LUCIANA PEREIRA DE ASSIS

ALGORITMOS PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEAS

Belo Horizonte

13 de julho de 2007

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

ALGORITMOS PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEAS

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

LUCIANA PEREIRA DE ASSIS

Belo Horizonte

13 de julho de 2007

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Algoritmos para o Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas

LUCIANA PEREIRA DE ASSIS

Dissertação defendida e aprovada pela banca examinadora constituída por:

D. Sc. Geraldo Robson Mateus – Orientador
Universidade Federal de Minas Gerais

D. Sc. Sebastián Alberto Urrutia
Universidade Federal de Minas Gerais

D. Sc. Flávio Keidi Miyazawa
Universidade Federal de Campinas

Belo Horizonte, 13 de julho de 2007

Resumo

Neste trabalho é abordado o Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas (VRPSPD) que é um problema básico de logística reversa, definido como: Dados uma rede de transporte com n consumidores e um depósito, na qual cada consumidor possui uma demanda de coleta e/ou uma demanda de entrega, e um conjunto de veículos com capacidade limitada, o VRPSPD consiste em definir rotas otimizadas que satisfaçam todas as demandas dos clientes, respeitando a capacidade dos veículos.

Devido à complexidade computacional do VRPSPD, o desenvolvimento e análise de heurísticas construtivas é de extrema importância. Este trabalho compara cinco heurísticas construtivas conhecidas da literatura e propõe três novas heurísticas. Os resultados mostram que as heurísticas propostas são melhores em 90% das instâncias testadas considerando a redução da distância trafegada pelos veículos.

A heurística com os melhores resultados da literatura (Heurística Baseada em Inserção) é estendida em duas direções. Na primeira pela introdução de alternativas permitindo que as rotas sejam finalizadas mais cedo e, na segunda direção, pelo uso de ferramentas de apoio a decisão multicritério para escolher o vértice a ser inserido na rota a cada passo da heurística. Os resultados também apresentaram uma melhora significativa da distância trafegada em 92% das instâncias.

As 4 melhores heurísticas analisadas neste trabalho foram utilizadas na fase de construção da heurística baseada no GRASP. Na fase de busca local foram aplicados movimentos de intercâmbio, realocação, eliminação de rotas, cruzamento e 2Opt. Os resultados obtidos foram comparados aos melhores resultados da literatura. Em 64% das instâncias analisadas, as soluções apresentadas pela heurística proposta reduziram a distância trafegada total significativamente. Os resultados também comprovaram que a utilização de uma boa heurística nesta fase melhora a qualidade dos resultados finais da heurística baseada no GRASP.

Abstract

In this work we deal with the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD). This is a basic problem of the reverse logistic, defined as: Given a transportation network with n customers and one depot, and each customer has a demand of pick-up and/or a demand of delivery, and a set of vehicles with limited capacity, the VRPSPD consists of defining optimized routes that satisfy all client demands and meet the capacity of the vehicles.

Due to computational complexity of VRPSPD, the development and analysis of constructive heuristics is extremely important. This work compares five well known constructive heuristics at the literature and proposes three new heuristics. The results show that the proposed heuristics are better in 90% of the tested instances considering the travel distance reduction of the vehicles.

The heuristic with better results at the literature (Insertion-Based Heuristic) ie extend in two direction. Firstly, the alternative allowing to close the current route earlier is considered. In the second direction, it is explored the use of a multicriteria decision aid tool to select the node to be inserted in the route in each step of the heuristic. The results show a significant travel distance improvement in 92% of instances.

The four better heuristics showed in this work was applied in the construction phase of a GRASP based heuristic. This algorithm uses four types of movements in the local search phase: interchange, reallocation, route elimination, crossover and 2Opt. The obtained results was compared with the better results in the literature. The solutions of the proposed heuristic reduced the total traveled distance significantly in 64% of instances. The results confirm that the application of good heuristics in this phase improves the GRASP solution quality.

À minha querida avó Adozinda Martíns Pereira

Agradecimentos

Agradeço primeiramente a Deus por me acompanhar em todos os momentos. Agradeço ao meu querido marido Alessandro Vivas Andrade por acreditar em mim mesmo quando nem eu mesmo acreditava e por ter feito minha inscrição no processo de seleção do mestrado.

Agradeço à minha família, aos meus pais Luciano e Fátima e ao meu irmão Leandro pela paciência, carinho, apoio.

Agradeço ao meu orientador Prof. Geraldo Robson Mateus por transmitir seus conhecimentos e sua experiência e por sua presença sempre constante em todos os passos dessa caminhada.

Por suas contribuições, agradeço ao Prof. Sebastián Urrutia e Martín Gómez Ravetti.

Não posso esquecer de agradecer às secretárias do departamento que estão sempre dispostas a nos ajudar e a Antônia pelos cafezinhos que ajudaram a me manter de pé quando não conseguia mais caminhar.

Agradeço a todos os meus amigos do LaPO (Laboratório de Pesquisa Operacional) e aos amigos do Departamento de Ciência da Computação da UFMG, em especial: Alla, Amiga, Djavan, Tatá, Jú, Osama, César, Aioffi, Bigoinha, Bobo e Raquel. Agradeço a todos que compartilharam comigo as segundas da tristeza, as terças da esperança, as quartas de expectativas, as quintas da alegria, as sextas de despedidas e os finais de semana de muito trabalho. Não citarei todos os nomes pois inúmeros foram os amigos que contribuíram, tomaram cafezinho e ficarão sempre em minhas lembranças.

Agradeço às minhas queridas amigas, Chrys e Adriana, por estarem por perto quando precisava e por compreenderem a minha ausência.

Obrigada a todos de coração!

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

Uma das grandes preocupações das empresas e indústrias é propiciar, com custo mínimo, melhor serviço e escoamento das suas mercadorias. O custo de escoamento de qualquer tipo de produto tem influência direta no custo dos mesmos e pode ser a chave do sucesso ou fracasso de uma empresa.

O transporte de cargas no Brasil é, em geral, rodoviário, e os gastos são elevados devido ao aumento constante no preço dos combustíveis, à cobrança de pedágios, às péssimas condições em que se encontram as rodovias, entre outros. Assim, o presente trabalho tem por objetivo definir rotas otimizadas para redução dos custos das empresas e indústrias.

Uma pesquisa feita em 2005 pela Confederação Nacional de Transportes (2006), mostra que 72% das rodovias brasileiras encontram-se em situação regular, ruim ou péssima em relação a sua conservação. Esse grande problema acarreta um aumento nos custos com manutenção de veículos e uma queda na qualidade dos produtos.

Já nas rodovias administradas por concessionárias, onde as estradas são bem conservadas, existe uma cobrança de pedágio que pode variar entre R$9,60 a R$21,30 nas principais estradas que passam por Minas Gerais como: BR040, BR116, BR262, segundo a Confederação Nacional de Transportes (2006).

Outro fator relevante que aumenta o custo de transporte é o combustível utilizado nos veículos para transporte de cargas. Para escolha do combustível deve-se verificar o quão poluente ele é, o preço, a disponibilidade, entre outros aspectos. Os mais encontrados em postos de abastecimento são: gasolina, álcool, óleo diesel e gás natural veicular.

Hoje o óleo diesel é o mais utilizado pelos veículos de carga. Isso se deve a redução

de impostos cobrado sobre este produto tornando-o mais acessível. Dessa forma o governo mantém o valor do óleo diesel constante em todo o território brasileiro e evita grandes reajustes de preços.

Mesmo sendo um dos combustíveis mais baratos no mercado, o óleo diesel teve um aumento superior a 12% em 2005. Na região nordeste, onde se encontra o menor preço, subiu de R$1,650 em Maio de 2005 para R$1,800 em Outubro de 2005. A região com o maior preço do Diesel é a Centro-Oeste, subindo de R$1,750 para R$1,900 entre Maio e Outubro de 2005. Esses dados foram obtidos através de um levantamento feito pela Associação Nacional de Petróleo (2006).

Com tantos custos, os consumidores finais chegam a pagar 10% a 15% do valor da mercadoria com transporte do mesmo. No Brasil, o gasto com transporte de carga corresponde a 10,8% do PIB (Produto Interno Bruto) (Alvarenga, 2005). Logo, o transporte é uma das atividades que mais contribuem nos custos logísticos e qualquer redução nos custos de transporte poderá acarretar em uma economia significativa para empresas e consumidores. Existem muitos estudos que visam minimizar os custos de transporte e otimizar a logística. Entre eles, vários trabalhos se concentram na Otimização Combinatória.

O Problema de Roteamento de Veículos tradicional consiste em definir rotas entre um depósito e um conjunto de pontos de entrega que minimize o custo de transporte. O Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas é uma variação deste problema e é definido como: dado uma rede de transporte onde se localizam n consumidores e 1 depósito, onde cada consumidor i tem uma demanda p_i de coleta ou demanda d_i de entrega ou ambas, e dado um conjunto de veículos com capacidade limitada Q, definir rotas otimizadas que satisfaçam a todas as demandas dos consumidores, respeitando a quantidade e a capacidade dos veículos (Dell'Amico et al., 2005).

Estudos envolvendo este problema tem crescido nos últimos anos por se tratar de um problema básico da logística reversa. Nos últimos anos as empresas vêm buscando formas eficientes de gerenciar o retorno de materiais após sua venda e consumo para reciclá-los, remanufaturá-los ou reaproveitá-los, agregando novos valores a eles. As vantagens dessa prática são: redução de custo, redução dos danos ao meio ambiente e garantia do direito do consumidor de devolução e troca de mercadorias entre outros.

A busca de um método eficiente para definir tais rotas viáveis e otimizadas que satisfaçam as demandas dos consumidores pode reduzir consideravelmente o custo das grandes empresas e grandes indústrias, além de ser um grande desafio para os pesquisadores já que ele é um problema da classe NP-difícil (Dethloff, 2001).

Apesar das grandes vantagens de definir rotas otimizadas, que minimize os custos com transportes, apenas 5% das grandes empresas de transporte rodoviário utilizam sistema de informação como roteirizadores, segundo pesquisa da Confederação Nacional de Transportes

(2006).

1.2 Principais Contribuições

Este trabalho apresenta um estudo comparativo entre algumas heurísticas construtivas abordadas na literatura e propõe outras novas a partir dessas. Outro método utilizado para buscar soluções para o problema de roteamento de veículos com coleta e entrega simultâneas foi a heurística baseada no GRASP (Feo e Resende, 1995). As 4 melhores heurísticas construtivas analisadas neste trabalho foram utilizadas na fase de construção. Na fase de busca local foram aplicados movimentos de intercâmbio, realocação, eliminação de rotas, cruzamento e 2Opt.

Os resultados da heurística GRASP melhoraram significativamente os resultados da literatura. Em 64% das instâncias analisadas, a heurística proposta reduziu a distância trafegada total, acarretando em uma melhoria da qualidade das melhores soluções da literatura em 2,58%. Os resultados também comprovaram que a utilização de uma boa heurística nesta fase melhora a qualidade dos resultados finais da heurística baseada no GRASP.

Vários autores utilizam diversas instâncias para o problema de roteamento de veículos com coleta e entrega simultâneas. Para analisar os resultados obtidos nesse trabalho, os testes foram feitos com instâncias encontradas na literatura. Portanto, todas essas instâncias foram alteradas para um formato único baseado no TSPLIB [1] possibilitando a criação de uma biblioteca completa de instâncias para trabalhar o problema.

Para finalizar, o algoritmo melhor avaliado foi inserido no software OTIMAL (Otimização Integrada em Aplicações Logísticas - Produção e Transporte - Projeto CT-INFO/CNPq - DCC/UFMG), proposto para oferecer soluções de logística e transporte, e tem o objetivo de otimizar a produção e escoamento das mercadorias em empresas e indústrias. A sua primeira versão irá oferecer ferramentas para definir rotas otimizadas para entrega e coleta de mercadorias, além de outros módulos que permitem ao usuário definir o sequenciamento de tarefas em máquinas, distribuição, empacotamento de mercadorias nos veículos, etc.

1.3 Organização

No Capítulo 2 é apresentado o Problema de Roteamento de Veículos, algumas de suas variações, trabalhos relacionados e é finalizado com a apresentação do Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas.

[1] Biblioteca de diversas instâncias para diversos problemas do caixeiro viajante, roteamento de veículos e outros

No Capítulo 3 são abordados alguns métodos heurísticos para resolver o Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas. Os métodos implementados muitas vezes requerem uma solução para o Problema Caixeiro Viajante ou Problema do Caixeiro Viajante com Coleta e Entrega. Portanto, inicialmente são descritos quatro algoritmos para solucionar esses problemas. Em seguida, oito heurísticas para resolver Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas são apresentadas, sendo que três delas são propostas neste trabalho.

No Capítulo 4 é detalhada a heurística proposta por Dethloff (2001) com algumas modificações e apresentada sua aplicação em ferramentas de apoio a decisão multicritérios.

No Capítulo 5 uma metaheurística GRASP é proposta tendo como base as heurísticas apresentadas nos Capítulos 3 e 4, aplicadas em sua fase de construção.

Finalmente, as conclusões e trabalhos futuros são abordados no Capítulo 6.

Capítulo 2

Problema de Roteamento de Veículos com Coleta e Entrega

O objetivo deste capítulo é introduzir o problema de roteamento de veículos e apresentar uma revisão bibliográfica sobre o problema de roteamento de veículos com coleta e entrega e as principais metodologias utilizadas para a sua solução. Na Seção 2.1 é feita uma descrição rápida do problema de roteamento de veículos e algumas de suas variações para contextualização do problema a ser tratado nessa dissertação. Na Seção 2.2 são descritas metodologias empregadas em diversos problemas que trabalham com serviços de coleta e entrega. A Seção 2.3 apresenta o problema de roteamento de veículos com coleta e entrega simultâneas que é o foco principal deste trabalho.

2.1 Problema de Roteamento de Veículos

O Problema de Roteamento de Veículos (VRP, do inglês *Vehicle Routing Problem*), introduzido por Dantzig e Ramser (1959), tem como objetivo definir rotas entre um depósito e um conjunto de pontos de entrega que minimize o número de veículos, a distância percorrida ou tempo. As restrições básicas do problema consistem em:

- Cada cidade é visitada uma única vez por um único veículo;
- Cada rota é iniciada num depósito e finalizada no mesmo depósito;
- Todas as demandas de todos os consumidores devem ser satisfeitas.

Outras restrições que podem ser acrescentadas ao problema são:

- Restrição no número de pontos de entrega em cada rota;
- Restrição de tempo ou distância de uma rota;
- Restrição de janela de tempo: cada ponto deve ser visitado em um período de tempo específico;
- Restrição de precedência entre cidades: o ponto de entrega i deve ser visitado antes do ponto j;
- Tipo de frota: veículos disponíveis para execução das rotas são homogêneos ou heterogêneos;
- Quantidade de veículos contidos na frota: quantidade limitada ou ilimitada;
- Capacidade dos veículos: veículos com capacidade limitada ou ilimitada;
- Quantidade de depósitos: único depósito ou vários depósitos.

Encontram-se na literatura vários trabalhos publicados que tratam diversos problemas de roteamento de veículos com diferentes abordagens:

- um único depósito (Beasley, 1983; Ho e Gendreau, 2006; Campos e Mota, 2000; Bräysy et al., 2004) ou vários depósitos (Salhi e Nagy, 1999; Fan et al., 2007),
- veículos homogêneos (Campos e Mota, 2000; Chin et al., 1999; Bräysy et al., 2004) ou heterogêneos (Belfiore e Yoshizaki, 2006; Choi e Tcha, 2007),
- modelo estático (Ho e Gendreau, 2006; Chin et al., 1999) ou dinâmico (Alvarenga et al., 2005; Montemanni et al., 2002),
- com serviços de coleta e/ou entrega (Min, 1989),
- com restrição de janela de tempo (Reimann et al., 2002; Ombuki et al., 2006; Rousseau et al., 2002; Bräysy et al., 2004), entre outros.

