 Formulações para o PML Baseadas no Problema do Caminho Mínimo

Nesta seção são apresentados os resultados comparando as 2 formulações de fluxo baseadas no Problema do Caminho Mínimo e apresentadas na seção (2.6). A formulação M3_PML é a formulação (2.75)-(2.81) que se baseia no grafo multinível G'. A formulação M4_PML é a formulação apresentada por Bianco *et al.* [BM+89].

As Tabelas (A.19)-(A.35) apresentam somente os tempos para provar a otimalidade da instância usando as formulações M3_PML e M4_PML juntamente com o limite superior encontrado pela heurística GRASP apresentada na seção (2.7). Nessas tabelas não foram apresentados os limites inferiores e tampouco os *gaps* de relaxação linear das instâncias pois esses valores são os mesmos das Tabelas (A.7) - (A.18).

As Tabelas (A.19)-(A.35) possuem os seguintes campos: *Problema*, apresenta a instância analisada; *Tempo (s) M3_PML Padrão* e *Tempo (s) M4_PML Padrão* mostram o tempo para se provar a otimalidade da instância usando o CPLEX com os parâmetros padrões, exceto o algoritmo de cálculo de limite inferior que, para todas as instâncias, foi utilizado o Método da Barreira; campos *Tempo (s) M3_PML Ênfase* e *Tempo (s) M4_PML Ênfase* apresentam o tempo, em segundos, para se provar a otimalidade da instância alterando a ênfase do CPLEX; para o modelo M3_PML ênfase do CPLEX ficou na otimalidade da solução, já para o modelo M4_PML a ênfase do CPLEX foi na integralidade da solução; campos *Tempo (s) M3_PML Padrão GRASP*, campos *Tempo (s) M4_PML Padrão GRASP* são os mesmos que os *Tempo (s) M3_PML Padrão* e *Tempo (s) M4_PML Padrão* mas utilizando como primeiro limite superior para o CPLEX o valor calculado pela heurística GRASP; os campos *Tempo (s) M3_PML Ênfase GRASP* e *Tempo (s) M4_PML Ênfase GRASP* são os mesmos que os campos *Tempo (s) M4_PML Ênfase* e *Tempo (s) M4_PML Ênfase* mas utilizando como primeiro limite superior para o CPLEX o valor calculado pela heurística GRASP.

A.5 Resultados Heurística GRASP

Nesta seção são apresentados os resultados da heurística GRASP apresentada na seção (2.7).

As Tabelas (A.19)-(A.35) apresentam somente os tempos para provar a otimalidade da instância usando as formulações M3_PML e M4_PML juntamente com o limite superior

Tabela A.19: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 10

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P10-1	0.01	0.01	0.02	0.02	0.02	0.02	0.02	0.02
P10-2	0.02	0.02	0.02	0.02	0.02	0.02	0.03	0.03
P10-3	0.05	0.05	0.03	0.03	0.06	0.07	0.02	0.03
P10-4	0.01	0.02	0.02	0.02	0.01	0.02	0.02	0.03
P10-5	0.01	0.02	0.03	0.03	0.02	0.02	0.03	0.03
P10-6	0.04	0.04	0.02	0.02	0.04	0.05	0.03	0.02
P10-7	0.06	0.06	0.02	0.03	0.04	0.05	0.02	0.02
P10-8	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
P10-9	0.02	0.02	0.02	0.02	0.03	0.02	0.03	0.03
P10-10	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.03
Média	0.02	0.02	0.02	0.02	0.02	0.03	0.02	0.02

Tabela A.20: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 12

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P12-1	0.16	0.16	0.05	0.05	0.18	0.19	0.05	0.05
P12-2	0.04	0.04	0.04	0.05	0.04	0.04	0.05	0.05
P12-3	0.24	0.25	0.04	0.05	0.14	0.16	0.06	0.06
P12-4	0.07	0.07	0.07	0.06	0.07	0.08	0.05	0.04
P12-5	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
P12-6	0.04	0.03	0.04	0.04	0.04	0.04	0.04	0.04
P12-7	0.04	0.03	0.05	0.04	0.03	0.03	0.04	0.05
P12-8	0.11	0.10	0.06	0.05	0.16	0.10	0.05	0.06
P12-9	0.04	0.04	0.04	0.04	0.04	0.03	0.04	0.05
P12-10	0.11	0.11	0.04	0.04	0.11	0.12	0.05	0.05
Média	0.08	0.08	0.04	0.04	0.08	0.08	0.04	0.04

Tabela A.21: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 15

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P15-1	0.10	0.11	0.12	0.11	0.09	0.09	0.10	0.10
P15-2	0.20	0.20	0.12	0.12	0.11	0.11	0.11	0.11
P15-3	0.10	0.10	0.12	0.11	0.08	0.08	0.10	0.10
P15-4	0.10	0.10	0.11	0.11	0.09	0.09	0.10	0.10
P15-5	0.11	0.11	0.12	0.11	0.09	0.08	0.10	0.10
P15-6	0.11	0.11	0.11	0.11	0.09	0.09	0.10	0.10
P15-7	0.11	0.11	0.11	0.11	0.09	0.09	0.10	0.10
P15-8	0.11	0.11	0.12	0.12	0.09	0.08	0.10	0.10
P15-9	0.19	0.19	0.13	0.12	1.47	0.30	0.12	0.12
P15-10	0.15	0.15	0.12	0.12	0.14	0.11	0.10	0.10
Média	0.12	0.12	0.12	0.12	0.23	0.11	0.11	0.11

Tabela A.22: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 20

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P20-1	1.59	0.83	0.38	0.39	2.76	0.71	0.36	0.36
P20-2	0.26	0.26	0.29	0.29	0.23	0.23	0.27	0.27
P20-3	0.49	0.49	0.31	0.32	0.35	0.34	0.27	0.27
P20-4	0.26	0.27	0.30	0.29	0.23	0.24	0.27	0.27
P20-5	0.27	0.27	0.30	0.30	0.24	0.24	0.27	0.27
P20-6	1.84	0.89	0.42	0.43	0.82	0.52	0.32	0.32
P20-7	0.86	0.48	0.35	0.36	0.76	1.03	0.33	0.33
P20-8	0.39	0.39	0.33	0.32	0.38	0.38	0.30	0.30
P20-9	0.27	0.27	0.29	0.30	0.24	0.24	0.27	0.27
P20-10	0.40	0.40	0.30	0.31	0.39	0.39	0.28	0.28
Média	0.66	0.45	0.32	0.33	0.64	0.43	0.29	0.29

Tabela A.23: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 25

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P25-1	0.59	0.63	0.71	0.75	0.54	0.50	0.65	0.66
P25-2	1.64	0.95	1.23	0.98	1.75	1.01	0.85	0.86
P25-3	1.26	1.26	0.80	0.85	2.32	2.26	0.76	0.76
P25-4	0.77	0.68	0.83	0.76	0.52	0.51	0.67	0.68
P25-5	20.91	7.87	6.43	2.50	12.71	5.53	10.31	3.83
P25-6	20.25	8.47	11.1	5.46	19.9	3.63	11.73	10.76
P25-7	0.68	0.59	0.73	0.75	0.52	0.51	0.69	0.68
P25-8	2.51	1.44	0.88	0.90	4.65	1.30	0.79	0.80
P25-9	7.47	2.05	3.29	1.49	11.49	2.27	3.22	1.68
P25-10	4.70	2.39	0.88	0.88	7.08	3.47	0.76	0.77
Média	6.07	2.63	2.68	1.53	6.14	2.09	3.04	0.30

Tabela A.24: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 30

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P30-1	11.51	4.03	1.81	1.78	21.79	3.69	1.76	1.72
P30-2	7.19	4.96	4.19	2.98	4.90	3.97	3.30	1.92
P30-3	29.30	6.47	9.60	3.53	21.23	5.50	8.97	5.22
P30-4	27.92	87.03	7.68	23.24	31.10	21.58	8.52	23.59
P30-5	17.59	9.09	6.80	2.97	21.67	7.78	7.65	3.59
P30-6	43.31	7.41	5.18	2.14	7.09	3.21	1.91	1.92
P30-7	5.21	2.31	1.63	1.56	8.19	5.56	1.52	1.48
P30-8	16.46	14.8	4.64	2.62	23.89	12.99	2.33	2.43
P30-9	6.95	3.15	2.65	1.84	5.79	3.19	2.41	1.82
P30-10	16.01	4.47	1.98	1.96	29.86	5.36	1.84	1.82
Média	11.14	14.37	4.61	4.46	17.55	7.28	4.02	4.55