Um problema bastante abordado na literatura, o Problema do Caixeiro Viajante (TSP, do inglês *Traveling Salesman Problem*) que consiste em encontrar um trajeto que visite N cidades diferentes, sem repetição, retornando à origem e utilizando a menor rota possível, pode ser considerado um Problema de Roteamento de Veículos quando esse possui apenas um único veículo com capacidade ilimitada.

Além da quantidade, as capacidades dos veículos podem divergir entre homogêneos ou heterogêneos(CVRP, do inglês *Capacitated Vehicle Routing Problem*). O serviço oferecido pode ser de coleta, entrega ou ambos, e estático ou dinâmico (DVRP, do inglês *Dynamic*

Vehicle Routing Problem). Quanto aos depósitos, existem problemas com um único depósito ou com vários depósitos.

Um outro problema bastante abordado na literatura é o Problema de Roteamento de Veículos com Janela de Tempo (VRPTW, do inglês *Vehicle Routing Problem With Time Windows*) que inclui um intervalo de tempo para começar e terminar o atendimento no consumidor e ainda um intervalo para saída e retorno ao depósito (Alvarenga, 2005).

Um dos problemas que efetua coleta durante o trajeto é o Problema de Roteamento de Veículos com *Backhauls* (VRPB). Esse consiste em efetuar entregas e coletas durante o trajeto, mas as entregas devem ser efetuadas antes das coletas e, em cada ponto desse trajeto, existe somente demanda de entrega ou coleta (Gelogullari, 2004).

Este trabalho irá tratar de problemas de roteamento de veículos com uma frota de veículos capacitados e homogêneos, com um único depósito e serviços de coleta e entrega.

2.2 Trabalhos Relacionados

Vários problemas de roteamento de veículos com coleta e entrega têm sido estudados para tratar as diferentes situações reais. Todos eles concentram em um único objetivo: uso eficiente de uma frota de veículos que deve satisfazer demandas de entrega e/ou coleta de um conjunto de consumidores ou clientes. As demandas de cada consumidor devem ser satisfeitas por um único veículo (Xu et al., 2003).

Tendo em vista essa semelhança, Savelsbergh e Sol (1995) apresenta o Problema Geral de Coleta e Entrega (GPDP, do inglês *General Pickup and Delivery Problem*) que combina várias características encontradas entre esses problemas práticos de coleta e entrega. O GPDP consiste em definir um conjunto de rotas afim de satisfazer um conjunto de demandas de transporte.

Cada consumidor possui demandas de coleta e entrega. As demandas de coleta possuem informações sobre a carga total a ser coletada pelo veículo e em quais consumidores essa carga deve ser entregue posteriormente. As demandas de entrega possuem informações sobre a carga total a ser entregue e em quais consumidores ela deve ter sido coletada anteriormente. Portanto, existe uma regra de precedência entre consumidores, pois para satisfazer uma demanda de entrega o veículo deverá satisfazer anteriormente a respectiva demanda de coleta.

A posição inicial e a posição final dos veículos também é informada. Assim, o objetivo é definir rotas otimizadas que tenham como ponto de partida e ponto de chegada aqueles definidos para os veículos e que satisfaçam todas as demandas dos consumidores sem extrapolar a capacidade dos veículos.

Parragh et al. (2006) divide o Problema Geral de Coleta e Entrega em dois grupos:

- Grupo 1: Transporte de cargas do depósito para os consumidores e dos consumidores para o depósito
- Grupo 2: Transporte de carga entre consumidores

No primeiro grupo todos os veículos partem do depósito contendo a carga necessária para satisfazer as demandas de entrega dos consumidores. As cargas recolhidas pelo veículo nos consumidores devem ser entregues no depósito. Já para o segundo grupo, os veículos partem de um ponto inicial sem qualquer carga e percorre vários pontos de coleta e/ou entrega de itens e então retornam a um ponto final. Neste caso demandas de entrega são sempre satisfeitas por itens coletados anteriormente pelo veículo.

Alguns problemas relacionados a esses grupos são apresentadas nas Subseções 2.2.1 a 2.2.6. O problema de roteamento de veículos com coleta e entrega simultâneas será abordado na próxima Seção (2.3).

2.2.1 Problema de Coleta e Entrega

O Problema de Coleta e Entrega (PDP, do inglês *Pickup and Delivery Problem*) é um problema do grupo onde o transporte de cargas é feito entre consumidores (Grupo 2). No PDP cada demanda de coleta possui informações sobre a carga total a ser coletada pelo veículo e o destino dessa carga. As demandas de entrega possuem informações sobre a carga total a ser entregue e a origem da carga. Todos os veículos, então, partem de um depósito para satisfazer as demandas e retornam novamente ao depósito finalizando o trajeto (Savelsbergh e Sol, 1995).

Lu e Dessouky (2006) apresentam uma heurística baseada em inserção aplicada ao Problema de Coleta e Entrega com Janela de Tempo (PDPTW, do inglês *Pickup and Delivery Problem with Time Windows*). O procedimento da heurísticas é inserir novos nós em rotas já existentes até que isso não seja mais possível. Quando isso ocorre, uma nova rota é criada e o processo continua. Para decidir qual o próximo nó a ser inserido e onde ele deve ser inserido, a maioria das heurísticas de inserção usam como critério o aumento do custo ou distância da rota com a inserção de um novo nó. Dessa forma, o autor propõe uma alternativa para melhorar os resultados inserindo mais um critério de verificação. Esse novo critério avalia o grau de liberdade para futuras inserções de acordo com as janelas de tempo.

Outro aspecto que difere a heurística proposta por Lu e Dessouky (2006) das demais é o fato de considerar soluções visualmente melhores. O autor desenvolve um método para avaliar soluções visivelmente mais atrativas chamado *crossing length percentage* (CLP) para incorporar na heurística de inserção. Os resultados obtidos com a heurística construtiva baseada

em inserção proposta pelo autor foram bons, inclusive, melhores que a heurística *Simulated Annealing* com Busca Tabu proposta por Li e Lim (2001) aplicada ao mesmo problema (PDPTW).

Outra heurística encontrada na literatura para o PDPTW é um Algoritmo Genético. Pankratz (2005) propõe um algoritmo em que cada gene representa um grupo de demandas. Os resultados apresentados mostraram que a proposta do GGA (do inglês *Grouping Genetic Algorithm*) foram bons e bastante competitivos em relação aos demais métodos apresentados na literatura para solucionar o PDPTW.

Nanry e Barnes (2000) apresentaram uma metaheurística Busca Tabu Reativa (RTS, do inglês *Reactive Tabu Search*) para resolver o problema de coleta e entrega com janela de tempo (PDPTW, do inglês *PDP With Time Windows*). Na abordagem feita pelo autor ele considera os veículos homogêneos localizados em um único depósito. O transporte requer a coleta de material em um lugar predeterminado durante um tempo especificado (janela de tempo) e entregue no seu destino. Esse trabalho marca a primeira aplicação da Busca Tabu Reativa para resolução deste problema apresentando bons resultados. Os testes foram feitos em redes com 25, 50 e 100 consumidores. No primeiro caso, os testes foram feitos em 29 instâncias e em todos a solução ótima foi encontrada. Já o segundo caso, dos 15 testes aplicados, 14 retornaram a solução ótima. E no terceiro, 9 testes foram feitos sendo que foram obtidas 8 soluções ótimas.

Caricato et al. (2003) abordam o problema de coleta e entrega com contenção de via. Neste contexto, a rede de transporte é composta por vias sendo que cada uma possui uma capacidade unitária. Ou seja, em cada via pode trafegar um único veículo. Quando dois ou mais veículos tentam entrar na via ao mesmo tempo, uma contenção é aplicada e o último deles é removido ou roteado novamente. O autor apresenta duas heurísticas seqüenciais e uma heurística baseada na metaheurística Busca Tabu Paralela.

Outro trabalho encontrado na literatura é o de Ropke e Pisinger (2006) que tratam o *PDP With Time Windows*. Nessa abordagem, o autor limita a quantidade de veículos. O método utilizado para resolução do problema é uma adaptação da heurística *Large Neighborhood Search*. Esse método mostrou bons resultados em um tempo computacional razoável em testes feitos em 350 redes distintas, com um número superior a 500 consumidores. O método, em 50% dos problemas, provê melhores soluções em relação aos melhores resultados conhecidos na literatura.

Lau e Liang (2001) tratam o mesmo problema com metaheurística Busca Tabu. O autor apresenta uma heurística denominada *Partitioned Insertion* que combina as vantagens da *insertion heuristic* e *sweep heuristic* e posteriormente utiliza a Busca Tabu para melhorar a solução. Além de um algoritmo, o trabalho apresenta uma estratégia para gerar boas instâncias para o problema.

Um algoritmo exato foi proposto por Ruland e Rodin (1997), usando o método *Branch-and-Cut*. Ele mostra formulações para o Problema de Coleta e Entrega como um problema de programação inteira. Os testes foram feitos com base em campos aéreos dos Estados Unidos, gerando redes com 7 a 15 nós de demanda e coleta. O tempo de resposta varia de 3 a 1246 segundos.

2.2.2 Problema Dial-a-Ride

O Problema Dial-a-Ride (DARP, do inglês *Dial-a-Ride Problem*) é um Problema de Coleta e Entrega onde a carga são pessoas, chamadas de clientes ou consumidores, e o tamanho das cargas são todas iguais a 1. Os clientes são recolhidos em locais pré-definidos e transportados para pontos conhecidos. O objetivo é minimizar o número de veículos, se o problema é composto por múltiplos veículos (m-dial-a-ride ou MDARP), ou distância trafegada, ou ambos. O problema deve considerar o tempo gasto no embarque e desembarque de passageiros.

Este problema foi proposto por Psaraftis (1980) que desenvolveu um algoritmo de programação dinâmica para casos com um único veículo. Existem na literatura algumas referências apresentando heurísticas para resolução do trabalho. Ho e Haugland (2004) implementara um algoritmo Busca-Tabu e um algoritmo híbrido utilizando GRASP e Busca-Tabu. Ele comprovou que os resultados do primeiro algoritmo foram melhores que o segundo, apesar da maior robustez do segundo. Cordeau e Laporte (2003) propuseram uma heurística Busca Tabu para o problema com múltiplos veículos. Bergvinsdottir (2004) apresenta um algoritmo genético que utiliza a clássica estratégia "dividir e rotear". Assim, a fase de divisão dos nós é feita com um algoritmo genético.

O trabalho apresentado por Cordeau (2006) compara um algoritmo *Branch-and-Cut* e o CPLEX. Os testes foram feitos com redes de, no máximo, 32 consumidores, mostrando um maior desempenho do método *Branch-and-Cut*.

O problema Dial-a-Ride pode ser associado ao grupo 1, pois considera as requisições de transporte associadas a uma origem e um destino.

2.2.3 Problema de Roteamento de Veículos com Coleta e Entrega

O Problema de Roteamento de Veículos com Coleta e Entrega (VRPPD, do inglês *Vehicle Routing Problem with Pickup and Delivery*) é uma extensão do Problema de Coleta e Entrega. Em ambos as cargas são transportada entre pontos de coleta e entrega (Grupo 2), mas no VRPPD uma carga coletada pode ser utilizada para satisfazer qualquer demanda de entrega (Parragh et al., 2006; Hernádez-Pérez e Salazar-Gonzáles, 2004).

Alguns trabalhos da literatura (Salhi e Nagy, 1999; Montané e Galvão, 2006; Mosheiov,

1998) trabalham com o transporte de cargas do depósito para os consumidores e dos consumidores para o depósito (Grupo 1). Nestes casos os veículos partem do depósito com as cargas a serem entregues nos consumidores e retorna ao depósito com as cargas coletadas pelos consumidores.

2.2.4 Problema de Entrega Expressa

O Problema de Entrega Expressa (EDP, do inglês *Express Delivery Problem*) foi proposto por Montané et al. (1997), a fim de resolver problemas de entrega aérea expressa. Ele é um outro problema com demandas de coleta e entrega em cada nó. Neste caso, o atendimento da demanda é composto por duas fases:

- Fase de Coleta
- Fase de Entrega

Deve-se notar que as fases de coleta e entrega não necessariamente coincidem. Uma rota é definida entre as cidades onde existem demandas de coleta. Todas demandas de uma mesma cidade são agrupadas em um carregamento (demanda de coleta). Depois de efetuadas todas as coletas, o carregamento é encaminhado a um "hub"(ponto intermediário que pode ser visto como o depósito definido no Problema de Roteamento de Veículos). No "hub" os carregamentos são desfeitos e as demandas de entrega são organizadas conforme a cidade destino. Uma nova rota é definida partindo do "hub" até as cidades (pontos de demanda de entrega).

O Problema de Entrega Expressa pode ser considerado um misto entre o grupo 1 e o grupo 2, as requisições de transporte especificam uma origem e um destino para cada item o que demonstra uma característica do grupo 2. No entanto, em uma mesma rota o veículo não efetua coletas e entregas permitindo que todas as demandas a serem entregues e coletadas partam do único ponto, o "hub".

Montané et al. (1997) mostram algumas heurísticas para resolução do problema e executa testes com 20 cidades comparando-as com resultados exatos e até 100 cidades fazendo comparações entre as heurísticas abordadas. Os resultados foram satisfatórios retornando soluções bem próximas da ótima.

2.2.5 Problema do Caixeiro Viajante com Coleta e Entrega

O Problema do Caixeiro Viajante com Coleta e Entrega (TSPPD, do inglês *Traveling Salesman Problem with Pickup and Delivery*) é uma extensão do TSP, sendo que, em cada ponto

ou cada cidade, existe demanda de coleta ou entrega de mercadoria. Se ambas as demandas (entrega e coleta) puderem ocorrer em um mesmo ponto, o problema é denominado Problema do Caixeiro Viajante com Coleta e Entrega Simultâneas (TSPSPD, do inglês *Traveling Salesman Problem With Simultaneous Pickup and Delivery*). Ambos problemas consideram o transporte de cargas do depósito para os consumidores e dos consumidores para o depósito. Portanto, eles estão incluídos no Grupo 1.

O Problema do Caixeiro Viajante com Coleta e Entrega foi proposto por Mosheiov (1994). O autor apresenta uma abordagem heurística para o problema e propõe algumas aplicações.

Gendreau et al. (1999) apresenta algoritmos aproximativos e um algoritmo exato (*Branch-and-Cut*) é descrito por Hernádez-Pérez e Salazar-Gonzáles (2004). Este último mostrou bons resultados em problemas com até 75 consumidores.

2.2.6 TSP com Backhaus e VRP com Backhaus

O Problema do Caixeiro Viajante com Backhaus e o Problema de Roteamento de Veículos como Backhaus são extensões do TSP e VRP, respectivamente. A diferença é que ambos trabalham com coleta e entrega, sendo que todas as entregas devem ser concluídas antes que as coletas possam ser efetuadas. Ambos problemas consideram o transporte de carga entre depósito e consumidor e vice-versa, tornando-o um problema do Grupo 1.

Goetschalckx e Jacobs-Blecha (1993) apresentam uma heurística baseada no problema de assinalamento generalizado. Gelogullari (2004) mostra um algoritmo exato para resolução do VRPB e Reimann et al. (2002) aplicam a metaheurística "Colônia de Formigas".

Potvin et al. (1996) mostram uma heurística construtiva para o VRPB. Ela insere cada consumidor na rota usando uma ordem de prioridade. Posteriormente, os autores propõem um algoritmo genético para melhorar a qualidade das soluções.

Ghaziri e Osman (2003) propõem uma rede neural para resolução do TSPB. Os resultados obtidos mostram que a abordagem proposta é comparável com as heurísticas encontradas na literatura em termos de qualidade da solução e recursos computacionais. Os testes foram aplicados em grandes instâncias, superiores a 1000 consumidores.

2.3 Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas

Todos os problemas citados anteriormente são referentes ao problema de roteamento de veículos com serviços de coleta e entrega, mas serviços não simultâneos. O Problema de

Roteamento de Veículos com Coleta e Entrega Simultâneas (VRPSPD) é uma variação clássica do VRP.

O objetivo do problema é reduzir os esforços e custos em ambas as atividades (coleta e entrega) executando-as simultaneamente. Um exemplo de aplicação do VRPSPD é a distribuição de bebidas. Nele, cada consumidor requer uma demanda de bebida e requer, também, a devolução dos recipientes ou cascos vazios. Assim, efetuando a entrega de bebidas e coleta dos cascos vazios simultaneamente reduz o custo de transporte.

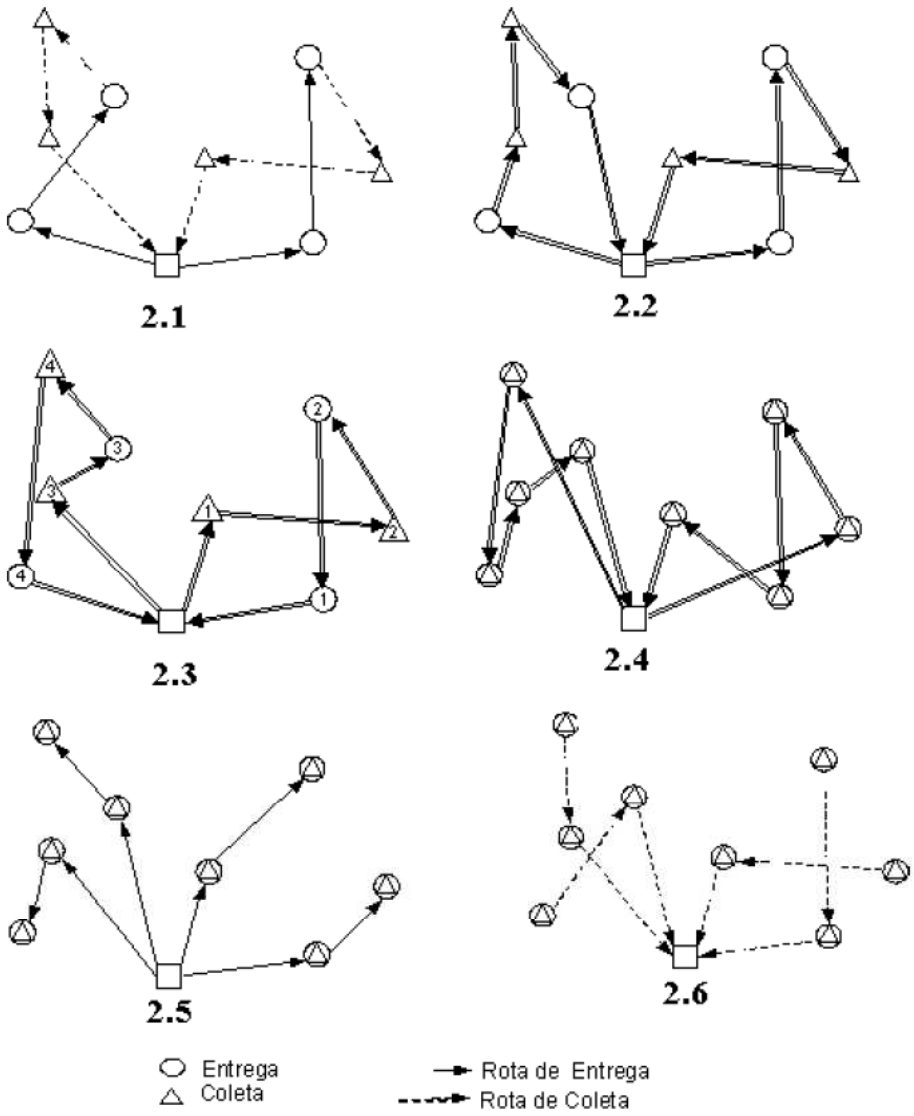


Figura 2.1: Problemas de Roteamento de Veículos com coleta e entrega. (2.1) VRPB; (2.2)VRP_PD; (2.3)MDARP; (2.4)VRPSPD; (2.5) e (2.6) EDP (Montané e Galvão, 2006).

A Figura 2.1 mostra alguns exemplos de problemas de roteamento com serviços de coleta e entrega, entre eles: Problema de Roteamento de Veículos com *Backhauls*, Problema de Rotea-

mento de Veículos com Coleta e Entrega, Problema *m-Dial-a-Ride*, Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas e as fases do Problema de Entrega Expressa. O VRPSPD pode ser reduzido a um VRPPD (*Vehicle Routing Problem with Pickup and Delivery*) ou ainda ao problema *M-Dial-a-Ride*, como mostra a Figura 2.2 (Montané e Galvão, 2006).

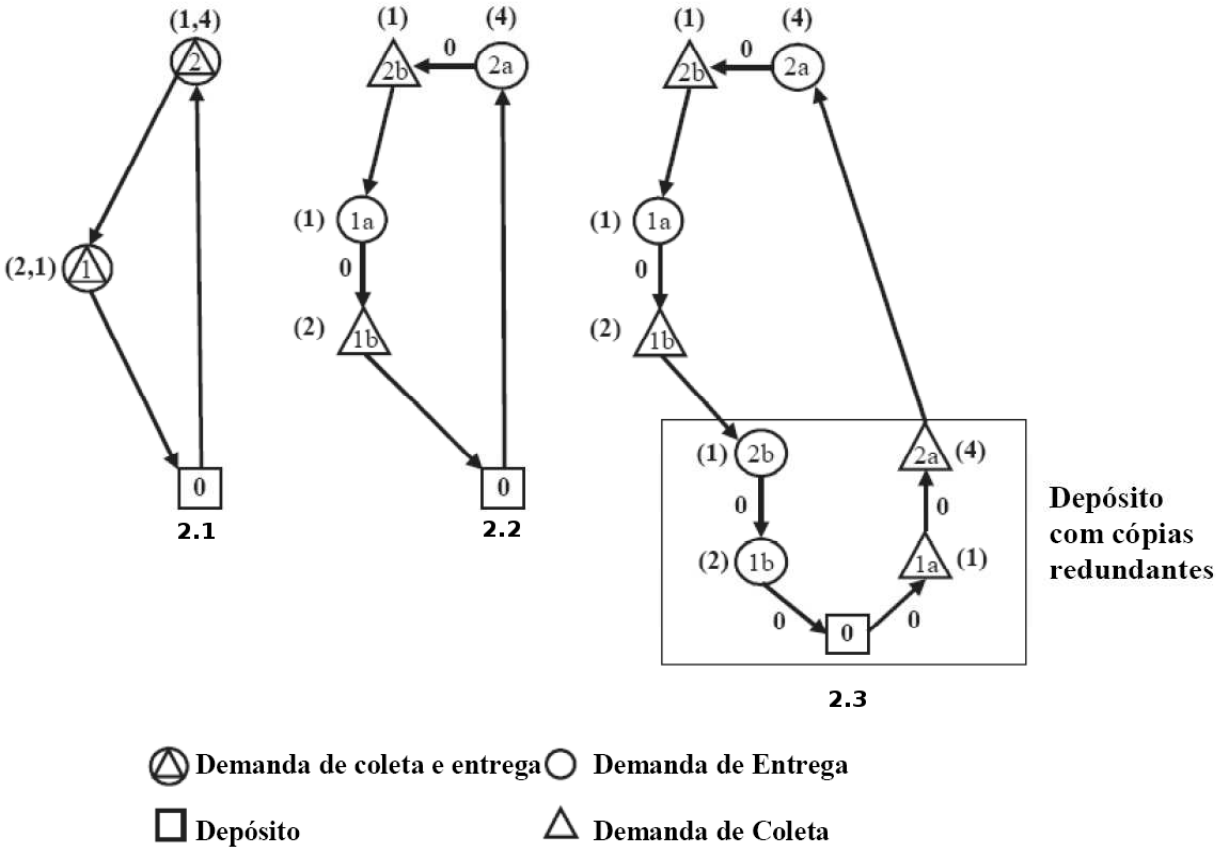


Figura 2.2: Redução do VRPSPD (2.1) em VRPPD(2.2) e MDARP(2.3). Obs: O número entre parênteses (i,j) indica a demanda de coleta (i) e entrega (j) em cada consumidor (Montané e Galvão, 2006).

Para fazer a transformação do VRPSPD em um VRPPD ou MDARP, basta transformar os nós com coleta e entrega em dois nós distintos, um com demanda de coleta e outro com entrega, dobrando o número de consumidores. Para o MDARP é necessário, ainda, criar cópias artificiais desses consumidores no depósito, pois neste problema os veículos devem partir e retornar ao depósito vazios. A distância entre os nós (consumidores) artificiais é igual a zero (Montané e Galvão, 2006).

Segundo Montané e Galvão (2006), existem poucas referências na literatura sobre VRPSPD. Algumas heurísticas construtivas, heurísticas baseadas em metaheurísitcas e algoritmos exatos foram utilizados para resolução desse problema. A partir de uma revisão na literatura, verifica-se que os estudos se tornaram constantes a partir de 2002, apesar do problema ter

sido proposto inicialmente por Min (1989).

Angelelli e Mansini (2002) propõem um algoritmo exato para o problema VRPSPD With Time Windows (VRPSPDTW), utilizando um método *Branch-and-Price*. Até então, nenhuma abordagem do problema utilizando *Branch-and-Price* tinha sido feita. O autor implementou tal método baseando-se na formulação de *set covering* para o problema mestre. A relaxação do problema de caminho mínimo com janela de tempo e das restrições de capacidade foram utilizadas como *pricing problem*.

Bianchessi (2003, 2004); Bianchessi e Righini (2007) apresentam um conjunto de heurísticas para solução do problema e ainda uma abordagem utilizando a metaheurística Busca Tabu.

Montané e Galvão (2006) apresentam duas heurísticas para solucionar o problema. Uma delas é o particionamento de rota e outra é uma adaptação do algoritmo proposto por Gillett e Miller (1974). Posteriormente, os autores apresentam uma abordagem baseada na metaheurística Busca Tabu na qual utiliza as heurísticas propostas anteriormente para encontrar uma solução inicial viável.

Vural e Çatay (2003) apresentam a primeira implementação de um algoritmo genético para resolução do problema, buscando minimizar a distância trafegada de cada rota. Recentemente, Dell'Amico et al. (2005) apresentaram um novo algoritmo *Branch-and-Price* para resolver o problema VRPSPD. Nesse trabalho, os autores mostram a viabilidade de resolver o VRPSPD utilizando o método e, ainda, comparam duas formas de resolver o *pricing problem*. Uma delas utilizando programação dinâmica e outra relaxando o espaço de estados.

Além da Busca Tabu e Algoritmo Genético, uma outra metaheurística encontrada na literatura para resolver o VRPSPD é a Colônia de Formiga proposta por Çatay (2006).

2.3.1 Definição Formal do Problema

Dado um conjunto de consumidores, cada qual com uma demanda p_i de coleta e d_i de entrega por produtos, e um depósito onde serão armazenados os produtos coletados nos consumidores e onde estão armazenados os produtos que deverão ser entregues a eles. Dado ainda $\overline{k}$ veículos de capacidade Q. O objetivo do problema é definir rotas para os veículos de forma que eles deverão partir do depósito e atender as demandas p_i e d_i dos consumidores, minimizando os custos de transporte. Seja $G = (V_0, E)$ um grafo direcionado, tal que $V_0 = \{v_0, v_1, ..., v_n\}$ é o conjunto dos vértices e $E = (v_i, v_j) : v_i, v_j \in V_0$ o conjunto de arestas. O vértice v_0 representa o depósito, sendo este a base de uma frota composta por $\overline{k}$ veículos homogêneos de capacidade Q, enquanto os vértices remanescentes correspondem aos consumidores. Cada consumidor v_i tem uma demanda não negativa p_i de coleta e d_i de entrega, sendo $p_0 = 0$ e $d_0 = 0$. Cada aresta (v_i, v_j) está associada a um valor não negativo c_{ij} que

representa a distância, tempo ou custo entre os consumidores. O Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas consiste em determinar um conjunto de rotas que deverão ser percorridas pelos veículos que minimize os custos de transporte e respeite as seguintes condições:

1. Cada rota começa e termina no depósito (v_0);
2. Toda cidade de $V_0 - v_0$ é visitada somente uma vez por somente um veículo;
3. A carga transportada não deve superar a capacidade Q do veículo;
4. Todas as demandas de todos os consumidores devem ser satisfeitas.

2.3.2 Formulação Matemática

A formulação matemática apresentada nesse trabalho foi proposta por Montané e Galvão (2006).

$$Minimize \sum_{k=1}^{\overline{k}} \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij}^{k} \tag{2.1}$$

Sujeito a

$$\sum_{i=0}^{n} \sum_{k=1}^{\overline{k}} x_{ij}^{k} = 1, \qquad j = 1, ..., n, \tag{2.2}$$

$$\sum_{i=0}^{n} x_{ij}^{k} - \sum_{i=0}^{n} x_{ji}^{k} = 0, \qquad j = 0, ..., n \; e \; k = 0, ..., \overline{k}, \tag{2.3}$$

$$\sum_{j=1}^{n} x_{0j}^{k} \leq 1, \qquad k = 1, ..., \overline{k}, \tag{2.4}$$

$$\sum_{i=0}^{n} y_{ji} - \sum_{i=0}^{n} y_{ij} = p_j, \qquad \forall \; j \neq 0, \tag{2.5}$$

$$\sum_{i=0}^{n} z_{ij} - \sum_{i=0}^{n} z_{ji} = d_j, \qquad \forall \; j \neq 0, \tag{2.6}$$

$$y_{ij} + z_{ij} \leq Q \sum_{k=1}^{\overline{k}} x_{ij}^{k}, \qquad i, j = 0, ..., n, \tag{2.7}$$

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

$$y_{ij} \geq 0, \qquad i, j = 0, ..., n, \tag{2.9}$$

$$z_{i,j} \geq 0, \qquad i, j = 0, ..., n \tag{2.10}$$

Sendo,

- n número total de consumidores;
- c_{ij} distância entre os consumidor i e j;
- p_j demanda de coleta do consumidor j, sendo $j = 1, 2, 3, ..., n$;
- d_j demanda de entrega do consumidor j, sendo $j = 1, 2, 3, ..., n$;
- Q capacidade do veículo;
- $\overline{k}$ número máximo de veículos;
- y_{ij} total de cargas transportadas na aresta (i, j) referentes as demandas de coletas atendidas do depósito ao consumidor i (inclusive);
- z_{ij} total de cargas transportadas na aresta (i, j) referentes as demandas de entrega a serem atendidas a partir do consumidor i (exclusive) até o depósito;
- Variável de Decisão:

$$x_{ij}^{k} = \begin{cases} 1 & \text{, se arco } (i, j) \text{ faz parte da rota trafegada pelo veículo } k \\ 0 & \text{, caso contrário} \end{cases}$$

A função objetivo busca minimizar a distância percorrida pelos veículos, sujeita as seguintes restrições:

- (2.2): Cada ponto de demanda deve ser visitado por um único veículo.
- (2.3): Conservação de fluxo.

- (2.4): $\overline{k}$ veículos, no máximo, podem ser utilizados.
- (2.5): Demandas de coleta devem ser satisfeitas em cada consumidor.
- (2.6): Demandas de entrega devem ser satisfeitas em cada consumidor.
- (2.7): As demandas devem ser transportadas nos arcos incluídos na solução e ainda impõe um limite para a carga total transportada pelo veículo. Os veículos são homogêneos e possuem a mesma capacidade.
- (2.8): Restrição de integralidade.
- (2.9 e 2.10): Restrição de não-negatividade para demandas (coleta e entrega).