Tabela A.25: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 35

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P35-1	49.85	13.22	15.64	7.95	98.75	11.54	16.45	10.83
P35-2	164.06	67.72	53.16	28.59	90.48	27.73	84.36	105.05
P35-3	8.03	5.37	3.76	3.35	7.92	8.01	3.42	3.41
P35-4	3.64	4.68	2.74	2.45	1.9	1.91	2.09	2.08
P35-5	16.31	11.13	3.33	3.62	7.65	5.40	2.59	2.60
P35-6	91.25	238.71	31.31	14.38	43.63	27.81	33.93	15.85
P35-7	44.52	24.09	5.38	4.12	36.21	8.81	5.22	3.45
P35-8	54.79	24.24	12.57	4.84	43.46	12.12	8.90	5.49
P35-9	144.98	1210.16	40.86	311.71	283.09	233.51	63.26	774.29
P35-10	26.98	14.7	9.80	4.52	51.64	8.04	10.78	5.63
Média	60.44	161.40	17.85	38.55	66.47	34.48	23.10	92.86

Tabela A.26: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 35

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P35-1	49.85	13.22	15.64	7.95	98.75	11.54	16.45	10.83
P35-2	164.06	67.72	53.16	28.59	90.48	27.73	84.36	105.05
P35-3	8.03	5.37	3.76	3.35	7.92	8.01	3.42	3.41
P35-4	3.64	4.68	2.74	2.45	1.90	1.91	2.09	2.08
P35-5	16.31	11.13	3.33	3.62	7.65	5.40	2.59	2.60
P35-6	91.25	238.71	31.31	14.38	43.63	27.81	33.93	15.85
P35-7	44.52	24.09	5.38	4.12	36.21	8.81	5.22	3.45
P35-8	54.79	24.24	12.57	4.84	43.46	12.12	8.90	5.49
P35-9	144.98	1210.16	40.86	311.71	283.09	233.51	63.26	774.29
P35-10	26.98	14.7	9.80	4.52	51.64	8.04	10.78	5.63
Média	60.44	161.40	17.85	38.55	66.47	34.48	23.10	92.86

Tabela A.27: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 40

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P40-1	107.51	51.38	91.11	42.77	182.03	68.00	207.03	49.39
P40-2	351.99	105.05	21.68	10.36	279.65	25.69	14.82	9.47
P40-3	64.74	45.11	42.36	16.34	95.83	104.29	46.95	19.55
P40-4	31.51	17.70	6.62	7.06	14.91	11.43	5.88	5.77
P40-5	215.9	219.63	59.20	74.05	304.94	1014.28	58.22	53.25
P40-6	417.86	231.62	62.32	29.84	1006.28	93.95	49.17	26.46
P40-7	169.40	54.34	46.02	19.49	71.19	185.65	32.56	28.11
P40-8	7.84	8.03	5.57	5.65	6.45	6.60	4.60	4.55
P40-9	93.90	51.62	22.82	14.40	78.87	59.57	23.89	16.36
P40-10	144.84	65.97	39.72	13.98	153.13	226.94	50.00	13.07
Média	160.54	85.04	39.74	23.39	219.32	179.64	49.31	22.59

Tabela A.28: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 45

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P45-1	1045.95	685.84	378.01	198.67	1323.95	1491.36	240.33	358.48
P45-2	68.10	29.94	46.66	15.33	68.59	95.38	24.10	27.17
P45-3	114.26	34.94	31.26	13.05	82.69	32.11	16.15	16.97
P45-4	448.98	250.48	98.62	36.92	241.14	236.67	95.04	65.72
P45-5	1215.31	297.65	206.02	78.56	740.71	189.25	141.48	94.64
P45-6	1516.59	482.24	211.56	54.09	868.84	157.56	146.98	98.52
P45-7	431.68	94.57	87.06	25.73	434.92	128.28	50.80	27.3
P45-8	105.59	63.88	81.11	19.34	99.63	80.81	41.43	17.99
P45-9	375.84	2628.55	79.60	73.11	271.40	2171.24	59.74	36.24
P45-10	253.72	222.80	87.31	32.10	696.68	109.64	119.76	36.49
Média	557.60	479.08	130.72	54.69	482.85	469.23	93.58	77.95

Tabela A.29: Comparação de de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 50

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P50-1	46.00	29.51	33.69	38.55	41.15	28.85	65.02	26.48
P50-2	1349.83	1635.65	1470.83	1381.59	1480.44	532.55	2428.73	2726.12
P50-3	1062.60	346.36	349.00	163.18	1506.21	356.26	370.52	590.32
P50-4	3273.57	3257.32	2747.62	1357.59	4408.51	766.21	2909.19	7435.63
P50-5	1506.60	683.80	1204.27	220.81	1739.53	312.19	1209.15	311.62
P50-6	135.32	50.83	96.39	40.15	242.88	42.65	69.14	59.33
P50-7	365.13	180.13	189.12	77.87	522.95	80.23	163.82	52.23
P50-8	82.85	40.85	93.66	44.30	209.67	30.46	62.66	43.59
P50-9	100.07	51.73	80.43	49.54	998.57	45.34	77.41	42.90
P50-10	1543.8	8246.70	619.87	4716.14	3451.68	478.25	804.85	17785.70
Média	946.57	1452.2	688.48	808.97	1460.15	267.29	816.04	2907.39

Tabela A.30: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 55

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P55-1	5273.62	5466.69	1151.65	2721.38	14514.54	706.48	4139.64	461.06
P55-2	1110.15	1109.03	137.17	88.83	2778.43	1197.41	123.2	54.91
P55-3	6313.14	8074.76	1593.03	5318.67	2013.69	764.13	3582.19	560.9
P55-4	9789.16	95650.15	11674.79	43213.28	34736.41	3029.97	10585.76	2579.49
P55-5	13976.29	105645.45	11175.00	163486.08	55784.1	2092.14	18141.4	2106.99
P55-6	3308.97	1107.94	732.35	1317.57	3843.44	1309.29	450.75	150.43
P55-7	512.18	407.63	88.87	74.80	449.33	112.26	93.82	37.92
P55-8	1092.86	960.71	617.31	829.87	3526.24	278.13	1257.88	207.94
P55-9	2266.06	296.10	159.94	250.94	1444.33	524.18	204.81	48.88
P55-10	6775.58	590.64	1209.49	435.55	1955.66	369.13	2995.61	281.13
Média	5041.80	21930.91	2853.96	21773.7	12104.62	1038.31	4157.50	648.96

Tabela A.31: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 60

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P60-1	1423.25	1508.51	455.95	234.16	1588.48	753.95	600.04	226.54
P60-2	1692.94	4017.49	4585.06	4632.05	25562.73	1857.98	985.78	1897.3
P60-3	4229.48	2395.92	3809.88	2086.08	6599.08	2361.69	1658.41	567.2
P60-4	2543.76	1972.89	1263.02	936.14	2611.35	733.54	1326.59	138.91
P60-5	8283.15	545.17	1805.58	483.10	3796.11	984.73	4406.03	610.14
P60-6	14326.07	17704.11	13840.04	23909.78	32034.41	4195.99	25867.68	3232.17
P60-7	1383.63	794.03	866.19	1076.86	4480.34	167.4	648.18	407.24
P60-8	44547.12	12499.34	64011.67	14774.72	11779.81	4673.73	10298.67	4255.68
P60-9	21082.28	8021.61	15571.7	10564.25	8156.53	1620.8	4863.69	2102.87
P60-10	2043.24	1039.12	832.23	976.29	3954.28	412.69	1275.38	204.07
Média	10155.49	5049.81	10704.13	5967.34	10056.31	1776.25	5193.04	1364.21

Tabela A.32: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 65

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P65-1	902.24	1499.14	*****	*****	*****	*****	1330.9	188.49
P65-2	1478.74	198.06	*****	*****	*****	*****	396.10	337.60
P65-3	46886.4	46634.15	*****	*****	*****	*****	106126.4	9230.56
P65-4	255.02	265.06	*****	*****	*****	*****	647.33	404.65
P65-5	1674.82	1193.82	*****	*****	*****	*****	9507.14	2376.79
P65-6	5046.91	2705.87	*****	*****	*****	*****	9314.84	975.50
P65-7	26286.34	10555.14	*****	*****	*****	*****	******	3479.82
P65-8	65447.69	59098.98	*****	*****	*****	*****	56489.61	9939.05
P65-9	64872.05	******	*****	*****	*****	*****	134921.22	28003.45
P65-10	11388.82	3990.13	*****	*****	*****	*****	10853.89	1263.43
Média	22423.90	14015.59	*****	*****	*****	*****	36620.83	5619.93