Montané e Galvão (2006) também abordam problemas que limitam a distância máxima percorrida pelos veículos. Neste trabalho esta restrição foi retirada da formulação.

Capítulo 3

Heurísticas Construtivas

Devido a complexidade computacional do Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas, o desenvolvimento e análise de heurísticas polinomiais para encontrar soluções é extremamente importante. Este capítulo aborda algumas heurísticas para esse problema. Em geral, os trabalhos encontrados na literatura para o VRP que utilizam métodos exatos buscam reduzir a distância total percorrida. Já os métodos heurísticos visam minimizar o número total de veículos utilizados (Alvarenga, 2005).

Dell'Amico et al. (2005) apresenta um algoritmo exato baseado na técnica do *Branch-and-Price* para resolver problemas com 20 e 40 consumidores. Em problemas reais, esse número pode ser considerado muito pequeno e passa a ser inviável optar por esse tipo de abordagem. Isso tem impulsionado o estudo de métodos heurísticos para resolvê-los.

As abordagens heurísticas que priorizam o número de veículos pode acarretar num aumento exorbitante da distância trafegada. Em alguns casos, é necessário finalizar a rota e incluir uma nova mesmo que esta suporte outros consumidores. Isso justifica um estudo mais aprofundado afim de minimizar a distância trafegada.

Este capítulo apresenta uma comparação entre heurísticas encontradas na literatura e heurísticas propostas neste trabalho objetivando a redução da distância total percorrida pelos veículos. Não foram consideradas restrições relacionadas ao número máximo de veículos e distância máxima percorrida.

O capítulo inicia apresentando nas Seções 3.1 e 3.2 algumas heurísticas para solucionar o Problema do Caixeiro Viajante e o Problema do Caixeiro Viajante com Coleta e Entrega Simultâneas, respectivamente. Estes algoritmos são utilizados em algumas heurísticas para o VRPSPD para encontrar uma solução viável para estes problemas.

Após apresentadas as heurísticas para o TSP e TSPSPD, as seções seguintes descrevem as heurísticas implementadas neste trabalho para resolver o VRPSPD. A Seção 3.3 aborda

heurísticas para o VRPSPD baseadas no modelo denominado Rotear e Agrupar. As heurísticas baseadas no modelo Dividir e Rotear são mostradas na Seção 3.4. A Seção 3.5 apresenta novas propostas para o método Dividir e Rotear que efetuam a divisão dos nós utilizando algoritmos para encontrar Árvores Geradoras Mínimas (AGM) [1]. A Seção 3.6 apresenta uma heurística baseada em inserção proposta por Dethloff (2001). Encerrando o capítulo, os resultados computacionais e conclusões são apresentadas na Seção 3.7.

3.1 Heurísticas para o Problema do Caixeiro Viajante

O problema do caixeiro viajante tradicional consiste em encontrar um circuito hamiltoniano de menor custo em um grafo completo não direcionado. Um circuito hamiltoniano é um caminho fechado que passa por todos os vértices do grafo sem repetição (Goldbarg e Luna, 2000).

Nessa seção serão apresentadas duas heurísticas simples e rápidas denominadas: Vizinho mais Próximo e Inserção da Cidade mais Próxima (Lawler et al., 1985).

3.1.1 Vizinho Mais Próximo

A heurística gulosa Vizinho Mais Proximo (VP) é muito utilizada por ser bastante simples e rápida. Os consumidores são inseridos na rota de acordo com o seu vizinho mais próximo, ou seja, a cada passo é adicionado na rota o vizinho mais próximo do último consumidor inserido na mesma.

O Algoritmo 3.1 mostra detalhadamente o procedimento do algoritmo Vizinho mais Próximo. O Algoritmo recebe como entrada um grafo completo contendo os consumidores e o depósito e retorna o um trajeto contendo estes nós.

Algoritmo 3.1: Algoritmo Vizinho Mais Próximo

```
1  VP(G {Grafo completo com os consumidores e depósito})
2
3      r = Criar nova rota
4      Inserir depósito na rota r
5      últimoVérticeInserido = depósito
6
7      Enquanto existir consumidor não roteado
8          vértice = Buscar vértice mais próximo de últimoVérticeInserido
9          Inserir vértice na rota na posição seguinte ao últimoVérticeInserido
10         últimoVérticeInserido = vértice
11
```

[1] Uma Árvore Geradora Mínima de um grafo valorado $G(V, A)$ é um subgrafo $G'(V, A')$ tal que a soma dos pesos de A' é mínima (Goldbarg e Luna, 2000)

```
12      Inserir depósito no final da rota r
13
14      Retornar r
```

3.1.2 Inserção da Cidade Mais Próxima

Algoritmo 3.2: Algoritmo Inserção da Cidade Mais Próxima

```
1  CP(G {Grafo completo com os consumidores e depósito})
2
3      r = Criar nova rota
4      Inserir depósito na rota r
5
6      Enquanto existir consumidor não roteado
7          vértice_i = Busca vértice mais próximo da rota r
8          posição_x = Buscar posição de menor custo
9                      para inserir vértice_i na rota r
10         Inserir vértice_i na posição_x da rota
11
12     Inserir depósito no final da rota r
13
14     Retornar r
```

Na heurística Inserção da Cidade Mais Próxima (CP), a cada passo é selecionado o consumidor mais próximo da rota para ser inserido na mesma. Essa escolha é feita comparando a distância entre cada consumidor não roteado cm cada um pertencente a rota. Posteriormente, todas as posições viáveis para inserção do nó são analisadas verificando em qual delas a distância trafegada é mínima. Esse processo é feito até que todos os consumidores estejam incluídos.

O Algoritmo 3.2 mostra detalhadamente o procedimento do algoritmo Inserção da Cidade Mais Próxima. O Algoritmo recebe como entrada um grafo completo contendo os consumidores e o depósito e retorna o um trajeto contendo estes nós.

3.2 Heurísticas para o Problema do Caixeiro Viajante com Coleta e Entrega Simultâneas

O problema do caixeiro viajante com coleta e entrega consiste em encontrar um circuito hamiltoniano de menor custo em um grafo completo não direcionado que satisfaça as demandas de coleta e entrega dos consumidores e não extrapole a capacidade do veículo.

Nessa seção serão apresentadas duas heurísticas denominadas Nó Inicial e Inserção Mais Barata propostas por Mosheiov (1994).

3.2.1 Nó Inicial

A heurística Nó Inicial (NI) obtém um trajeto resolvendo o TSP incluindo todos os nós que representam os consumidores, exceto o depósito. Após definido o trajeto, busca-se o consumidor i no qual o veículo contém a carga máxima entre todos os pontos da rota. Se essa carga máxima não extrapola a capacidade do veículo, então o trajeto é viável e a rota é retornada. Caso contrário, um novo trajeto é definido no qual o consumidor $i+1$ ocupa a primeira posição e o consumidor i a última posição do novo trajeto. A ordem dos demais é mantida. Após definido um trajeto viável, o depósito é inserido entre o primeiro e o último consumidor da rota.

O algoritmo nem sempre retorna uma solução para o TSPSPD. Dependendo do trajeto definido não haverá possibilidade de encontrar uma solução viável independente do vértice que irá iniciar ou finalizar a rota. Para solucionar este problema, alguns consumidores são removidos da rota e inseridos na melhor posição viável.

O Algoritmo 3.3 mostra o procedimento do algoritmo Nó Inicial.

Algoritmo 3.3: Algoritmo Nó Inicial

```
1  NI(G {Grafo completo com os consumidores e depósito},
2     Q {Capacidade dos veículos},
3     D {Conjunto de Demandas dos consumidores})
4
5     Rota r = AlgoritmoTSP(G)
6     i = Posição da rota r no qual o
7         veículo está com carga Máxima (CMAX)
8     CMAX = Carga máxima do veículo na rota r
9
10    Enquanto CMAX > Q
11          r = Ajustar rota para que: inicie no vértice i+1 e
12              termine no vértice i. As posições dos demais
13              vértices são mantidas
14          i = Posição da rota no qual o
15              veículo está com carga Máxima (CMAX)
16          CMAX = Carga máxima do veículo na rota r
17
18    Inserir depósito entre o primeiro e o último consumidor do trajeto
19
20    Retornar r
```

3.2.2 Inserção Mais Barata

A heurística Inserção Mais Barata (IMB) obtém um trajeto pela solução do TSP incluindo todos os nós (consumidores e depósito) no qual a demanda de entrega é maior ou igual a demanda de coleta. Em seguida, o consumidor i mais próximo da rota é selecionado para ser

incluído na mesma. Deve-se analisar a cada inserção se a capacidade do veículo está sendo extrapolada e buscar sempre minimizar a distância trafegada. Esse processo é repetido até que todos os nós estejam inseridos na rota. O Algoritmo 3.4 mostra o procedimento da heurística Inserção Mais Barata.

A heurística IMB apresenta o mesmo problema do algoritmo descrito na seção anterior. A forma como os consumidores estão dispostos no trajeto definido pelo algoritmo para o TSP pode impossibilitar a inserção dos demais consumidores, assim uma solução viável para o TSPSPD pode não ser encontrada.

Algoritmo 3.4: Algoritmo Inserção Mais Barata

```
1  IB(G {Grafo completo com os consumidores e depósito},
2     Q {Capacidade do veículo},
3     D {Conjunto de Demandas dos consumidores})
4
5     Grupo g = Selecionar todos os consumidores no qual a
6               demanda de entrega é maior ou igual a demanda de coleta
7     Rota r = AlgoritmoTSP(g)
8
9     Enquanto existir consumidor fora da rota
10          Buscar consumidor com menor custo de inserção na rota r
11          Inserir consumidor na rota r
12
13    Retornar r
```

3.3 Heurística Rotear e Dividir

Nesta seção e nas seções seguintes serão apresentadas heurísticas para solucionar o Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas que é o foco deste trabalho.

A heurística da literatura Rotear e Dividir, também conhecida como Particionamento de Rota, é baseada no método proposto por Beasley (1983) para o problema de Roteamento de Veículos (VRP). Como o próprio nome diz, ela é composta de duas fases:

1. Definição de um ciclo inicial a partir de um algoritmo para o Problema do Caixeiro Viajante (TSP)
2. Inserção dos vértices na rota respeitando a ordem definida na primeira fase e verificando sempre a capacidade do veículo.

Quando a capacidade do veículo é extrapolada, o algoritmo pode simplesmente finalizar a

rota atual e iniciar outra ou pular para o próximo vértice do ciclo definido pelo TSP. Ambas estratégias foram implementadas e definidas como:

- Rotear e Dividir
- Rotear e Dividir Máximo

Elas são descritas na Subseções 3.3.1 e 3.3.2.

3.3.1 Rotear e Dividir

A heurística construtiva Rotear e Dividir (RD) inicia definindo um trajeto com todos os nós (consumidores e depósito) utilizando uma solução do TSP. Seguindo o trajeto obtido por esta solução, os nós vão sendo inseridos em uma rota. Caso a inserção de algum deles extrapole a capacidade do veículo, uma nova rota é criada continuando o processo até que o último nó da solução do TSP seja alcançado

Algoritmo 3.5: Algoritmo Rotear e Dividir

```
1  RD(G {Grafo completo com os consumidores e depósito},
2     Q {Capacidade dos veículos},
3     D {Demandas dos consumidores})
4
5     trajeto = AlgoritmoTSP(G)
6     Inicializar rota r
7     Inserir depósito na rota r
8     Percorrer trajeto consumidor por consumidor
9         Se a inserção do consumidor_i na rota r
10         não extrapola a capacidade do veículo
11              Inserir consumidor_i na rota r
12              Remover consumidor_i do trajeto
13         senão
14              Inserir depósito na rota r
15              Inserir r no conjunto de rotas R
16              Inicializar rota r
17              Inserir depósito na rota r
18    Retornar R
```

A Figura 3.1 mostra os passos da heurística que inicialmente define uma rota com todos os nós usando uma solução para o problema do caixeiro viajante tradicional. Na fase de divisão, o nó circulado na figura é aquele que extrapola a capacidade do veículo, portanto, uma nova rota é iniciada partindo deste nó.

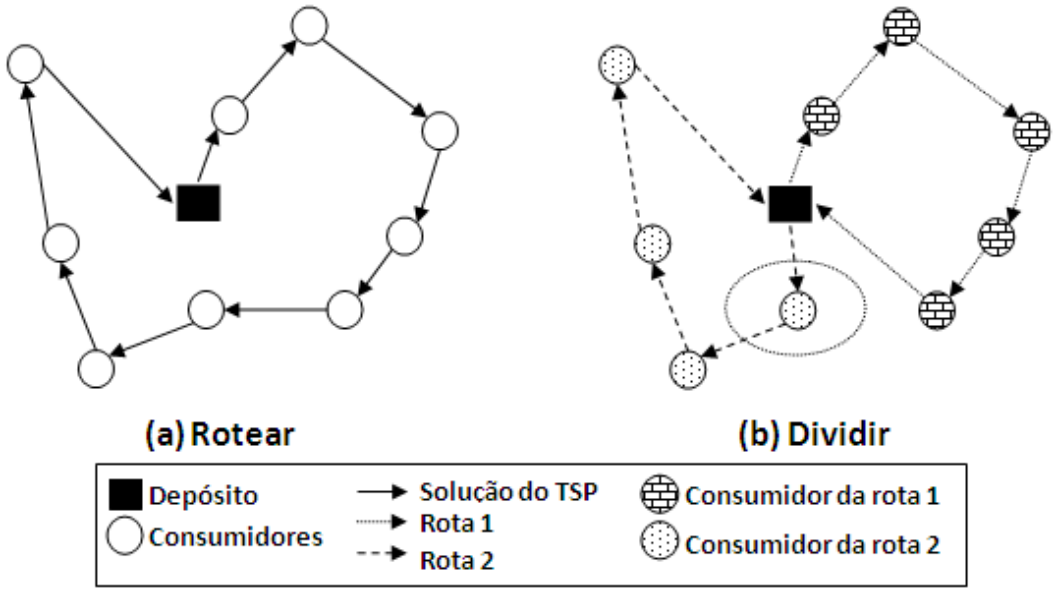


Figura 3.1: Exemplo do algoritmo Rotear e Dividir

3.3.2 Rotear e Dividir Máximo

Na segunda versão implementada da heurística Rotear e Dividir, denominada Rotear e Dividir Máximo (RDM) o processo de divisão das rotas difere do método descrito na subseção anterior.

A fase de roteamento consiste em definir um trajeto inicial com todos os nós (consumidores e depósito) utilizando uma solução do TSP. Este trajeto define a ordem no qual o consumidores devem ser inseridos na rota. Sempre que um deles, ao ser inserido, extrapolar a capacidade do veículo, este é desconsiderado e o processo continua com o consumidor seguinte até alcançar o último do trajeto. Quando isso ocorre, a rota é finalizada e uma nova é criada para que o processo se repita com todos aqueles consumidores que foram desconsiderados e ainda não foram roteados, mantendo sempre a ordem de inserção definida no trajeto inicial. O algoritmo termina quando todos os consumidores foram inseridos em alguma rota.

Algoritmo 3.6: Algoritmo Rotear e Dividir Máximo

```
1  RDM(G {Grafo completo com os consumidores e depósito},
2      Q {Capacidade dos veículos},
3      D {Demandas dos consumidores})
4
5      trajeto = AlgoritmoTSP(G)
6      Enquanto existir consumidor não roteado
7          Inicializar rota r
8          Inserir depósito na rota r
9          Percorrer trajeto consumidor por consumidor
10             Se a inserção do consumidor_i na rota r
11              não extrapola a capacidade do veículo
```

```
12              Inserir consumidor_i na rota r
13              Remover consumidor_i do trajeto
14          Inserir depósito na rota r
15          Inserir r no conjunto de rotas R
16
17      Retornar R
```

A Figura 3.2 e o Algoritmo 3.6 mostram os passos da heurística Rotear e Dividir Máximo. A Figura 3.2 (a) define uma rota com todos os nós usando uma solução para o problema do caixeiro viajante tradicional. A Figura 3.2 (b) apresenta a fase de divisão do trajeto em 2 rotas, sendo que os nós circulados são aqueles que extrapolaram a capacidade do veículo na tentativa de inserí-los na rota 1, portanto, uma nova rota é criada para que estes nós sejam inseridos na mesma.

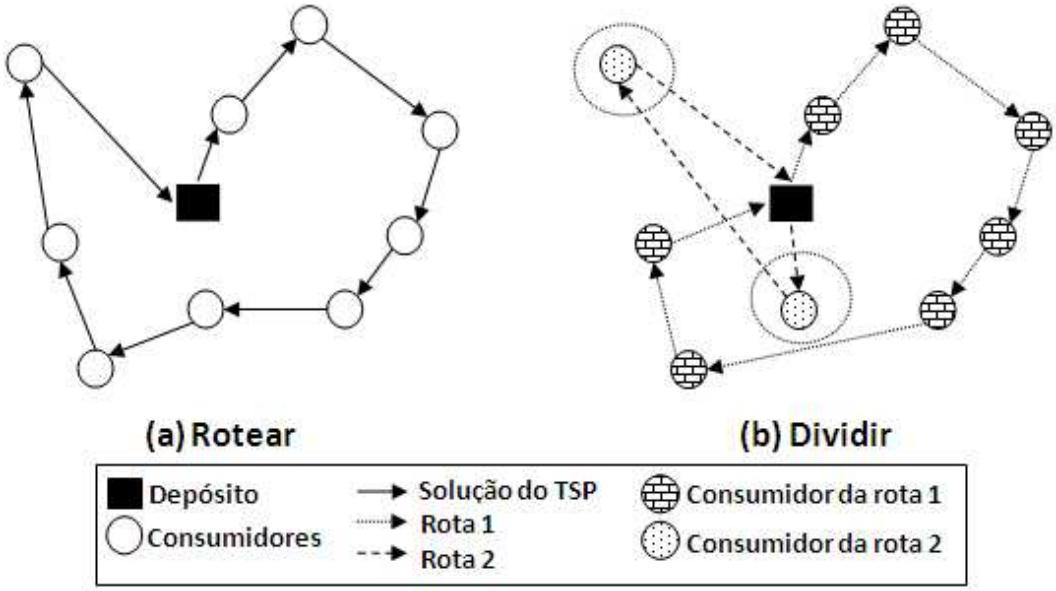


Figura 3.2: Exemplo do algoritmo Rotear e Dividir Máximo

3.4 Heurística Dividir e Rotear

A heurística Dividir e Rotear é baseada no método proposto por Min (1989) para solucionar o VRPSPD. Como o próprio nome indica, ela é composta por duas fases:

- **Divisão:** Separar os consumidores em m grupos.
- **Roteamento:** Definir uma rota para cada um dos m grupos de consumidores resolvendo ou TSP ou TSPSPD.

As estratégias de agrupamento são baseadas no conceito de demanda líquida (ND, do inglês *Net Demand*). A demanda líquida do consumidor i, $nd(i)$, é a diferença entre a demanda de coleta e a demanda de entrega do mesmo. Se $nd(i)$ for:

- $nd(i) < 0$**:** a carga do veículo será reduzida em $nd(i)$ após visitar o consumidor i.
- $nd(i) > 0$**:** a carga do veículo será incrementada em $nd(i)$ após visitar o consumidor i.

Para a primeira fase (fase de divisão) foram definidas duas estratégias: Dividir e Rotear com Carga Mínima e Dividir e Rotear com Carga Máxima. Cada estratégia exige uma forma diferente de roteamento na segunda fase.

3.4.1 Dividir e Rotear com Carga Mínima

Dado um conjunto de demandas, o algoritmo Dividir e Rotear com Carga Mínima (DR_MIN) divide os consumidores em grupos de acordo com a menor carga possível (MINLOAD) do veículo. A viabilidade dos grupos é analisada considerando que o veículo atenderá primeiramente todos os consumidores com demanda líquida negativa ($nd(i) < 0$) e, em seguida, os demais. Se, nesta ordem, a capacidade do veículo não é extrapolada, então existe pelo menos uma rota viável para atender a todos os consumidores deste grupo.

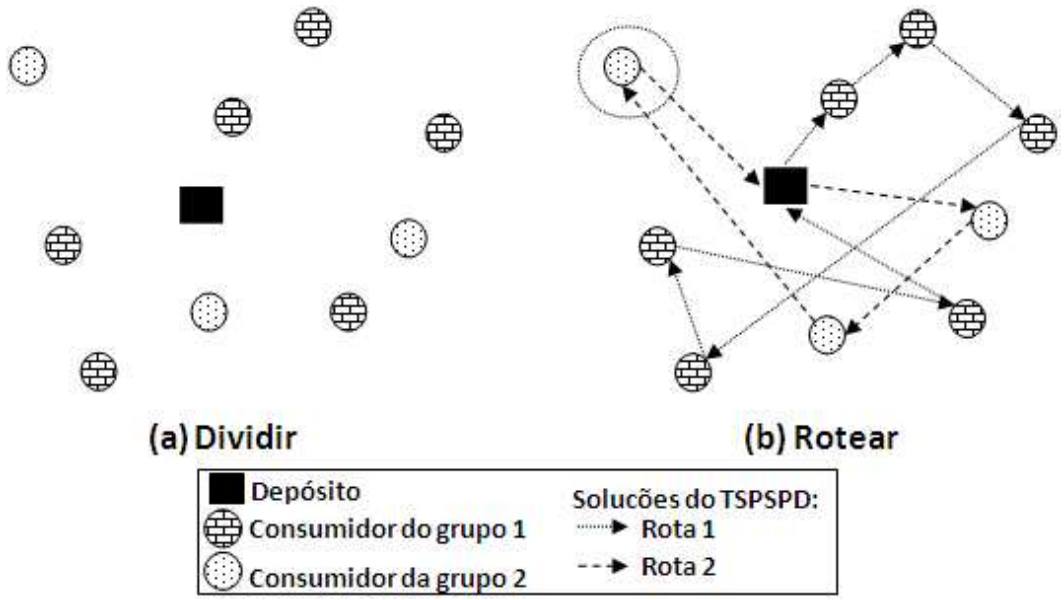


Figura 3.3: Exemplo do algoritmo Dividir e Rotear com Carga Mínima

O algoritmo seleciona consumidores para serem incluídos num determinado grupo, verificando, antes de cada inclusão, se a capacidade do veículo não é extrapolada. Se MINLOAD

é maior a capacidade do veículo, a inserção não é efetivada. Caso contrário, o consumidor é inserido no grupo e um outro é selecionado para dar continuidade ao processo.

Quando o grupo estiver saturado e não suportar a inserção de outros consumidores, o mesmo é finalizado e um novo grupo é inicializado. O processo é repetido até que todos os consumidores estejam contidos em algum grupo.

Após a divisão dos consumidores em grupos, a fase de roteamento aplica um algoritmo para solucionar o TSPSPD em cada grupo. A Figura 3.3 e o Algoritmo 3.7 mostram as duas fases do algoritmo. Primeiramente, os nós que representam os consumidores são divididos em dois grupos de acordo com MINLOAD e posteriormente define-se uma rota para cada grupo.

Algoritmo 3.7: Algoritmo Dividir e Rotear com Carga Mínima

```
1  DR_MIN(G {Grafo completo com os consumidores e depósito},
2         Q {Capacidade dos veículos},
3         D {Demandas dos consumidores})
4
5     Enquanto existir consumidor não pertencente a algum grupo
6        Grupo g = Criar um grupo para inserir os consumidores
7        Inserir depósito em g
8        Para cada consumidor i não pertencente a algum grupo
9             Consumidor i = Selecionar um consumidor
10            Calcular MINLOAD do grupo g + consumidor i
11            Se MINLOAD < Q
12                  Inserir i em g;
13
14    Para cada grupo g criado
15          Rota r = AlgoritmoTSPSPD(g);
16          Inserir r no conjunto de rotas R
17
18    Retornar R
```

O objetivo principal desse algoritmo é inserir o maior número possível de nós no grupo, reduzindo o número de veículos. A segunda fase utiliza um algoritmo para o TSPSPD para gerar uma rota em cada grupo, respeitando a capacidade do veículo e tentando minimizar a distância trafegada.

3.4.2 Dividir e Rotear com Carga Máxima

O algoritmo Dividir e Rotear com Carga Máxima (DR_MAX) utiliza o mesmo procedimento do algoritmo anterior, mas analisa a viabilidade dos grupos de acordo com a maior carga possível (MAXLOAD) do veículo. A maior carga possível do veículo é obtida considerando que o veículo atenderá primeiramente os consumidores com demanda líquida positiva ($nd(i) > 0$) e, em seguida, os demais. Se os consumidores forem atendidos nesta ordem e a

capacidade do veículo não é extrapolada em nenhum ponto da rota, então qualquer trajeto definido para esse conjunto de consumidores é viável.

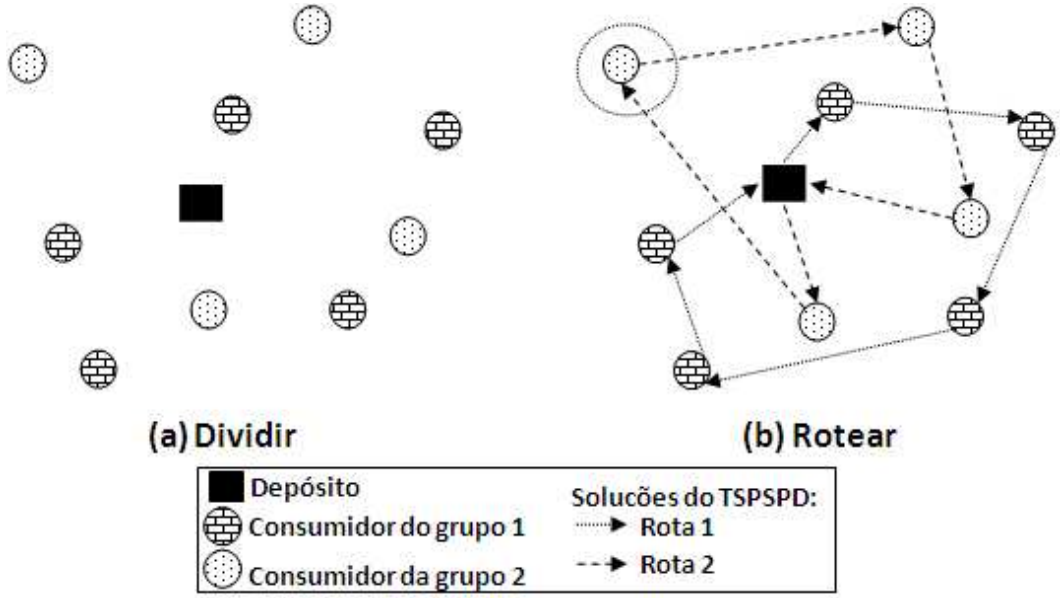


Figura 3.4: Exemplo do algoritmo Dividir e Rotear com Carga Máxima

Assim como a heurística anterior, a cada passo um consumidor é selecionado para ser inserido no grupo. Antes de cada inclusão, calcula-se a MAXLOAD deste grupo e verifica-se a viabilidade do grupo, ou seja, se MAXLOAD é maior que a capacidade do veículo. Caso isso ocorra, a inserção não é efetuada e um novo consumidor é selecionado dando continuidade ao procedimento.

Algoritmo 3.8: Algoritmo Divisão de Rotas com Carga Máxima

```
1  DR_MAX(G {Grafo completo com os consumidores e depósito},
2         Q {Capacidade dos veículos},
3         D {Demandas dos consumidores})
4
5     Enquanto existir consumidor não pertencente a algum grupo
6        Grupo g = Criar um grupo para inserir os consumidores
7        Inserir depósito em g
8        Para cada consumidor i não pertencente a algum grupo
9             Consumidor i = Selecionar um consumidor aleatoriamente
10            Calcular MAXLOAD do grupo g + consumidor i
11            Se MAXLOAD < Q
12                  Inserir i em g;
13
14    Para cada grupo g criado
15          Rota r = AlgoritmoTSP(g);
16          Inserir r no conjunto de rotas R
17
18    Retornar R
```

Quando o grupo não suportar a inclusão de outros consumidor, o mesmo é finalizado e um novo é criado. O processo então é repetido afim de inserir os consumidores restantes.

A viabilidade dos grupos neste algoritmo é analisada pelo pior caso e, então, qualquer trajeto definido é viável. Assim, a fase de roteamento consiste em aplicar um algoritmo para TSP em cada grupo. A Figura 3.4 e o Algoritmo 3.8 mostram as duas fases do algoritmo. Os nós que representam os consumidores são selecionados e divididos em dois grupos de acordo com MAXLOAD. Este algoritmo prioriza a inserção de um maior número de consumidores nos grupos, afim de minimizar o número de veículos. A redução da distância trafegada é considerada na segunda fase quando um algoritmo para o TSP é aplicado em cada grupo.

3.5 Divisão por Árvores Geradoras Mínimas

A Divisão por Árvores Geradoras Mínimas, proposta neste trabalho para solucionar o VRPSPD, é baseada no método Dividir e Rotear com Carga Máxima, sendo que a divisão dos nós é feita através de árvores geradoras mínimas. Não se conhece, até então, nenhum trabalho que utilize esse método numa heurística para o VRPSPD.

As Subseções 3.5.1, 3.5.2 e 3.5.3 apresentam três algoritmos que utilizam a idéia dos algoritmos de Prim e Kruskal (detalhes em Cormen et al. (2001)) na fase de divisão dos nós.

Assim como o algoritmo Dividir e Rotear com Carga Máxima (Subseção 3.4.2), o algoritmo Dividir com Árvores Geradoras Mínimas visa dividir os consumidores em grupos de acordo com a maior carga possível (MAXLOAD) do veículo. Dessa forma, a fase de roteamento é feita utilizando uma solução do TSP. A Figura 3.5 mostra os passos dessa heurística.

3.5.1 Divisão com Prim

Na heurística Divisão com Prim (DIV_PRIM) são criadas Árvores Geradoras Mínimas seguindo o algoritmo de Prim. O processo de criação da árvore ocorre da seguinte forma: um nó "semente"é escolhido para iniciar uma árvore. Este nó vai dar início a uma Árvore Geradora Mínima, seguindo o método de Prim. A árvore é finalizada quando a MAXLOAD do conjunto de nós nela contida extrapolar a capacidade do veículo. Portanto, a cada inserção, deve-se calcular a MAXLOAD da árvore. Quando um nó não puder ser inserido, a árvore é finalizada e o processo é repetido até que todos os consumidores estejam contidos em alguma árvore.

A segunda fase do algoritmo efetua o roteamento dos nós das árvores resolvendo o TSP.

O Algoritmo 3.9 mostra resumidamente os passos desta heurística.