Tabela A.33: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 70

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P70-1	53843.98	44466.66	*****	*****	*****	*****	*****	*****
P70-2	71055.12	7277.84	*****	*****	*****	*****	*****	*****
P70-3	3320.14	3576.03	*****	*****	*****	*****	*****	*****
P70-4	28927.64	15291.65	*****	*****	*****	*****	*****	*****
P70-5	80865.09	78202.41	*****	*****	*****	*****	*****	*****
P70-6	16484.53	10474.46	*****	*****	*****	*****	*****	*****
P70-7	21612.42	53140.04	*****	*****	*****	*****	*****	*****
P70-8	1280.37	592.14	*****	*****	*****	*****	*****	*****
P70-9	115513.22	64990.75	*****	*****	*****	*****	*****	*****
P70-10	6105.96	18411.04	*****	*****	*****	*****	*****	*****
Média	39900.84	29642.30	*****	*****	*****	*****	*****	*****

Tabela A.34: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 75

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P75-1	*****	60651.68	*****	*****	*****	*****	*****	*****
P75-2	*****	31531.58	*****	*****	*****	*****	*****	*****
P75-3	*****	658.99	*****	*****	*****	*****	*****	*****
P75-4	*****	11411.91	*****	*****	*****	*****	*****	*****
Média	*****	26023.54	*****	*****	*****	*****	*****	*****

Tabela A.35: Comparação de Tempos - Modelos do PML Baseados no Caminho Mínimo-Instâncias 80

Problema	Tempo (s) M3_PML Padrão	Tempo (s) M3_PML Ênfase	Tempo (s) M3_PML Padrão GRASP	Tempo (s) M3_PML Ênfase GRASP	Tempo (s) M4_PML Padrão	Tempo (s) M4_PML Ênfase	Tempo (s) M4_PML Padrão GRASP	Tempo (s) M4_PML Ênfase GRASP
P80-1	*****	25767.53	*****	*****	*****	*****	*****	*****
P80-2	*****	*****	*****	*****	*****	*****	*****	*****
P80-3	*****	133329.0	*****	*****	*****	*****	*****	*****
P80-4	*****	25592.22	*****	*****	*****	*****	*****	*****
Média	*****	65562.91	*****	*****	*****	*****	*****	*****

Tabela A.36: Resultados GRASP para o PML - Instâncias 10

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P10-1	803	0.00	1	803	0.00	0.00	0.00
P10-2	332	0.00	1	332	0.00	0.00	0.00
P10-3	758	0.00	6	595	0.00	0.00	6.72
P10-4	805	0.00	1	767	0.00	0.00	0.00
P10-5	543	0.00	1	535	0.00	0.00	0.00
P10-6	644	0.00	2	561	0.00	1.43	1.43
P10-7	906	0.00	4	811	0.00	0.00	2.47
P10-8	766	0.00	1	694	0.00	0.00	0.00
P10-9	730	0.00	1	636	0.00	0.00	0.00
P10-10	410	0.00	1	410	0.00	0.00	0.00
Média		0.00	1.9		0.00	0.14	1.16

encontrado pela heurística GRASP apresentada na seção (2.7). Nessas tabelas não foram apresentados os limites inferiores e tampouco os *gaps* de relaxação linear das instâncias pois esses valores são os mesmos da Tabelas (A.7) - (A.18).

As Tabelas (A.19)-(A.35) possuem os seguintes campos: *Problema*, apresenta a instância analisada; *Tempo (s) M3_PML Padrão* e *Tempo (s) M4_PML Padrão* mostram o tempo para se provar a otimalidade da instância usando o CPLEX com os parâmetros padrões, exceto o algoritmo de cálculo de limite inferior que, para todas as instâncias, foi utilizado o Método da Barreira; campos *Tempo (s) M3_PML Ênfase* e *Tempo (s) M4_PML Ênfase* apresentam o tempo, em segundos, para se provar a otimalidade da instância alterando a ênfase do CPLEX; para o modelo M3_PML ênfase do CPLEX ficou na otimalidade da solução, já para o modelo M4_PML a ênfase do CPLEX foi na integralidade da solução; campos *Tempo (s) M3_PML Padrão GRASP*, campos *Tempo (s) M4_PML Padrão GRASP* são os mesmos que os *Tempo (s) M3_PML Padrão* e *Tempo (s) M4_PML Padrão* mas utilizando como primeiro limite superior para o CPLEX o valor calculado pela heurística GRASP; os campos *Tempo (s) M3_PML Ênfase GRASP* e *Tempo (s) M4_PML Ênfase GRASP* são os mesmos que os campos *Tempo (s) M4_PML Ênfase* e *Tempo (s) M4_PML Ênfase* mas utilizando como primeiro limite superior para o CPLEX o valor calculado pela heurística GRASP.

Tabela A.37: Resultados GRASP para o PML - Instâncias 12

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P12-1	1116	0.01	119	1116	0.00	2.62	2.62
P12-2	1140	0.00	2	1140	0.00	0.00	0.00
P12-3	915	0.01	131	915	0.00	0.00	2.06
P12-4	592	0.00	0	592	0.00	0.00	0.08
P12-5	1178	0.00	2	1178	0.00	0.00	0.00
P12-6	1031	0.00	1	1031	0.00	0.00	0.00
P12-7	982	0.00	8	982	0.00	0.00	0.00
P12-8	575	0.00	1	575	0.00	3.60	5.91
P12-9	1032	0.00	5	1032	0.00	0.00	0.00
P12-10	899	0.00	1	899	0.00	1.00	1.00
Média		0.00	27		0.00	0.72	1.17

Tabela A.38: Resultados GRASP para o PML - Instâncias 15

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P15-1	1066	0.00	1	1066	0.00	0.00	0.00
P15-2	643	0.00	37	643	0.00	0.00	0.05
P15-3	685	0.00	1	685	0.00	0.00	0.00
P15-4	781	0.00	4	781	0.00	0.00	0.00
P15-5	945	0.00	41	945	0.00	0.00	0.00
P15-6	1121	0.00	16	1121	0.00	0.00	0.00
P15-7	944	0.00	4	944	0.00	0.00	0.00
P15-8	1273	0.00	3	1273	0.00	0.00	0.00
P15-9	1032	0.00	5	1032	0.00	1.21	2.23
P15-10	1170	0.03	155	1170	0.00	0.00	0.06
Média		0.00	26.6		0.00	0.12	0.23

Tabela A.39: Resultados GRASP para o PML - Instâncias 20

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P20-1	1345	0.28	301	1066	0.00	3.23	3.23
P20-2	1083	0.00	7	643	0.00	0.00	0.00
P20-3	1793	0.09	13	685	0.00	0.10	0.00
P20-4	1504	0.01	73	781	0.00	0.00	0.00
P20-5	1537	0.07	15	945	0.00	0.00	0.00
P20-6	1379	0.01	138	1121	0.00	0.95	0.95
P20-7	1848	0.11	47	944	0.00	2.74	2.69
P20-8	1018	0.03	5	1273	0.00	0.11	0.11
P20-9	1789	0.00	7	1032	0.00	0.00	0.00
P20-10	921	0.00	64	1170	0.00	0.02	0.00
Média		0.06	72.72		0.00	0.72	0.70

Tabela A.40: Resultados GRASP para o PML - Instâncias 25

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P25-1	1207	0.00	3	1207	0.00	0.00	0.00
P25-2	1255	0.20	120	1255	0.00	0.50	0.73
P25-3	1987	0.67	339	1987	0.00	0.38	0.38
P25-4	1482	0.50	200	1482	0.00	0.00	0.00
P25-5	2295	5.65	2973	2295	0.00	1.14	3.07
P25-6	2349	0.08	44	2349	0.00	4.17	4.19
P25-7	1845	0.19	126	1845	0.00	0.00	0.00
P25-8	1998	0.06	37	1998	0.00	0.21	1.40
P25-9	1687	2.27	1194	1687	0.00	1.70	1.72
P25-10	1830	0.21	119	1830	0.00	0.00	1.12
Média		0.98	515.5		0.00	0.22	0.28

Tabela A.41: Resultados GRASP para o PML - Instâncias 30

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P30-1	2119	1.25	18	2119	0.00	0.14	1.46
P30-2	1737	5.87	206	1737	0.00	1.27	1.27
P30-3	2098	0.05	16	2098	0.00	2.88	2.89
P30-4	2259	0.32	499	2259	0.00	0.17	3.44
P30-5	2414	5.21	1803	2414	0.00	1.80	1.80
P30-6	3333	1.87	1702	3333	0.00	0.91	1.42
P30-7	1847	3.01	540	1847	0.00	0.00	0.88
P30-8	2082	6.44	264	2082	0.00	0.86	3.11
P30-9	1817	0.09	111	1807	0.55	0.78	0.78
P30-10	2389	1.12	1165	2389	0.00	1.29	1.51
Média		2.52	573.22		0.06	0.98	1.90