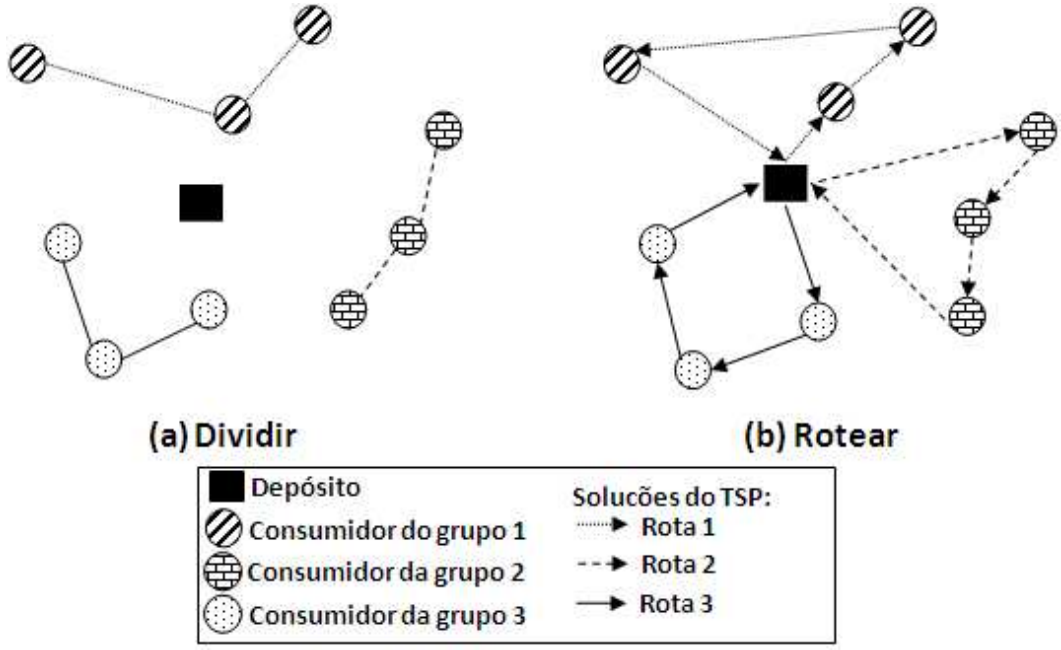


Figura 3.5: Exemplo do algoritmo Dividir com Árvores Geradoras Mínimas

Algoritmo 3.9: Algoritmo Divisão com Prim

```
1  DIV_PRIM(G {Grafo completo com os consumidores e depósito},
2           Q {Capacidade dos veículos},
3           D {Demandas dos consumidores})
4
5     Enquanto existir consumidor fora de alguma árvore
6        Consumidor i = Selecionar consumidor não pertencente a alguma árvore
7                       mais distante do depósito
8        a = Criar nova árvore
9        MAXLOAD = 0
10       Enquanto MAXLOAD < Q
11            Inserir consumidor i na árvore a
12            i = Selecionar um consumidor para ser inserido
13                na árvore seguindo os princípios do algoritmo de Prim
14            Calcular MAXLOAD da árvore a + consumidor i
15
16    Para cada árvore criada
17          Definir uma rota com os consumidores da árvore utilizando
18            um algoritmo do TSP
19          Inserir rota r no conjunto de rotas R
20
21    Retornar R
```

3.5.2 Divisão com Prim e União de Árvores

Na heurística Divisão com Prim e União de Árvores (DIV_PUA) são criadas, inicialmente, n árvores, uma para cada consumidor. Ou seja, cada árvore será composta por um único nó.

A cada passo do algoritmo, um novo consumidor (x_i) é selecionado para que sua árvore seja expandida. O processo de expansão da árvore é feito pelo método de Prim, ou seja, um consumidor (x_j) não pertencente a árvore e mais próximo dela é selecionado. Este consumidor (x_j) também está contido em uma árvore. Portanto, o algoritmo irá unir ambas árvores verificando sempre se essa união é viável (capacidade do veículo não é extrapolada segundo a MAXLOAD do conjunto de nós).

Após formados os grupos, a segunda fase do algoritmo efetua o roteamento resolvendo o TSP para cada conjunto de nós que compõem as árvores. O Algoritmo 3.10 mostra resumidamente os passos dessa heurística.

Algoritmo 3.10: Algoritmo Divisão com Prim e União de Árvores

```
1  DIV_PUA(G {Grafo completo com os consumidores e depósito},
2          Q {Capacidade dos veículos},
3          D {Demandas dos consumidores}
4          numConsumidores {número de consumidores})
5
6     Para cada consumidor pertencente a G
7          a = Criar uma nova árvore
8          Inserir consumidor na árvore a
9
10    Para i = 0 até numConsumidores
11       Consumidor_i = Selecionar um consumidor pela sua ordem de indexação
12       Consumidor_j = Selecionar um consumidor para ser inserido
13                      na árvore seguindo os princípios do algoritmo de Prim
14       Calcular MAXLOAD da árvore de i + árvore de j
15       Se MAXLOAD < Q
16          Unir árvores de i e j
17
18    Para cada árvore resultante
19         Rota r=fazer o roteamento dos consumidores da árvore resolvendo um TSP
20         Inserir rota r no conjunto de rotas R
21
22    Retornar R
```

3.5.3 Divisão com Kruskal

Na heurística Divisão com Kruskal (DIV_KRUSKAL) são criadas, inicialmente, n árvores geradoras, sendo n o número de consumidores. Posteriormente, as distâncias entre os consu-

midores são ordenadas de forma crescente. Seguindo essa ordenação, as árvores são agregadas verificando sempre a capacidade do veículo segundo sua MAXLOAD.

Após formados os grupos, a segunda fase do algoritmo efetua o roteamento utilizando um algoritmo para o TSP.

O Algoritmo 3.11 mostra resumidamente os passos desta heurística.

Algoritmo 3.11: Algoritmo Divisão com Kruskal

```
1  DIV_KRUSKAL(G {Grafo completo com os consumidores e depósito},
2              Q {Capacidade dos veículos},
3              D {Demandas dos consumidores})
4
5     Para cada consumidor pertencente a G
6          a = Criar uma nova árvore
7          Inserir consumidor na árvore a
8
9     Ordenar arestas de acordo com os seus custos
10
11    Para cada aresta(i,j)que liga árvores distintas
12       Calcular a MAXLOAD da árvore i + árvore j
13       Se MAXLOAD < Q
14          Unir árvores i e j
15
16    Para cada árvore resultante
17         Rota r=Fazer o roteamento dos consumidores da árvore resolvendo um TSP
18         Inserir rota r no conjunto de rotas R
19
20    Retornar R
```

3.6 Heurística Baseada em Inserção

Nesta seção é apresentada a Heurística Baseada em Inserção, proposta por Dethloff (2001) para solucionar o Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas.

A Heurística Baseada em Inserção (IBH, do inglês *Insertion-Based Heuristic*), utiliza o conceito "Inserção Mais Barata". Inicialmente, o algoritmo deve escolher um consumidor (semente) e define uma nova rota: depósito $\longrightarrow$ semente $\longrightarrow$ depósito

Posteriormente, é analisada a inserção de cada consumidor restante em cada posição viável da rota atual. Cada uma dessas alternativas é avaliada de acordo com alguns critérios de inserção.

Segundo Dethloff (2001), uma simples abordagem gulosa calcula apenas a distância trafegada da rota R com a inserção de um consumidor k entre os consumidores i e j, denotada

por Ψ_{DT}:

$$\Psi_{DT} = C_{ik} + C_{kj} - C_{ij}, \forall k \in R \wedge \forall (i,j) \in R \tag{3.1}$$

Algoritmo 3.12: Heurística Baseada em Inserção

```
1  IBH(G {Grafo completo com os consumidores e depósito},
2     Q {Capacidade dos veículos},
3     D {Demandas dos consumidores}
4     pesoDistância, pesoCapacidadeResidual, pesoRaio)
5
6     Enquanto existir consumidor fora de alguma rota
7
8        semente = Selecionar um consumidor aleatoriamente
9        r = Criar uma nova rota (depósito - semente - depósito)
10
11       menorCusto = nulo       {Menor custo de inserção}
12       menorConsumidor = nulo {Consumidor a ser inserido na rota}
13       menorPosição = nulo     {Posição de inserção do consumidor na rota}
14
15       continua = true {Rota comporta algum consumidor}
16       Enquanto continua
17
18             Para cada consumidor c não inserido em alguma rota
19                 Para cada posição i da rota
20
21                     Se consumidor c puder ser inserido na posição i
22
23                         distância = Calcular distância da rota após
24                                     inserir consumidor c na posição i
25                         cr = Calcular capacidade residual da rota
26                              após inserir consumidor c na posição i
27                         raio = Calcular raio de agrangência após inserir
28                                consumidor c na posição i
29                         custoInserção = pesoDistância * distância +
30                                         pesoCapacidadeResidual * cr +
31                                         pesoRaio * raio
32                         Se menorCusto = nulo ou custoInserção < menorCusto
33                             menorCusto = custoInserção
34                             menorConsumidor = c
35                             menorPosição = i
36
37             Se menorCusto não é nulo
38                 Inserir o consumidor menorConsumidor
39                  na posição menorPosição da rota r
40             Senão
41                 continua = false
42
43       Inserir r no conjunto R
44
45    Retornar R
```

Calculada a distância trafegada, a menor é escolhida. Esse processo é feito para todos os consumidores não contidos na rota e para todas as posições possíveis.

Esse critério de inserção é extremamente simples e não verifica alguns aspectos, como:

1. O efeito de uma inserção viável na carga de um veículo, ou seja, o grau de liberdade para futuras inserções.
2. Consumidores localizados remotamente podem ser deixados para ser inseridos em estágios de inserção tardios, resultando em um distância trafegada desfavorável.

O autor, então, considera três critérios para inserir um consumidor na rota atual:

- **Distância Trafegada:** abordagem gulosa. Calcula-se a distância total da rota após inserção.
- **Capacidade Residual:** indica o grau de liberdade para futuras inserções.
- **Raio de Abrangência:** distância do consumidor para o depósito. Esse critério evita que consumidores localizados remotamente sejam inseridos de forma tardia.

O custo final de uma inserção de cada nó é dado pela soma ponderada dos valores dos três critérios.

Quando nenhum consumidor puder ser inserido na rota atual, uma nova rota é criada com uma nova semente e o processo é repetido até que todas as demandas tenham sido satisfeitas.

Mais detalhes de como foram definidos os critérios podem ser encontrados em (Dethloff, 2001). O Algoritmo 3.12 mostra resumidamente os passos desta heurística.

3.7 Resultados Computacionais

As heurísticas foram testadas em instâncias com 50 a 200 consumidores, propostas por Salhi e Nagy (1999) para o VRPSPD. As instâncias possuem informações sobre as coordenadas (x,y) e as demandas de coleta e entrega de cada consumidor e o número máximo de veículos disponíveis. Algumas instâncias apresentam a distância máxima que os veículos poderão percorrer. As restrições relacionadas ao número máximo de veículos e distância máxima foram desconsideradas neste trabalho.

Os algoritmos foram implementadas em Java (JDK 1.4) e executados em um Pentium IV 2.40 GHz.

Problema	N	CP		VP	
		Custo	Tempo	Custo	Tempo
CMT01	51	495	0	555	0
CMT02	76	643	0	630	0
CMT03	101	716	2	815	0
CMT04	151	838	10	870	0
CMT05	200	938	36	997	0
CMT06	51	495	0	555	0
CMT07	76	643	0	630	0
CMT08	101	716	2	815	0
CMT09	151	838	10	870	0
CMT10	200	938	35	997	1
CMT11	121	714	4	627	0
CMT12	101	602	2	597	0
CMT13	121	714	4	627	0
CMT14	101	602	2	597	0
Médias	–	707	4,07	727	0,07

Tabela 3.1: Resultados das heurísticas Vizinho Mais Próximo e Inserção da Cidade Mais Próxima

As heurísticas propostas para o TSP, apresentadas na Seção 3.1, foram testadas em 14 instâncias, pois, nas demais, a disposição dos consumidores é a mesma variando apenas as demandas de coleta e entrega. Os dados das instâncias relacionados aos veículos e as demandas foram desconsiderados. Os resultados dos testes são apresentados na Tabela 3.1. As duas primeiras colunas mostram o nome e o número de consumidores de cada instância. As duas próximas colunas mostram a distância percorrida (D) e o tempo de execução (T) [milissegundos] apresentados pela heurística Inserção da Cidade Mais Próxima (CP). As duas últimas colunas mostram os mesmos dados da heurística Vizinho mais Próximo (VP).

Os resultados mostraram que a heurística Inserção da Cidade Mais Próxima foi melhor em 8 instâncias (57%) tendo como critério a distância trafegada. A diferença entre ambas não foi muito significativa, como mostra a última linha contendo a distância trafegada média retornada pelas heurísticas. Por outro lado, o tempo de execução da heurística CP é em média 4 milissegundos maior que o da heurística VP.

As heurísticas propostas para o TSPSPD, apresentadas na Seção 3.2, foram testadas em 28 instâncias contendo valores referentes às demanda de entrega d_i e às demanda de coleta p_i de cada consumidor. A capacidade do veículo apresentada nas instâncias são para o VRPSPD. Portanto, foi necessário estipular um novo valor para se adequar ao TSPSPD:

$MAX\{\sum d_i, \sum p_j\}$

Os resultados são apresentados na Tabela 3.2. As duas primeiras colunas mostram o nome da instância e o número de consumidores. As quatro próximas colunas mostram a

Problema	N	NI(VP)		NI(CP)		IB(CP)		IB(VP)	
		D	T	D	T	D	T	D	T
CMT01_A	51	525	0	499	0	476	0	487	0
CMT01_B	51	587	0	524	0	492	0	502	0
CMT02_A	76	709	0	674	2	631	0	651	0
CMT02_B	76	669	0	623	0	584	0	594	0
CMT03_A	101	841	1	784	7	669	0	690	2
CMT03_B	101	848	1	726	7	714	0	728	4
CMT04_A	151	856	0	907	35	800	6	843	0
CMT04_B	151	855	0	898	10	795	4	813	1
CMT05_A	200	1021	0	971	95	957	12	894	1
CMT05_B	200	1013	9	1114	105	780	592	793	589
CMT06_A	51	525	0	499	0	458	0	487	0
CMT06_B	51	587	0	524	0	455	0	462	0
CMT07_A	76	709	0	674	2	631	0	651	0
CMT07_B	76	669	0	623	0	579	0	594	0
CMT08_A	101	841	1	784	7	668	1	695	1
CMT08_B	101	848	1	726	9	711	0	741	1
CMT09_A	151	856	0	907	37	796	5	843	0
CMT09_B	151	855	0	898	10	829	8	865	1
CMT10_A	200	1021	0	971	109	871	13	894	1
CMT10_B	200	1013	6	1015	97	807	624	848	596
CMT11_A	121	745	0	758	4	603	0	616	0
CMT11_B	121	891	0	882	13	694	1	645	0
CMT12_A	101	621	0	645	10	628	1	693	2
CMT12_B	101	641	0	643	2	552	0	580	0
CMT13_A	121	745	0	758	4	687	0	616	0
CMT13_B	121	844	0	882	16	697	1	645	0
CMT14_A	101	621	0	645	8	538	1	654	3
CMT14_B	101	641	0	643	2	571	0	580	0
MÉDIAS	—	771	—	757	—	667	—	682	—

Tabela 3.2: Resultados das heurísticas Nó Inicial e Inserção Mais Barata utilizando as heurísticas Vizinho Mais Próximo (VP) e Inserção da Cidade Mais Próxima (CP)

distância percorrida (D) e o tempo de execução (T) [milissegundos] das soluções obtidas com a heurística Nó Inicial utilizando a heurística Vizinho Mais Próximo, NI(VP), e utilizando a heurística Inserção da Cidade Mais Próxima, NI(CP), para definir o trajeto. As quatro últimas colunas mostram os mesmos dados para a heurística Inserção Mais Barata utilizando a heurística Inserção da Cidade Mais Próxima, IB(CP) e Vizinho Mais Próximo, IB(VP), para definir o trajeto.

Para ambas heurísticas (IB e NI) foram feitas versões utilizando os algoritmos VP e CP. Na média apresentada, a heurística CP apresenta melhores resultados e, portanto, foi utilizada nos métodos de resolução do VRPSPD.

Os resultados mostraram a superioridade da heurística Inserção Mais Barata em 100%

das instâncias em relação a heurística Nó Inicial, tendo como critério o distância trafegada. A última linha da tabela mostra as médias das distâncias trafegadas obtidas pelos algoritmos comprovando essa superioridade.