Tabela A.42: Resultados GRASP para o PML - Instâncias 35

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P35-1	2525	1.3	148	2525	0.00	1.44	1.54
P35-2	2287	6.41	692	2287	0.00	2.76	2.76
P35-3	2291	13.92	1594	2291	0.00	0.65	0.87
P35-4	2401	1.96	206	2401	0.00	0.00	0.00
P35-5	1994	1.56	174	1994	0.00	0.49	0.60
P35-6	2554	11.71	1131	2554	0.00	3.15	3.21
P35-7	2168	11.78	1144	2168	0.00	1.01	1.08
P35-8	2204	8.45	887	2204	0.00	1.90	2.25
P35-9	2435	7.28	821	2435	0.00	0.68	5.14
P35-10	2670	10.15	1097	2670	0.00	1.08	1.42
Média		7.45	789.04		0.00	1.32	1.89

Tabela A.43: Resultados GRASP para o PML - Instâncias 40

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P40-1	3027	31.73	1923	3027	0.00	2.47	2.65
P40-2	2467	9.34	599	2467	0.00	0.90	1.82
P40-3	2809	115.51	8161	2809	0.00	1.62	2.30
P40-4	3168	82.13	6056	3168	0.00	0.97	1.96
P40-5	2197	29.38	2262	2197	0.00	0.28	0.41
P40-6	3745	260.82	16032	3745	0.00	2.70	2.81
P40-7	2757	7.34	568	2757	0.00	1.05	2.45
P40-8	4225	9.97	790	4209	0.38	1.81	1.85
P40-9	3185	80.23	7166	3185	0.00	0.02	0.02
P40-10	2685	188.55	15806	2685	0.00	1.47	1.47
Média		81.05	5936.3		0.04	1.33	1.77

Tabela A.44: Resultados GRASP para o PML - Instâncias 45

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P45-1	3382	53.76	2750	3382	0.00	2.42	2.91
P45-2	2608	2.26	118	2608	0.00	0.31	1.25
P45-3	3009	39.15	1933	3009	0.00	0.00	0.60
P45-4	2885	30.62	1519	2885	0.00	0.86	1.45
P45-5	3535	10.24	538	3535	0.00	1.15	2.75
P45-6	2841	30.61	1821	2841	0.00	1.53	1.68
P45-7	3406	222.17	11402	3406	0.00	0.18	0.98
P45-8	3155	210.56	11376	3155	0.00	0.87	1.03
P45-9	3484	341.86	15497	3484	0.00	1.96	2.42
P45-10	3532	167.65	8988	3522	0.28	1.83	1.83
Média		110.88	5594.2		0.03	1.11	1.69

Tabela A.45: Resultados GRASP para o PML - Instâncias 50

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P50-1	3150	508.63	18825	3141	0.29	0.57	0.58
P50-2	3549	60.76	1802	3501	1.35	3.34	3.63
P50-3	3962	496.69	16601	3957	0.13	1.38	1.76
P50-4	3852	63.05	2205	3781	1.84	3.08	5.32
P50-5	4535	97.46	3623	4490	0.99	2.25	2.27
P50-6	3460	83.63	2733	3447	0.38	1.04	1.04
P50-7	3658	270.03	8161	3658	0.00	0.86	1.41
P50-8	3783	203.01	6133	3767	0.42	0.96	0.96
P50-9	3024	2.73	81	3024	0.00	0.97	0.97
P50-10	3724	215.74	6347	3718	0.16	2.81	3.18
Média		200.17	6651.1		0.56	1.73	2.11

Tabela A.46: Resultados GRASP para o PML - Instâncias 55

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P55-1	4266	195,55	4345	4229	0.87	3.01	3.07
P55-2	4159	735,73	16273	4159	0.00	0.15	2.19
P55-3	3471	130,43	2870	3412	1.70	3.75	3.76
P55-4	3893	405,73	9224	3823	1.80	4.37	4.60
P55-5	4363	204,87	4457	4308	1.26	4.22	5.35
P55-6	3849	332,77	7214	3826	0.60	2.12	2.23
P55-7	3802	138,9	3503	3796	0.16	0.78	0.79
P55-8	3803	609,3	13490	3720	2.18	4.05	4.06
P55-9	4022	191,12	4065	4011	0.27	1.24	1.27
P55-10	4603	650,98	12926	4520	1.80	3.90	3.91
Média		359.56	7836.7		1.06	2.76	3.12

Tabela A.47: Resultados GRASP para o PML - Instâncias 60

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P60-1	4877	1055.93	13740	4833	0.90	1.03	1.78
P60-2	5094	113.87	1626	4945	2.93	3.51	4.72
P60-3	3894	767.42	9756	3851	1.10	1.94	2.59
P60-4	4834	253.24	3675	4811	0.48	0.84	1.57
P60-5	4638	1093.56	14281	4483	3.34	4.61	4.73
P60-6	4671	1299.54	16875	4564	2.29	3.81	4.46
P60-7	4348	122.23	1738	4251	2.23	3.23	3.23
P60-8	4876	624.21	8198	4704	3.53	5.11	5.24
P60-9	4762	1030.04	13441	4712	1.05	3.03	3.14
P60-10	4428	1381.21	18181	4376	1.17	2.13	2.16
Média		774.12	10151.1		1.90	2.93	3.36

Tabela A.48: Resultados GRASP para o PML - Instâncias 65

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P65-1	4912	1657.00	16358	4844	1.38	2.03	2.19
P65-2	5262	890.07	8903	5106	2.96	3.53	3.53
P65-3	5283	700.32	7032	5140	2.71	5.08	5.29
P65-4	5479	806.86	8082	5274	3.74	4.20	4.20
P65-5	5213	274.52	2723	5000	4.09	4.81	4.81
P65-6	4613	814.23	8200	4539	1.60	3.32	3.32
P65-7	5518	113.27	1085	5308	3.81	5.78	5.78
P65-8	5571	1803.95	17732	5422	2.67	5.47	5.48
P65-9	6386	1051.81	10217	6173	3.34	6.35	6.35
P65-10	5195	415.87	4382	4993	3.89	5.27	7.22
Média		774.12	8471.4		3.02	4.58	4.82

Tabela A.49: Resultados GRASP para o PML - Instâncias 70

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P70-1	5427	1888.28	13039	5207	4.05	6.61	6.63
P70-2	6270	2784.25	19460	6056	3.41	4.84	4.87
P70-3	5810	1630.37	11094	5539	4.66	5.20	5.70
P70-4	6377	1941.86	13307	6159	3.42	5.17	5.17
P70-5	6363	1074.24	7480	6097	4.18	5.50	6.10
P70-6	5858	1805.05	12233	5605	4.32	5.40	5.40
P70-7	5485	674.85	4818	5340	2.64	4.20	4.32
P70-8	6186	349.91	2475	5889	4.80	5.42	5.43
P70-9	5523	2618.57	17941	5236	5.20	7.51	7.52
P70-10	5600	802.64	5603	5476	2.21	3.67	3.70
Média		802.64	10745		3.89	5.35	5.48

Tabela A.50: Resultados GRASP para o PML - Instâncias 75

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P75-1	5659	2815.87	14784	5369	5.12	*****	7.63
P75-2	5886	3485.19	19824	5527	6.10	*****	7.68
P75-3	6402	1486.52	7772	6119	4.42	*****	5.45
P75-4	5832	1698.51	8830	5539	5.02	*****	6.32
Média		2371.52	12802.5		5.17	*****	6.77

Tabela A.51: Resultados GRASP para o PML - Instâncias 80

Problema	Solução GRASP	Tempo(s) Solução GRASP	Iterações GRASP	Ótimo	Gap Ótimo GRASP (%)	Gap LI Mod 1 GRASP (%)	Gap LI Mod 2 GRASP (%)
P80-1	7070	202.51	1152	6552	6.93	*****	7.24
P80-2	*****	******	*****	****	*****	*****	*****
P80-3	6945	3239.58	18729	6494	6.49	*****	7.93
P80-4	6935	1655.82	9539	6389	5.14	*****	6.35
Média		1699.30	9806.66		6.18	*****	7.17

A.6 Comparação CPLEX x CPLEX Usando GRASP como Primeiro Limite Superior

Nesta seção são apresentados os resultados comparando o tempo para se encontrar o ótimo inteiro utilizando o CPLEX com e sem um limite superior pré-computado pela heurística GRASP. Os valores dos limites superiores foram apresentados na seção (A.5).

As Tabelas (A.52)-(A.61) mostram o impacto de se utilizar um limite superior pré-computado no cálculo do tempo total para se provar a otimalidade de uma instância. Utilizou-se como referência o modelo M2_PML com e sem o limite superior pré-computado. Entretanto, uma análise com o modelo M3_PML ou o M4_PML também poderia ser realizada.

As Tabelas (A.52)-(A.61) possuem os seguintes campos: *Problema*, campo que mostra a instância; *Tempo (s) Ótimo M2_PML* e *Tempo (s) Ótimo M2_PML Usando LS GRASP* campos que mostram o tempo utilizado pelo CPLEX para provar a otimalidade da instância usando o modelo M2 com duas estratégias. Na primeira estratégia o CPLEX trabalha com os parâmetros padrões, na segunda, o CPLEX trabalha com a ênfase na integralidade da solução. Isto é viável pois essas estratégias podem ser computadas em paralelo, quando uma conseguir provar a otimalidade a outra técnica pode ser descartada. O campo *Tempo (s) Ótimo M2_PML Usando LS GRASP* mostra o tempo do CPLEX quando ele utiliza a solução encontrada pelo algoritmo GRASP como primeiro limite superior. Já o campo *Tempo (s) Ótimo M2_PML Usando LS* apresenta o tempo quando o CPLEX não utiliza um limite superior pré-computado. O campo *Ganho (%) Usando LS GRASP* relata o ganho relativo ao usar o limite superior do GRASP. Quando esse ganho é positivo significa que utilizar o limite superior do GRASP torna o CPLEX mais rápido. Quando esse valor é negativo significa que não foi vantajoso utilizar o limite gerado pela heurística GRASP.

Tabela A.52: Resultados CPLEX usando Limite Superior GRASP - Instâncias 15

Problema	Tempo (s) Ótimo M2_PML	Tempo (s) Ótimo M2_PML Usando LS GRASP	Ganho (%) Usando LS GRASP	Gap Ótimo GRASP (%)
P15-1	0.06	0.07	-16.67	0.00
P15-2	0.10	0.07	30.00	0.00
P15-3	0.06	0.07	-16.67	0.00
P15-4	0.06	0.06	0.00	0.00
P15-5	0.05	0.07	-40.00	0.00
P15-6	0.05	0.07	-40.00	0.00
P15-7	0.06	0.07	-16.67	0.00
P15-8	0.06	0.07	-16.67	0.00
P15-9	0.31	0.23	25.81	0.00
P15-10	0.11	0.08	27.27	0.00
Média	0.09	0.08	11.11	0.00

Tabela A.53: Resultados CPLEX usando Limite Superior GRASP - Instâncias 20

Problema	Tempo (s) Ótimo M2_PML	Tempo (s) Ótimo M2_PML Usando LS GRASP	Ganho (%) Usando LS GRASP	Gap Ótimo GRASP (%)
P20-1	2.49	1.9	23.69	0.00
P20-2	0.16	0.18	-12.50	0.00
P20-3	0.44	0.43	2.27	0.00
P20-4	0.16	0.18	-12.50	0.00
P20-5	0.19	0.23	-21.05	0.00
P20-6	2.64	0.88	66.67	0.00
P20-7	1.04	1.06	-1.92	0.00
P20-8	0.55	0.29	47.27	0.00
P20-9	0.18	0.21	-16.67	0.00
P20-10	0.29	0.25	13.79	0.00
Média	0.81	0.56	30.86	0.00

Tabela A.54: Resultados CPLEX usando Limite Superior GRASP - Instâncias 25

Problema	Tempo (s) Ótimo M2_PML	Tempo (s) Ótimo M2_PML Usando LS GRASP	Ganho (%) Usando LS GRASP	Gap Ótimo GRASP (%)
P25-1	0.34	0.41	-20.59	0.00
P25-2	1.76	0.98	44.32	0.00
P25-3	0.93	0.68	26.88	0.00
P25-4	0.36	0.43	-19.44	0.00
P25-5	9.83	3.42	65.21	0.00
P25-6	10.6	9.17	13.49	0.00
P25-7	0.38	0.42	-10.53	0.00
P25-8	0.95	0.79	16.84	0.00
P25-9	4.46	3.03	32.06	0.00
P25-10	0.76	0.63	17.11	0.00
Média	3.03	1.99	34.32	0.00

Tabela A.55: Resultados CPLEX usando Limite Superior GRASP - Instâncias 30

Problema	Tempo (s) Ótimo M2_PML	Tempo (s) Ótimo M2_PML Usando LS GRASP	Ganho (%) Usando LS GRASP	Gap Ótimo GRASP (%)
P30-1	5.67	4.11	27.51	0.00
P30-2	3.25	3.26	-0.31	0.00
P30-3	7.68	7.01	8.72	0.00
P30-4	38.35	31.23	18.57	0.00
P30-5	18.40	9.23	49.84	0.00
P30-6	18.47	5.1	72.39	0.00
P30-7	3.72	3.76	-1.08	0.00
P30-8	7.14	7.54	-5.60	0.00
P30-9	4.57	2.1	54.05	0.55
P30-10	6.10	4.44	27.21	0.00
Média	11.33	7.77	31.42	0.06

Tabela A.56: Resultados CPLEX usando Limite Superior GRASP - Instâncias 35

Problema	Tempo (s) Ótimo M2_PML	Tempo (s) Ótimo M2_PML Usando LS GRASP	Ganho (%) Usando LS GRASP	Gap Ótimo GRASP (%)
P35-1	30.26	17.39	42.53	0.00
P35-2	71.90	54.24	24.56	0.00
P35-3	4.21	3.62	14.01	0.00
P35-4	1.49	1.65	-10.74	0.00
P35-5	3.25	2.88	11.38	0.00
P35-6	44.42	37.56	15.44	0.00
P35-7	34.42	28.69	16.65	0.00
P35-8	16.81	7.34	56.34	0.00
P35-9	152.10	112.24	26.21	0.00
P35-10	18.86	17.08	9.44	0.00
Média	37.77	28.26	25.17	0.00

Tabela A.57: Resultados CPLEX usando Limite Superior GRASP - Instâncias 40

Problema	Tempo (s) Ótimo M2_PML	Tempo (s) Ótimo M2_PML Usando LS GRASP	Ganho (%) Usando LS GRASP	Gap Ótimo GRASP (%)
P40-1	151.15	110.49	26.90	0.00
P40-2	32.01	17.62	44.95	0.00
P40-3	200.93	205.44	-2.24	0.00
P40-4	19.63	10.73	45.34	0.00
P40-5	108.16	65.84	39.13	0.00
P40-6	110.23	64.19	41.77	0.00
P40-7	82.41	81.67	0.90	0.00
P40-8	3.82	3.45	9.69	0.38
P40-9	72.77	72.75	0.03	0.00
P40-10	62.74	40.61	35.27	0.00
Média	84.38	67.27	20.27	0.04

Tabela A.58: Resultados CPLEX usando Limite Superior GRASP - Instâncias 45

Problema	Tempo (s) Ótimo M2_PML	Tempo (s) Ótimo M2_PML Usando LS GRASP	Ganho (%) Usando LS GRASP	Gap Ótimo GRASP (%)
P45-1	602.79	263.38	56.31	0.00
P45-2	51.91	29.83	42.54	0.00
P45-3	21.46	19.44	9.41	0.00
P45-4	319.76	159.81	50.02	0.00
P45-5	665.85	536.82	19.38	0.00
P45-6	149.10	150.58	-0.99	0.00
P45-7	252.89	263.97	-4.38	0.00
P45-8	79.16	43.16	45.48	0.00
P45-9	330.50	179.56	45.67	0.00
P45-10	159.38	107.8	32.36	0.28
Média	263.28	175.34	33.40	0.03

Tabela A.59: Resultados CPLEX usando Limite Superior GRASP - Instâncias 50

Problema	Tempo (s) Ótimo M2_PML	Tempo (s) Ótimo M2_PML Usando LS GRASP	Ganho (%) Usando LS GRASP	Gap Ótimo GRASP (%)
P50-1	30.98	27.56	11.04	0.29
P50-2	5468.17	4198.17	23.23	1.35
P50-3	2443.27	912.79	62.64	0.13
P50-4	5107.16	5096.99	0.20	1.84
P50-5	630.12	615.62	2.30	0.99
P50-6	83.73	74.73	10.75	0.38
P50-7	722.75	116.87	83.83	0.00
P50-8	30.91	19.21	37.85	0.42
P50-9	233.51	152.48	34.70	0.00
P50-10	2175.70	2325.26	-6.87	0.16
Média	1692.63	1353.96	20.00	0.56

Tabela A.60: Resultados CPLEX usando Limite Superior GRASP - Instâncias 55

Problema	Tempo (s) Ótimo M2_PML	Tempo (s) Ótimo M2_PML Usando LS GRASP	Ganho (%) Usando LS GRASP	Gap Ótimo GRASP (%)
P55-1	2566.86	1125.39	56.16	0.87
P55-2	783.17	215.16	72.53	0.00
P55-3	4146.93	4125.48	0.52	1.70
P55-4	5468.22	5399.04	1.27	1.80
P55-5	32591.09	19776.28	39.32	1.26
P55-6	1982.98	442.51	77.68	0.60
P55-7	129.54	110.06	15.04	0.16
P55-8	790.60	751.08	5.00	2.18
P55-9	310.45	202.05	34.92	0.27
P55-10	760.62	641.69	15.64	1.80
Média	4953.04	3278.87	33.80	1.06

Tabela A.61: Resultados CPLEX usando Limite Superior GRASP - Instâncias 60

Problema	Tempo (s) Ótimo M2_PML	Tempo (s) Ótimo M2_PML Usando LS GRASP	Ganho (%) Usando LS GRASP	Gap Ótimo GRASP (%)
P60-1	1176.75	1085.19	7.78	0.90
P60-2	17790.20	6872.25	61.37	2.93
P60-3	13653.13	12774.95	6.43	1.10
P60-4	3637.69	3505.60	3.63	0.48
P60-5	10787.50	17495.50	-62.18	3.34
P60-6	59682.29	65018.07	-8.94	2.29
P60-7	1342.98	1265.26	5.79	2.23
P60-8	32543.50	18857.60	42.05	3.53
P60-9	29608.56	23021.64	22.25	1.05
P60-10	5657.83	5772.18	-2.02	1.17
Média	17588.04	15566.82	11.49	1.90

Apêndice B

Problema de Um Veículo de Entrega

B.1 Introdução

Este apêndice tem como objetivo mostrar de uma forma mais detalhada os resultados das instâncias do Problema de Um Veículo de Entrega (PUVE) apresentadas anteriormente. No capítulo 3 foram apresentadas sempre as médias para as dez instâncias de cada conjunto. Neste apêndice são apresentados os tempos e os *gaps* para cada instância, usando cada uma das formulações analisadas.