Ambas heurísticas podem não retornar um trajeto viável, como foi descrito na Seção 3.2. Este problema foi contornado retirando alguns consumidores do trajeto e tentando inserí-los novamente na melhor posição viável. O problema foi freqüente na heurística Nó Inicial. Já a heurística Inserção Mais Barata poucas vezes utilizou-se deste artifício para retornar uma solução viável.

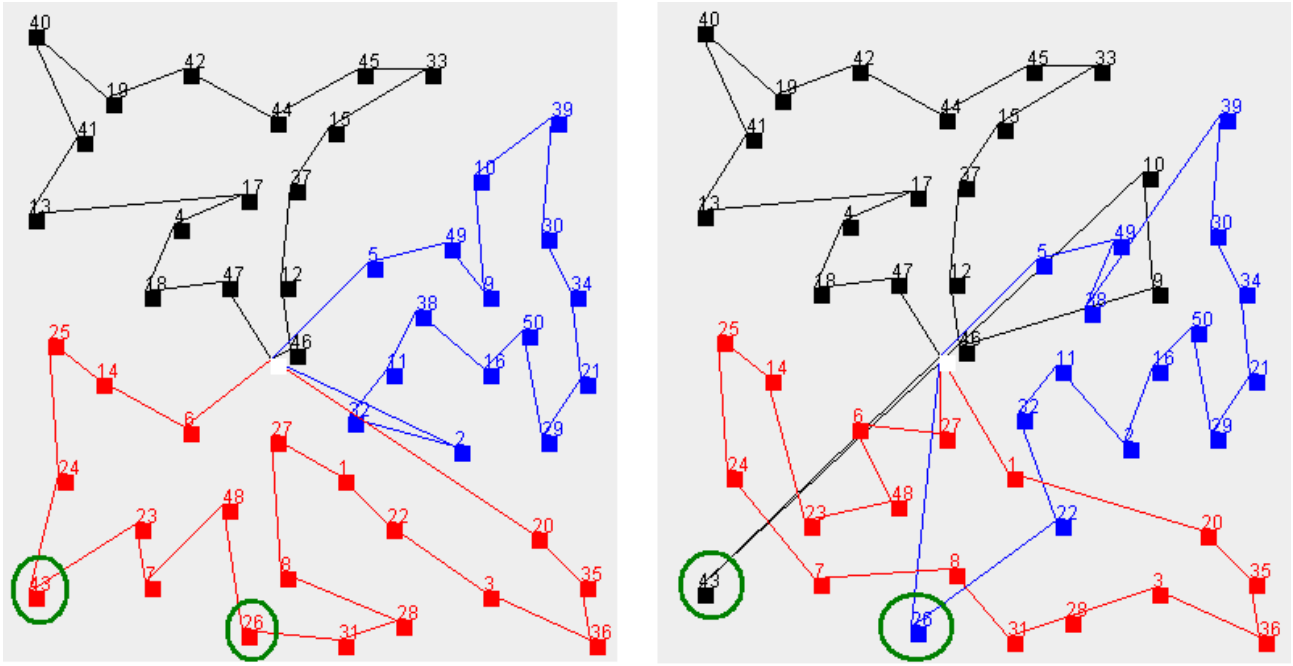
Os algoritmos apresentados nas Seções 3.3, 3.4, 3.5 e 3.6 foram testados em 28 instâncias. O objetivo do experimento é comparar as heurísticas implementadas considerando a distância trafegada pelos veículos. Os resultados obtidos são apresentados nas Tabelas 3.3, 3.4, 3.5 e 3.6. As duas primeiras colunas das tabelas mostram o nome da instância e o número de consumidores (N). Na Tabela 3.3, as demais colunas mostram o número de veículos (V), a distância percorrida (D) e o tempo de execução (T) [milissegundos] dos resultados obtidos com as heurísticas Rotear e Dividir e Rotear e Dividir Máximo. Na Tabela 3.4, as demais colunas mostram o número de veículos (V), a distância percorrida (D) e o tempo de execução (T) [milissegundos] dos resultados das heurísticas Dividir e Rotear com Carga Mínima utilizando heurística Inserção Mais Barata e heurística Nó Inicial na fase de roteamento e Dividir e Rotear com Carga Máxima.

A Tabela 3.5, apresenta os resultados das heurísticas propostas neste trabalho. A primeira coluna apresenta as instâncias utilizadas. As demais colunas mostram os resultados das heurísticas Divisão com Prim, Divisão com Árvores Geradoras e Divisão com Kruskal. A última tabela (Tabela 3.6) apresenta os resultados da heurística Baseada em Inserção.

Os primeiros experimentos foram feitos com as heurísticas Rotear e Dividir e Rotear e Dividir Máximo. Elas são heurísticas extremamente simples, comparadas com as demais. A segunda heurística (RDM) prioriza a redução do número de veículos. Dessa forma, o valor distância trafegada, para todas as instâncias, é superior aos resultados da heurística Rotear e Dividir.

A Figura 3.6(b) mostra o resultado apresentado pela heurística Rotear e Dividir Máximo para a instância CMT01_A. No caso apresentado, os vértices 43 e 26 poderiam ser inseridos na próxima rota. Como o algoritmo prioriza prioriza a redução do número de veículo, os consumidores são inserido ao máximo na rota antes de finalizá-la. Isso pode acarretar em um aumento excessivo da distância trafegada. A Figura 3.6(a) mostra como o algoritmo Rotear e Dividir se comporta nesta mesma instância. A rota é encerrada quando um vértice excede a capacidade do veículo e os nós 43 e 26 são inseridos em rotas posteriormente criadas. Isso reduz a distância trafegada consideravelmente.

Em alguns casos, o algoritmo RD pode finalizar uma rota precipitadamente. Isso pode



a) Resultado Heurística Rotear e Dividir

b) Resultado Heurística Rotear e Dividir Máximo

Figura 3.6: Exemplo de uma solução dada pelos algoritmos Rotear e Dividir e Rotear e Dividir Máximo

ocasionar rotas desnecessárias causando um aumento, no número de veículo e na distância trafegada. Os resultados da Tabela 3.3 mostram que, mesmo com esse ponto fraco, esse algoritmo aumenta em média 0,57 veículos em cada instância. A maior diferença encontrada é de 2 veículos.

Portanto, entre as duas versões do método Rotear e Dividir, conclui-se que os resultados foram melhores na heurística RD como mostra a última linha da Tabela 3.3 com a média das distâncias trafegadas e veículos utilizados. Além disso, entre todas as heurísticas ela é a mais simples e fácil de ser implementada.

Outra técnica apresentada na literatura e implementada nesse trabalho são as heurísticas do tipo Dividir e Rotear. Ambas as versões implementadas: Dividir e Rotear com Carga Mínima (DR_MIN), Dividir e Rotear com Carga Máxima (DR_MAX), priorizam minimizar o número de veículos, mas a segunda, por separar os grupos pelo pior caso (Carga Máxima Possível) tende a fechar a rota mais cedo obtendo um número maior de rotas. Por outro lado, qualquer rota é viável em relação a capacidade do veículo e isso ajuda a reduzir a distância trafegada. No momento de se definir as rotas, o DR_MAX preocupa-se apenas em reduzir a distância trafegada resolvendo um TSP.

A segunda versão divide os nós pelo melhor caso (Carga Mínima Possível) e, com isso, no momento de definir as rotas deve-se utilizar uma solução para o TSPSPD, e portanto, existe

Problema	N	RD(CP)			RDM(CP)		
		V	D	T	V	D	T
CMT01_A	51	4	675	0	4	613	0
CMT01_B	51	3	541	0	3	595	0
CMT02_A	76	8	848	2	7	1116	4
CMT02_B	76	8	847	1	7	1030	3
CMT03_A	101	6	1017	4	5	1119	6
CMT03_B	101	6	1012	4	5	1114	5
CMT04_A	151	8	1238	22	8	1341	48
CMT04_B	151	9	1253	22	8	1717	51
CMT05_A	200	12	1382	63	11	1829	182
CMT05_B	200	13	14817	65	11	1872	179
CMT06_A	51	4	669	0	4	680	0
CMT06_B	51	4	669	0	4	730	0
CMT07_A	76	8	871	1	8	1335	3
CMT07_B	76	8	870	1	8	1549	3
CMT08_A	101	6	1024	4	5	1157	7
CMT08_B	101	6	1022	5	6	1519	9
CMT09_A	151	8	1238	22	8	1519	46
CMT09_B	151	9	1260	16	8	1642	67
CMT10_A	200	12	1382	60	11	2151	181
CMT10_B	200	13	1481	59	11	2012	169
CMT11_A	121	5	1230	7	5	1373	14
CMT11_B	121	5	1093	6	5	1506	13
CMT12_A	101	7	844	5	6	1053	9
CMT12_B	101	7	873	5	6	924	10
CMT13_A	121	5	1230	8	5	1373	10
CMT13_B	121	5	1093	10	5	1506	14
CMT14_A	101	7	844	4	6	1053	9
CMT14_B	101	7	873	5	6	924	10
MÉDIAS		7.25	1031	—	6.64	1298	–

Tabela 3.3: Resultados das heurísticas Divisão de Rotas e Divisão de Rotas Máximo.

uma preocupação para que as demandas dos clientes não extrapolem a capacidade do veículo.

Um importante detalhe desta abordagem é a ausência de critérios para agrupar os consumidores. A inclusão de um critério pode melhorar significativamente os resultados.

Nas heurísticas Dividir e Rotear com Carga Mínima e Dividir e Rotear com Carga Máxima, o TSP foi resolvido utilizando a heurística Cidade Mais Próxima (CP). Para o TSPSPD a solução foi obtida mediante as heurísticas Nó Inicial (NI) e Inserção mais Barata (IB), que também necessitam de uma solução para o TSP e, neste caso, foi utilizada a heurística CP.

A Tabela 3.4 mostra os resultados das heurísticas Dividir e Rotear. Em relação as versões implementadas do algoritmo DR_MIN com a heurística NI e IB, em 92,86% dos casos, IB mostrou melhores resultados em relação a distância trafegada pelos veículos. Comparando

todas as versões do método Dividir e Rotear, a heurística DR_MAX foi melhor comparando a distância percorrida média. Apesar da heurística DR_MAX fechar rotas mais cedo a média do número de veículos não foi discrepante, comparando com as outras duas heurísticas.

Como as heurísticas do método Dividir e Rotear não possuem um critério para agrupar os nós, o principal objetivo das heurísticas que fazem o agrupamento por árvores geradoras é reduzir esse problema. Isso acarretou em uma melhoria significativa dos resultados, como mostram as Tabelas 3.4 e 3.5. O método Dividir e Rotear por Árvores Geradoras Mínimas apresentou os melhores resultado em 90% das instâncias. A heurística DIV_KRUSKAL foi melhor em 72% e a heurística DIV_PUA foi melhor em 18%.

Problema	N	DR_MIN(IBCP)			DR_MIN(NICP)			DR_MAX(CP)		
		V	D	T	V	D	T	V	D	T
CMT01_A	51	4	819	0	4	824	0	4	807	0
CMT01_B	51	4	792	0	4	921	0	4	786	0
CMT02_A	76	8	1390	0	8	1590	0	8	1344	0
CMT02_B	76	7	1315	0	7	1425	0	8	1376	0
CMT03_A	101	6	1391	2	6	1562	0	6	1297	0
CMT03_B	101	6	1271	0	6	1412	0	6	1373	0
CMT04_A	151	9	2101	2	9	2333	1	9	1906	1
CMT04_B	151	9	1946	0	9	2277	0	9	2008	0
CMT05_A	200	13	2723	0	13	3083	1	13	2734	1
CMT05_B	200	12	2489	0	12	2904	1	13	2745	1
CMT06_A	51	4	843	0	4	854	0	4	812	0
CMT06_B	51	4	784	0	4	916	0	4	776	0
CMT07_A	76	8	1370	0	8	1648	0	8	1350	0
CMT07_B	76	7	1310	0	7	1528	0	8	1368	0
CMT08_A	101	6	1361	1	6	1572	0	6	1280	0
CMT08_B	101	6	1253	0	6	1435	0	6	1362	0
CMT09_A	151	9	1995	2	9	2104	1	9	1947	1
CMT09_B	151	9	1913	1	9	2271	2	9	2013	1
CMT10_A	200	13	2657	1	13	3101	2	13	2691	2
CMT10_B	200	12	2509	1	12	2926	2	13	2728	1
CMT11_A	121	6	1206	0	6	1498	0	6	1268	0
CMT11_B	121	6	1263	1	6	1605	1	6	1213	1
CMT12_A	101	7	1764	0	7	1080	0	7	1005	0
CMT12_B	101	7	992	0	7	1212	0	7	967	0
CMT13_A	121	6	1206	0	6	1498	1	6	1268	0
CMT13_B	121	6	1263	1	6	1605	1	6	1213	1
CMT14_A	101	7	1764	0	7	1080	0	7	1005	0
CMT14_B	101	7	992	0	7	1212	0	7	967	0
MÉDIAS		7,43	1524	—	7,43	1696	—	7,57	1486	—

Tabela 3.4: Resultados das heurísticas Rotear e Dividir com Carga Mínima (heurística Inserção mais Barata e Nó Inicial na segunda fase) e Rotear e Dividir com Carga Máxima.

O método utilizado pelas heurísticas Divisão por Árvores Geradoras Mínima foi o Dividir e

Rotear com Carga Máxima devido aos melhores resultados em comparação ao método Dividir e Rotear com Carga Mínima, citados anteriormente.

Problema	N	DIV_PRIM(CP)			DIV_PUA(CP)			DIV_KRUSKAL(CP)		
		V	D	T	V	D	T	V	D	T
CMT01_A	51	4	588	0	5	583	15	3	592	71
CMT01_B	51	4	588	0	5	569	13	4	568	72
CMT02_A	76	8	880	0	11	909	38	7	838	109
CMT02_B	76	8	880	0	10	926	43	8	891	74
CMT03_A	101	6	963	1	9	959	157	6	983	755
CMT03_B	101	6	963	1	10	1012	153	5	958	995
CMT04_A	151	9	1219	4	11	1141	770	9	1090	3493
CMT04_B	151	9	1219	4	11	1123	781	9	1107	2811
CMT05_A	200	13	1517	8	19	1508	2141	12	1344	7076
CMT05_B	200	13	1517	8	18	1528	2173	12	1313	6963
CMT06_A	51	4	588	0	5	569	12	3	592	69
CMT06_B	51	4	588	0	5	585	12	4	568	67
CMT07_A	76	10	1058	0	10	879	45	7	838	102
CMT07_B	76	10	1058	0	10	904	37	8	891	74
CMT08_A	101	7	1087	1	9	962	161	6	983	752
CMT08_B	101	7	1087	1	9	947	159	5	958	1022
CMT09_A	151	10	1314	4	11	1127	771	9	1090	3518
CMT09_B	151	10	1314	4	11	1120	770	9	1107	3111
CMT10_A	200	14	1626	9	19	1509	2245	12	1344	7234
CMT10_B	200	14	1626	7	18	1513	2162	12	1313	7720
CMT11_A	121	6	1234	3	7	1083	474	6	1082	2465
CMT11_B	121	6	1234	2	7	1088	412	5	1005	3312
CMT12_A	101	8	1051	1	9	838	142	6	751	722
CMT12_B	101	8	1051	1	10	841	137	7	765	489
CMT13_A	121	6	1234	3	7	1097	412	6	1082	2553
CMT13_B	121	6	1234	4	7	1085	432	5	1005	3731
CMT14_A	101	8	1051	1	10	838	140	6	751	718
CMT14_B	101	8	1051	1	10	838	138	7	765	490
MÉDIA		8,07	1101	—	10,10	1002	—	7,07	949	—

Tabela 3.5: Resultados das heurísticas Divisão com Prim, Divisão Prim e União de Árvores e Divisão com Kruskal

Os resultados da heurísticas DIV_PRIM não foram tão satisfatórios quanto as outras duas versões. Ao analisar os experimentos foi verificado que o algoritmo muitas vezes apresentava o mesmo problema da heurística RDM: inserir alguns vértices na rota afim de aproveitar ao máximo a capacidade do veículo. Um nó distante da árvore pode vir a ser a única opção para inserção, pois os demais estão ainda mais distantes. Dessa forma, se a capacidade do veículo comportar esse nó, ele é inserido. Esse mesmo nó, se acrescentado em outra árvore, onde existem outros nós mais próximos, reduziria a distância total trafegada.