B.2 Formulações Fluxo

Nesta seção são apresentados os resultados comparando as 2 formulações de fluxo apresentadas na seção (3.4.1). A formulação F1_PUVE, é a formulação multifluxo sem as variáveis p, (3.1)-(3.1). A formulação F2_PUVE, é a formulação F1_PUVE acrescida das restrições (3.13)-(3.18).

As Tabelas (A.1)-(A.6) possuem os seguintes campos: *Problema*, apresenta o nome da instância; campos *Gap RL F1_PML (%)* e *Gap RL F2_PML (%)* apresentam o *gap* de relaxação linear de cada instância e para cada formulação apresentada; campos *Tempo (s) RL F1_PML* e *Tempo (s) RL F2_PML*, mostram o tempo, em segundos, que cada formulação levou para calcular o limite de programação linear de cada instância utilizando-se o Método da Barreira como algoritmo de relaxação linear; campos *Tempo (s) Ótimo F1_PML* e *Tempo (s) Ótimo F2_PML*, mostram o tempo, em segundos, que cada formulação levou para provar a otimalidade da instância.

Tabela B.1: Comparação das Formulações Multiproduto para o PUVE - Instâncias 10

Problema	Gap (%) RL F1_PUVE	Tempo (s) RL F1_PUVE	Tempo (s) Ótimo F1_PUVE	Gap (%) RL F2_PUVE	Tempo (s) RL F2_PUVE	Tempo (s) Ótimo F2_PUVE
I10-1	6.90	0.07	1.69	0.00	0.07	0.09
I10-2	0.00	0.06	0.07	0.00	0.06	0.08
I10-3	3.83	0.07	0.70	0.00	0.08	0.09
I10-4	17.43	0.07	2.44	0.66	0.10	0.27
I10-5	4.06	0.07	1.01	0.00	0.07	0.08
I10-6	0.00	0.07	0.08	0.00	0.08	0.09
I10-7	0.00	0.08	0.09	0.00	0.07	0.09
I10-8	0.00	0.05	0.06	0.00	0.06	0.08
I10-9	10.14	0.07	2.03	2.78	0.09	1.19
I10-10	14.76	0.06	2.36	0.00	0.07	0.09
Média	5.71	0.06	1.05	0.34	0.07	0.21

Tabela B.2: Comparação das Formulações Multiproduto para o PUVE - Instâncias 12

Problema	Gap (%) RL F1_PUVE	Tempo (s) RL F1_PUVE	Tempo (s) Ótimo F1_PUVE	Gap (%) RL F2_PUVE	Tempo (s) RL F2_PUVE	Tempo (s) Ótimo F2_PUVE
I12-1	8.27	0.13	4.44	0.00	0.15	0.18
I12-2	3.41	0.15	3.12	0.00	0.15	0.18
I12-3	8.95	0.14	7.11	0.00	0.15	0.18
I12-4	8.72	0.14	6.08	0.54	0.20	2.93
I12-5	2.32	0.12	0.99	0.00	0.15	0.18
I12-6	1.89	0.15	1.87	0.00	0.14	0.17
I12-7	13.70	0.15	9.59	0.00	0.16	0.18
I12-8	4.45	0.13	3.41	0.00	0.18	0.21
I12-9	14.63	0.11	4.18	0.54	0.23	2.59
I12-10	14.24	0.14	8.04	0.00	0.17	0.20
Média	8.06	0.13	4.88	0.11	0.16	0.70

B.3 Formulações Baseadas em Bianco *et al.*

Nesta seção são apresentados os resultados comparando as 4 formulações baseadas no modelo de Bianco *et al.* [BM$^+$89], apresentadas na seção (3.4.2). Além disso, são apresentados os resultados heurísticos para o PUVE.

Nas Tabelas (B.7) - (B.16) são apresentados tantos os resultados exatos - associados aos modelos M1_PUVE, M2_PUVE, M3_PUVE e M4_PUVE - como os heurísticos para todas as instâncias analisadas. Essas tabelas possuem os seguintes campos: *Tamanho* que mostra o tamanho das instâncias analisadas; *Gap (%) Relaxação Linear M1_PUVE*, *Gap (%) Relaxação Linear M2_PUVE* e *Gap (%) Relaxação Linear M3_PUVE*, campos que mostram os *gaps* de relaxação linear para as formulações M1_PUVE, M2_PUVE e

Tabela B.3: Comparação das Formulações Multiproduto para o PUVE - Instâncias 15

Problema	Gap (%) RL F1_PUVE	Tempo (s) RL F1_PUVE	Tempo (s) Ótimo F1_PUVE	Gap (%) RL F2_PUVE	Tempo (s) RL F2_PUVE	Tempo (s) Ótimo F2_PUVE
I15-1	4.95	0.35	5.10	0.00	0.42	0.47
I15-2	4.55	0.40	10.41	0.00	0.45	0.51
I15-3	13.65	0.39	93.23	0.01	0.62	1.56
I15-4	5.01	0.36	9.53	0.00	0.68	1.39
I15-5	10.43	0.32	24.30	2.11	0.62	11.04
I15-6	7.83	0.42	24.72	1.36	0.81	9.89
I15-7	11.81	0.39	20.11	2.89	0.60	9.96
I15-8	16.38	0.46	32.30	0.00	0.53	0.58
I15-9	6.33	0.40	15.85	1.26	0.70	14.30
I15-10	6.75	0.36	20.72	0.17	0.77	6.72
Média	8.77	0.38	25.62	0.78	0.62	5.64

Tabela B.4: Comparação das Formulações Multiproduto para o PUVE - Instâncias 20

Problema	Gap (%) RL F1_PUVE	Tempo (s) RL F1_PUVE	Tempo (s) Ótimo F1_PUVE	Gap (%) RL F2_PUVE	Tempo (s) RL F2_PUVE	Tempo (s) Ótimo F2_PUVE
I20-1	4.94	1.92	83.28	0.00	1.71	1.92
I20-2	12.51	1.48	443.06	3.74	2.60	82.03
I20-3	13.79	1.49	1153.64	1.62	2.96	51.71
I20-4	14.19	2.19	631.89	1.14	2.86	47.19
I20-5	11.57	1.73	5764.87	1.03	3.02	125.51
I20-6	9.31	1.51	223.86	0.00	2.02	2.17
I20-7	22.06	1.53	11924.09	6.64	2.52	137.08
I20-8	14.01	1.84	2588.25	2.98	2.55	122.34
I20-9	12.75	1.42	1425.38	0.00	1.98	2.11
I20-10	8.13	1.66	199.98	1.43	2.34	57.51
Média	12.33	1.67	2443.83	1.86	2.45	62.95

M3_PUVE. Para a abordagem M4_PUVE, esse *gap* não foi apresentado pois é igual ao da formulação M3_PUVE. *Tempo Ótimo M1_PUVE (s)*, *Tempo Ótimo M2_PUVE (s)*, *Tempo Ótimo M3_PUVE (s)* e *Tempo Ótimo M4_PUVE (s)*, campos que apresentam o tempo, em segundos, para se provar a otimalidade usando-se cada uma das abordagens citadas anteriormente. Além desses, são apresentados resultados relativos à heurística GRASP. O campo *Tempo Grasp (s)* apresenta o tempo, em segundos, para se encontrar o melhor limite superior e, finalmente, o campo *Gap (%) GRASP Ótimo* mostra o *gap*, em porcentagem, entre a melhor solução encontrada pelo GRASP e a solução ótima. Esse *gap* é computado para cada instância pela fórmula (3.27):

Tabela B.5: Comparação das Formulações Multiproduto para o PUVE - Instâncias 25

Problema	Gap (%) RL F1_PUVE	Tempo (s) RL F1_PUVE	Tempo (s) Ótimo F1_PUVE	Gap (%) RL F2_PUVE	Tempo (s) RL F2_PUVE	Tempo (s) Ótimo F2_PUVE
I25-1	*****	*****	*****	0.00	6.17	6.47
I25-2	*****	*****	*****	2.38	6.71	424.15
I25-3	*****	*****	*****	1.50	7.42	252.38
I25-4	*****	*****	*****	2.13	6.90	458.13
I25-5	*****	*****	*****	3.59	7.24	691.64
I25-6	*****	*****	*****	1.45	8.15	398.59
I25-7	*****	*****	*****	3.79	6.79	401.30
I25-8	*****	*****	*****	3.29	7.45	419.63
I25-9	*****	*****	*****	0.50	8.77	201.18
I25-10	*****	*****	*****	1.00	7.61	168.68
Média	*****	*****	*****	1.96	7.32	342.21

Tabela B.6: Comparação das Formulações Multiproduto para o PUVE - Instâncias 30

Problema	Gap (%) RL F1_PUVE	Tempo (s) RL F1_PUVE	Tempo (s) Ótimo F1_PUVE	Gap (%) RL F2_PUVE	Tempo (s) RL F2_PUVE	Tempo (s) Ótimo F2_PUVE
I30-1	*****	*****	*****	1.23	12.92	524.44
I30-2	*****	*****	*****	2.08	16.07	1253.55
I30-3	*****	*****	*****	4.14	14.43	1218.94
I30-4	*****	*****	*****	3.07	14.66	2112.32
I30-5	*****	*****	*****	3.73	13.68	3674.33
I30-6	*****	*****	*****	1.60	11.68	1002.88
I30-7	*****	*****	*****	1.06	13.33	1183.59
I30-8	*****	*****	*****	1.36	14.45	1194.57
I30-9	*****	*****	*****	2.04	14.83	824.24
I30-10	*****	*****	*****	1.11	14.52	562.45
Média	*****	*****	*****	2.14	14.05	1355.13

Tabela B.7: Comparação das Abordagens para o PUVE - Instâncias 10

Problema	Gap (%) Relaxação Linear M1_PUVE	Tempo (s) Ótimo M1_PUVE	Gap (%) Relaxação Linear M2_PUVE	Tempo (s) Ótimo M2_PUVE	Gap (%) Relaxação Linear M3_PUVE	Tempo (s) Ótimo M3_PUVE	Tempo GRASP (s)	Gap (%) GRASP Ótimo	Tempo (s) Ótimo M4_PUVE
I10-1	0.00	0.03	0.00	0.04	0.00	0.02	0.00	0.00	0.02
I10-2	0.00	0.03	0.00	0.04	0.00	0.03	0.00	0.00	0.03
I10-3	0.00	0.03	0.00	0.04	0.00	0.02	0.00	0.00	0.02
I10-4	6.60	0.1	6.60	0.06	6.60	0.08	0.00	0.00	0.08
I10-5	0.00	0.04	0.00	0.04	0.00	0.03	0.00	0.00	0.03
I10-6	0.00	0.05	0.00	0.07	0.00	0.04	0.00	0.00	0.04
I10-7	0.00	0.03	0.00	0.07	0.00	0.05	0.00	0.00	0.05
I10-8	0.00	0.04	0.00	0.07	0.00	0.06	0.00	0.00	0.06
I10-9	2.01	0.22	2.01	0.37	2.01	0.29	0.00	0.00	0.2
I10-10	0.00	0.06	0.00	0.06	0.00	0.1	0.00	0.00	0.1
Média	0.86	0.063	0.86	0.086	0.86	0.072	0.00	0.00	0.063

Tabela B.8: Comparação das Abordagens para o PUVE - Instâncias 12

Problema	Gap (%) Relaxação Linear M1_PUVE	Tempo (s) Ótimo M1_PUVE	Gap (%) Relaxação Linear M2_PUVE	Tempo (s) Ótimo M2_PUVE	Gap (%) Relaxação Linear M3_PUVE	Tempo (s) Ótimo M3_PUVE	Tempo GRASP (s)	Gap (%) GRASP Ótimo	Tempo (s) Ótimo M4_PUVE
I12-1	0.00	0.04	0.00	0.05	0.00	0.05	0.00	0.00	0.05
I12-2	0.00	0.04	0.00	0.05	0.00	0.04	0.00	0.00	0.04
I12-3	0.00	0.07	0.00	0.09	0.00	0.08	0.00	0.00	0.08
I12-4	3.69	0.32	3.69	0.41	3.69	0.55	0.00	0.00	0.40
I12-5	0.00	0.11	0.00	0.14	0.00	0.10	0.00	0.00	0.10
I12-6	0.00	0.09	0.00	0.10	0.00	0.08	0.00	0.00	0.08
I12-7	0.00	0.09	0.00	0.13	0.00	0.09	0.00	0.00	0.09
I12-8	0.00	0.22	0.00	0.21	0.00	0.19	0.00	0.00	0.19
I12-9	4.05	0.60	4.05	0.49	4.05	0.35	0.00	0.00	0.35
I12-10	0.00	0.12	0.00	0.14	0.00	0.10	0.00	0.00	0.10
Média	0.77	0.17	0.77	0.181	0.77	0.163	0.00	0.00	0.148

Tabela B.9: Comparação das Abordagens para o PUVE - Instâncias 15

Problema	Gap (%) Relaxação Linear M1_PUVE	Tempo (s) Ótimo M1_PUVE	Gap (%) Relaxação Linear M2_PUVE	Tempo (s) Ótimo M2_PUVE	Gap (%) Relaxação Linear M3_PUVE	Tempo (s) Ótimo M3_PUVE	Tempo GRASP (s)	Gap (%) GRASP Ótimo	Tempo (s) Ótimo M4_PUVE
I15-1	0.00	0.12	0.00	0.13	0.00	0.11	0.01	0.00	0.11
I15-2	1.03	0.21	1.03	0.41	1.03	0.16	0.02	0.00	0.14
I15-3	0.00	0.22	0.00	0.23	0.00	0.20	0.00	0.00	0.20
I15-4	0.00	0.15	0.00	0.17	0.00	0.14	0.02	0.00	0.14
I15-5	2.18	0.36	2.18	0.67	2.18	0.34	0.00	0.00	0.21
I15-6	3.10	0.84	3.10	0.62	3.10	0.42	0.00	0.00	0.32
I15-7	0.17	0.47	0.00	0.42	0.00	0.36	0.01	0.00	0.36
I15-8	0.00	0.26	0.00	0.28	0.00	0.27	0.00	0.00	0.27
I15-9	0.00	0.33	0.00	0.40	0.00	0.33	0.00	0.00	0.33
I15-10	1.94	1.13	1.94	0.87	1.94	0.61	0.01	0.00	0.54
Média	0.84	0.409	0.82	0.42	0.82	0.29	0.01	0.00	0.26

Tabela B.10: Comparação das Abordagens para o PUVE - Instâncias 20

Problema	Gap (%) Relaxação Linear M1_PUVE	Tempo (s) Ótimo M1_PUVE	Gap (%) Relaxação Linear M2_PUVE	Tempo (s) Ótimo M2_PUVE	Gap (%) Relaxação Linear M3_PUVE	Tempo (s) Ótimo M3_PUVE	Tempo GRASP (s)	Gap (%) GRASP Ótimo	Tempo (s) Ótimo M4_PUVE
I20-1	0.00	0.35	0.00	0.36	0.00	0.34	0.05	0.00	0.38
I20-2	4.17	2.71	4.17	2.65	4.17	1.27	0.08	0.00	0.58
I20-3	3.11	1.99	4.23	8.61	3.11	1.68	0.04	0.00	0.78
I20-4	0.68	0.64	0.68	1.14	0.68	1.13	0.09	0.00	0.64
I20-5	0.57	1.08	0.57	0.88	0.57	0.91	0.11	0.00	1.48
I20-6	0.00	0.88	0.00	0.99	0.00	0.85	0.05	0.00	0.87
I20-7	8.14	27.8	8.27	63.3	8.14	16.84	0.07	0.00	3.58
I20-8	1.16	2.53	1.50	20.36	1.16	2.1	0.08	0.00	1.57
I20-9	0.00	1.04	0.00	1.08	0.00	0.97	0.11	0.00	1.06
I20-10	0.00	1.68	0.13	2.50	0.00	1.55	0.08	0.00	1.66
Média	1.78	4.07	1.96	0.42	1.78	2.76	0.07	0.00	1.26

Tabela B.11: Comparação das Abordagens para o PUVE - Instâncias 25

Problema	Gap (%) Relaxação Linear M1_PUVE	Tempo (s) Ótimo M1_PUVE	Gap (%) Relaxação Linear M2_PUVE	Tempo (s) Ótimo M2_PUVE	Gap (%) Relaxação Linear M3_PUVE	Tempo (s) Ótimo M3_PUVE	Tempo GRASP (s)	Gap (%) GRASP Ótimo	Tempo (s) Ótimo M4_PUVE
I25-1	0.00	0.89	0.00	0.93	0.00	0.90	0.80	0.00	0.88
I25-2	0.78	4.25	0.78	4.17	0.78	7.79	0.73	0.00	2.94
I25-3	0.00	1.29	0.00	1.40	0.00	1.26	0.95	0.00	1.23
I25-4	1.74	19.05	1.93	23.07	1.74	3.82	1.12	0.00	3.83
I25-5	4.20	22.34	4.20	52.02	4.20	43.97	0.90	0.00	16.48
I25-6	1.37	11.15	1.37	3.89	1.37	3.90	0.73	0.00	2.45
I25-7	4.30	26.29	4.35	24.44	4.30	13.05	0.98	0.00	8.18
I25-8	4.19	15.8	4.19	20.77	4.19	18.31	0.89	0.00	7.81
I25-9	2.56	47.04	2.70	16.29	2.56	25.07	0.87	0.00	4.79
I25-10	0.00	3.63	0.18	5.80	0.00	3.62	1.34	0.00	3.82
Média	1.91	15.17	1.97	15.27	1.91	12.16	0.93	0.00	5.24

Tabela B.12: Comparação das Abordagens para o PUVE - Instâncias 30

Problema	Gap (%) Relaxação Linear M1_PUVE	Tempo (s) Ótimo M1_PUVE	Gap (%) Relaxação Linear M2_PUVE	Tempo (s) Ótimo M2_PUVE	Gap (%) Relaxação Linear M3_PUVE	Tempo (s) Ótimo M3_PUVE	Tempo GRASP (s)	Gap (%) GRASP Ótimo	Tempo (s) Ótimo M4_PUVE
I30-1	0.23	15.08	0.33	38.74	0.23	5.93	1.48	0.00	2.94
I30-2	5.76	135.39	6.94	589.35	5.76	92.62	1.16	0.00	32.65
I30-3	1.12	26.35	1.12	25.31	1.12	25.36	3.99	0.00	3.89
I30-4	2.18	20.07	2.18	76.46	2.18	70.58	0.45	0.00	12.89
I30-5	1.37	143.6	1.37	139.15	1.37	56.4	0.82	0.00	8.17
I30-6	1.09	115.22	1.12	83.59	1.09	45.57	0.32	0.00	6.39
I30-7	5.18	9845.37	3.55	1083.54	3.55	339.93	17.25	0.00	27.59
I30-8	0.66	13.31	0.66	28.25	0.66	40.73	2.97	0.00	7.89
I30-9	0.46	54.44	0.46	18.66	0.46	64.26	3.13	0.00	8.43
I30-10	4.20	472.41	4.20	694.26	4.15	334.17	0.52	0.00	123.56
Média	2.23	1084.12	2.19	277.73	2.06	107.55	3.21	0.00	23.44

Tabela B.13: Comparação das Abordagens para o PUVE - Instâncias 35

Problema	Gap (%) Relaxação Linear M1_PUVE	Tempo (s) Ótimo M1_PUVE	Gap (%) Relaxação Linear M2_PUVE	Tempo (s) Ótimo M2_PUVE	Gap (%) Relaxação Linear M3_PUVE	Tempo (s) Ótimo M3_PUVE	Tempo GRASP (s)	Gap (%) GRASP Ótimo	Tempo (s) Ótimo M4_PUVE
I35-1	3.39	170.55	3.40	368.9	3.39	40.18	7.02	0.00	28.77
I35-2	0.44	15.72	0.44	22.32	0.44	17.42	2.71	0.00	4.89
I35-3	1.17	247.67	1.72	331.29	1.17	42.70	20.3	0.00	24.61
I35-4	0.79	115.59	1.06	230.09	0.79	124.1	2.05	0.22	12.01
I35-5	0.67	68.34	0.67	203.87	0.67	59.91	16.75	0.00	9.29
I35-6	0.00	7.01	0.00	7.36	0.00	6.95	15.77	0.00	7.37
I35-7	1.84	160.51	1.84	177.21	1.84	151.15	4.08	0.00	82.59
I35-8	1.32	99.85	1.32	398.13	1.32	172.74	24.77	0.00	18.08
I35-9	2.47	640.25	2.47	137.94	2.47	418.06	1.84	0.00	67.37
I35-10	0.82	54.24	0.82	92.10	0.82	93.87	4.11	0.00	18.39
Média	1.29	157.97	1.37	196.92	1.29	107.55	9.94	0.02	27.33

Tabela B.14: Comparação das Abordagens para o PUVE - Instâncias 40

Problema	Gap (%) Relaxação Linear M1_PUVE	Tempo (s) Ótimo M1_PUVE	Gap (%) Relaxação Linear M2_PUVE	Tempo (s) Ótimo M2_PUVE	Gap (%) Relaxação Linear M3_PUVE	Tempo (s) Ótimo M3_PUVE	Tempo GRASP (s)	Gap (%) GRASP Ótimo	Tempo (s) Ótimo M4_PUVE
I40-1	****	****	****	****	2.06	556.79	28.71	0.00	277.97
I40-2	****	****	****	****	2.44	349.2	161.87	0.00	37.81
I40-3	****	****	****	****	0.94	43.38	17.82	0.00	15.76
I40-4	****	****	****	****	6.27	13894	20.99	0.00	2437.16
I40-5	****	****	****	****	1.49	397.17	137.98	0.00	83.57
I40-6	****	****	****	****	0.79	29.84	118.64	0.00	15.26
I40-7	****	****	****	****	1.82	604.72	140.64	0.00	150.29
I40-8	****	****	****	****	1.59	1414.24	19.87	0.00	242.43
I40-9	****	****	****	****	0.38	66.36	87.28	0.00	24.72
I40-10	****	****	****	****	2.09	905.76	16.52	0.00	117.87
Média	****	****	****	****	1.99	1826.14	75.03	0.00	340.28

Tabela B.15: Comparação das Abordagens para o PUVE - Instâncias 45

Problema	Gap (%) Relaxação Linear M1_PUVE	Tempo (s) Ótimo M1_PUVE	Gap (%) Relaxação Linear M2_PUVE	Tempo (s) Ótimo M2_PUVE	Gap (%) Relaxação Linear M3_PUVE	Tempo (s) Ótimo M3_PUVE	Tempo GRASP (s)	Gap (%) GRASP Ótimo	Tempo (s) Ótimo M4_PUVE
I45-1	****	****	****	****	1.17	****	100.71	0.81	79.56
I45-2	****	****	****	****	4.50	****	240.60	0.23	4038.65
I45-3	****	****	****	****	3.35	****	175.87	0.00	1612.24
I45-4	****	****	****	****	2.50	****	221.54	0.37	870.34
I45-5	****	****	****	****	4.48	****	35.20	0.00	7719.77
I45-6	****	****	****	****	2.09	****	210.98	0.00	177.63
I45-7	****	****	****	****	3.33	****	161.87	0.28	2248.06
I45-8	****	****	****	****	3.45	****	613.59	0.00	22769.06
I45-9	****	****	****	****	3.75	****	3.54	0.00	50.26
I45-10	****	****	****	****	0.97	****	247.62	1.16	137.42
Média	****	****	****	****	2.96	****	201.15	0.28	3970.29

Tabela B.16: Comparação das Abordagens para o PUVE - Instâncias 50

Problema	Gap (%) Relaxação Linear M1_PUVE	Tempo (s) Ótimo M1_PUVE	Gap (%) Relaxação Linear M2_PUVE	Tempo (s) Ótimo M2_PUVE	Gap (%) Relaxação Linear M3_PUVE	Tempo (s) Ótimo M3_PUVE	Tempo GRASP (s)	Gap (%) GRASP Ótimo	Tempo (s) Ótimo M4_PUVE
I50-1	****	****	****	****	1.30	****	57.09	2.21	2044.02
I50-2	****	****	****	****	4.24	****	773.45	1.36	12075.48
I50-3	****	****	****	****	****	****	****	****	****
I50-4	****	****	****	****	3.07	****	630.48	0.78	4039.37
I50-5	****	****	****	****	1.71	****	681.79	0.09	457.08
I50-6	****	****	****	****	****	****	****	****	****
I50-7	****	****	****	****	1.49	****	602.73	1.03	2275.67
I50-8	****	****	****	****	1.75	****	800.16	0.40	15246.99
I50-9	****	****	****	****	****	****	****	****	****
I50-10	****	****	****	****	****	****	****	****	****
Média	****	****	****	****	2.26	****	590.05	0.97	6023.10