Para solucionar esse problema, a heurística Divisão com Prim e União de Árvores foi

implementada de forma que várias árvores são expandidas quase simultaneamente. Com isso, o problema citado anteriormente foi minimizado. A Figura 3.7 apresenta um exemplo de como as soluções do algoritmo Divisão com Prim e do algoritmo Divisão com Prim e União de Árvores podem variar.

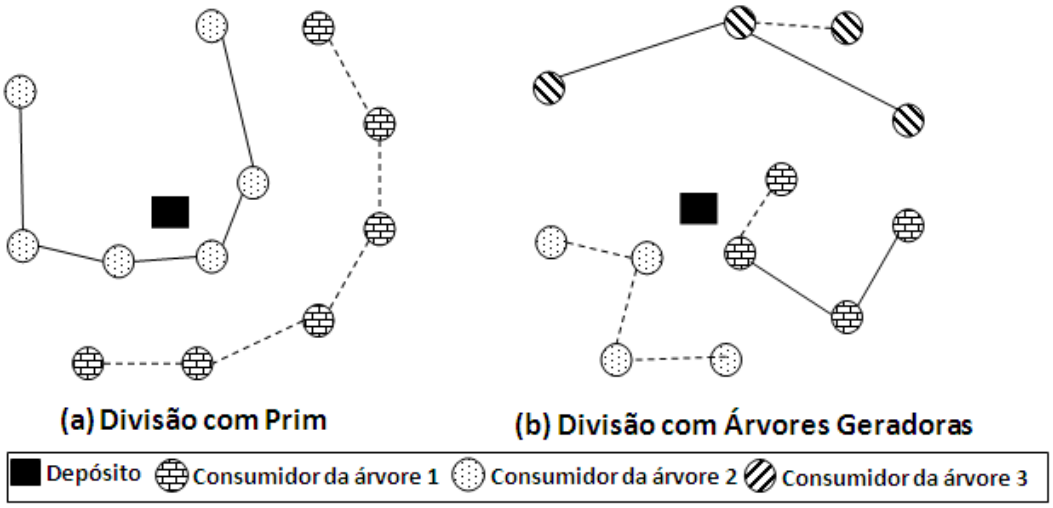


Figura 3.7: Exemplo de uma solução dada pelos algoritmos Divisão com Prim (a) e Divisão com Prim e União de Árvores (b)

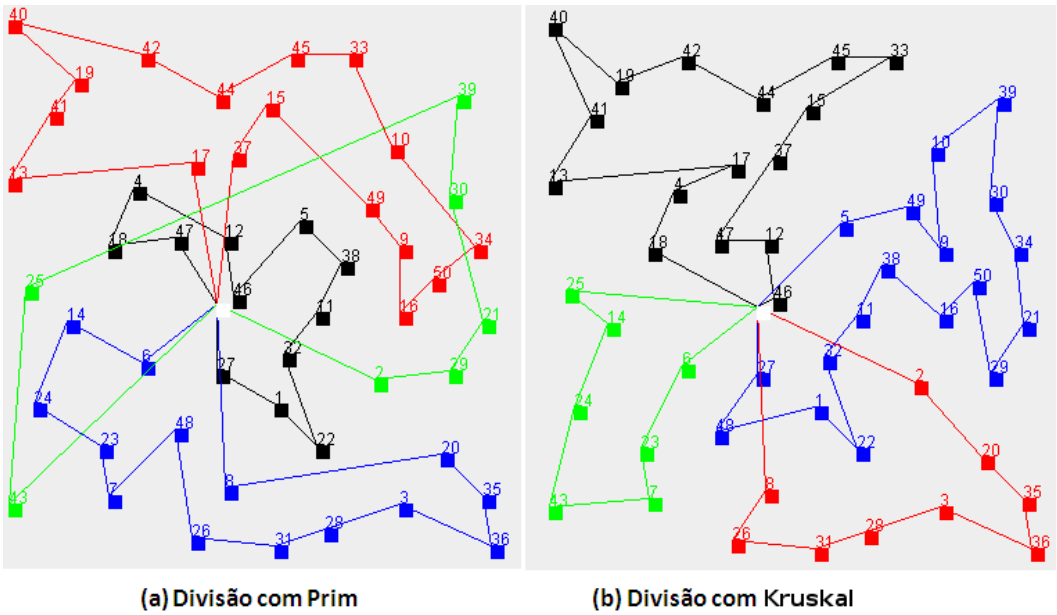


Figura 3.8: Exemplo de uma solução dada pelos algoritmos Divisão com Prim (a) e Divisão com Kruskal (b)

Os resultados foram ainda melhores ao aplicar o algoritmo de Kruskal na heurística DIV_KRUSKAL. Nela, a regra para geração das árvores é pela ordenação dos custos das arestas. A Tabela 3.5 mostra como a distância trafegada foi reduzida entre os DIV_PRIM,

Problema	N	Heuristica Baseada em Inserção (HIB)	
		V	D
CMT01_A	51	3	510
CMT01_B	51	3	523
CMT02_A	76	7	809
CMT02_B	76	7	792
CMT03_A	101	5	861
CMT03_B	101	5	822
CMT04_A	151	7	1095
CMT04_B	151	7	1073
CMT05_A	200	11	1325
CMT05_B	200	11	1327
CMT11_A	121	4	1019
CMT11_B	121	4	1057
CMT12_A	101	6	755
CMT12_B	101	5	794
MÉDIA	—	6,07	911

Tabela 3.6: Resultados da Heurística Baseada em Inserção

DIV_PUA e DIV_KRUSKAL. A primeira só é melhor em uma instância, a segunda é melhor em cinco instâncias e a última é melhor em 22 instâncias.

A Figura 3.8 apresenta as soluções dadas pelos algoritmos Divisão com Prim e Divisão com Kruskal para a instância CMT01_A. É visível como a divisão dos consumidores é mais eficiente na Figura 3.8(b).

A última heurística implementada (Heurística Baseada em Inserção - IBH) foi testada em 14 instâncias de Salhi e Nagy (1999). As instâncias não utilizadas contém restrições quanto a distância máxima percorrida pelos veículos e os resultados apresentados por Dethloff (2001) respeitam essa restrição. Como o presente trabalho não considera tais restrições, para não haver discordância entre os resultados apresentados pelo autor, elas não foram utilizadas.

A Heurística Baseada em Inserção foi executada 121 vezes para cada instância alternando os pesos da Capacidade Residual e Raio entre 0.0 a 1.0. Os melhores resultados entre as 121 execuções são apresentados na Tabela 3.6. Entre todos os métodos implementados a Heurística Baseada em Inserção apresentou melhores resultados em 64% das instâncias.

Comparando os resultados das duas melhores heurísticas, Divisão com Kruskal e a Baseada em Inserção, pode-se concluir que a primeira, por ser uma heurística construtiva e não necessitar de ajustes de pesos, apresentou os melhores resultados.

O próximo capítulo (Capítulo 4) apresentará algumas modificações feitas na Heurística Baseada em Inserção afim de reduzir alguns casos inconveniente, como em algumas situações onde a inserção de um novo consumidor pode acarretar num aumento exorbitante da distância

trafegada. Outro problema identificado é o cálculo do custo de inserção de um consumidor na rota. Dethloff (2001) faz os cálculos através da soma ponderada dos valores dados por cada critérios. Por serem critérios muito distintos seria adequado a utilização de um método de apoio a decisão multi-critérios.

Capítulo 4

Análise de Decisão Multicritérios

Em muitos problemas de otimização é crucial definir as possíveis soluções para o problema e compará-las. Muitas vezes, é necessário ainda avaliar cada alternativa sobre diferente aspectos ou critérios. Contudo, existe inúmeras ferramentas, como o AHP (*Analytic Hierarchy Process*), PROMETHEE (*Preference Ranking Organization Methods for Enrichment Evaluations*), ELECTRE (*Elimination and Choice Translating Reality*) para tratar esses tipos de problemas que necessitam de uma abordagem multicritérios.

A heurística baseada em inserção apresentada na Seção 3.6 utiliza vários critérios para decidir qual consumidor deve ser inserido na rota. Cada um desses critérios possui um peso de acordo com a sua prioridade. Essa etapa de priorização da heurística pode ser caracterizado como um problema de apoio a decisão multicritérios (Vincke, 1992). Este capítulo apresenta uma heurística de decisão multicritério para apoio à etapa de priorização com base no método PROMETHEE.

A Seção 4.1 apresenta a ferramenta PROMETHEE. A Seção 4.2 apresenta a forma como a heurística proposta por Dethloff (2001) foi trabalhada com uma abordagem multicritério. A Seção 4.3 apresenta os resultados obtidos e as conclusões do capítulo.

4.1 Método Promethee

O PROMETHEE (do inglês, *Preference Ranking Organization METHods for Enrichment Evaluations*) é um método de subordinação que tem por finalidade auxiliar no processo de tomada de decisão. Através dele, pode-se obter classificações de um conjunto finito de alternativas, que representam as possíveis ações, avaliadas segundo vários critérios.

Dado um conjunto A de n alternativas candidatas a serem classificadas e que representa as possíveis ações ($A = a_1, a_2, a_3, ..., a_n$) e um conjunto C de k critérios ($C = c_1, c_2, c_3, ..., c_k$)

que podem ser de maximização ou minimização e com unidades distintas de medida. A função $c_j(a_i)$ é a função de avaliação da ação i segundo critério j. Os valores das funções de avaliação podem ser representados pela matriz $M(n \times k)$:

$$M = \begin{bmatrix} c_1(a_1) & c_2(a_1) & c_3(a_1) & \cdots & c_k(a_1) \\ c_1(a_2) & c_2(a_2) & c_3(a_2) & \cdots & c_k(a_2) \\ c_1(a_3) & c_2(a_3) & c_3(a_3) & \cdots & c_k(a_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_1(a_n) & c_2(a_n) & c_3(a_n) & \cdots & c_k(a_n) \end{bmatrix}$$

A partir da matriz M é possível avaliar a preferência de cada par de soluções com base em cada um dos critérios. Para essa fase, é necessário definir uma função F_j de preferência para cada critério. Segundo Brans e Vincke (1985), F_j varia de zero a um e representa o papel do decisor dada as diferenças entre os pares de alternativas, para cada critério. O PROMETHEE apresenta seis formas diferentes de se representar preferências entre as funções de avaliação $c_j(a_i)$ (Figura 4.1). Os critérios podem ter funções de preferência distintas (Brans et al., 1986).

Definida uma função de preferência F_j para cada critério, é necessário calcular o valor de preferência de cada par de função de avaliação ($c_j(a_x)$, $c_j(a_y)$) aplicando a função de preferência sobre a diferença d entre ambas.

Como os critérios podem ser de maximização ou minimização, então:

$$d(a_x, a_y) = \begin{cases} c_j(a_x) - c_j(a_y) & \text{, se critério de maximização} \\ c_j(a_y) - c_j(a_x) & \text{, caso contrário} \end{cases}$$

$$\text{Valor de preferência} = F_j(d)$$

Na função de preferência Usual, dada duas funções de avaliação $c_j(a_x)$ e $c_j(a_y)$, se $d \leq 0$, então $F_j(d) = 0$, caso contrário $F_j(d) = 1$. Isso significa que, se $d > 0$ então o avaliador tem preferência estrita pela alternativa que possui melhor valor de avaliação, ou seja, a alternativa a_x.

$$F_j(d) = \begin{cases} 0 & \text{, se } d \leq 0 \\ 1 & \text{, caso contrário} \end{cases}$$

Na função de preferência Quase-Critério existe uma região de indiferença englobando todas as diferenças entre $c_j(a_x)$ e $c_j(a_y)$ inferiores ao limite de indiferença q. Para diferenças maiores, a preferência é total.

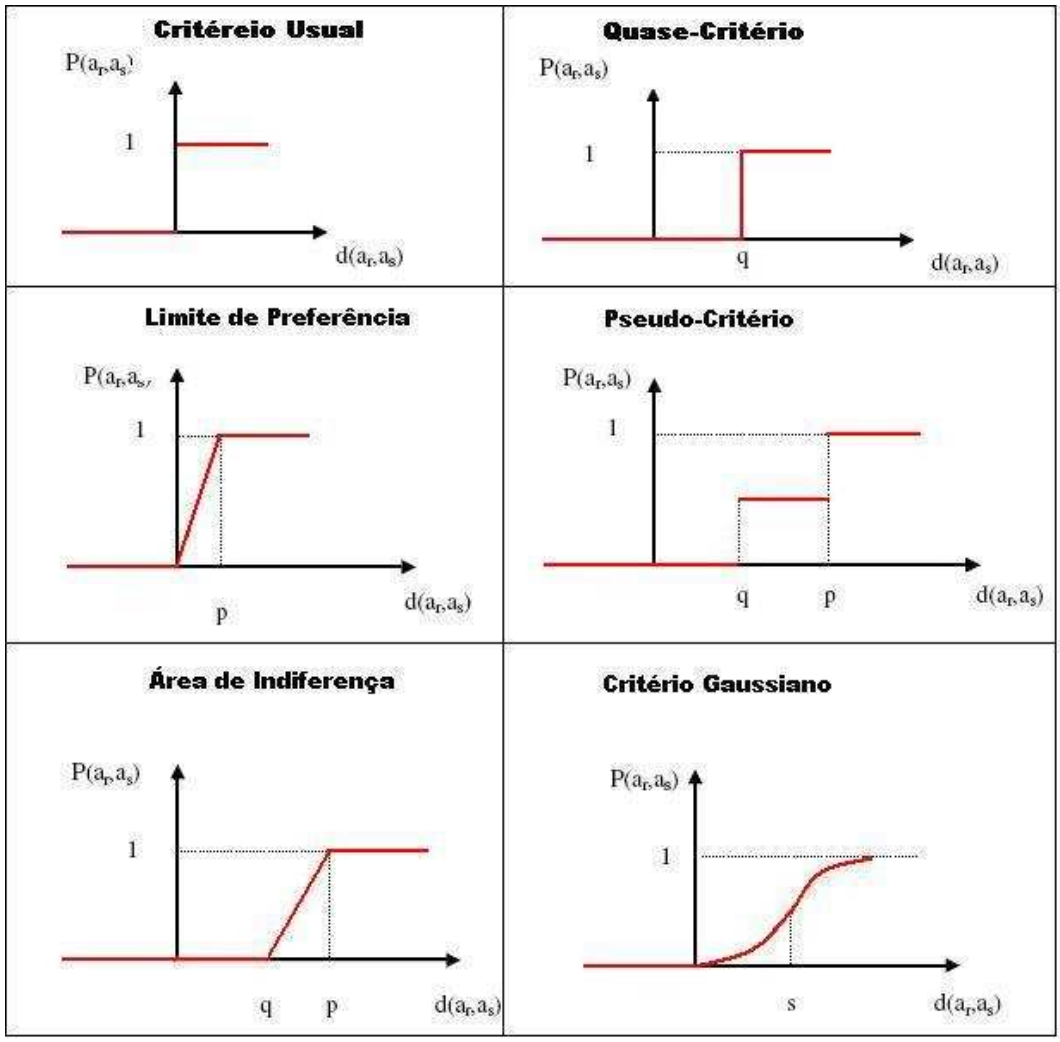


Figura 4.1: Funções de Preferência

$$F_j(d) = \begin{cases} 0 & \text{, se } d \leq q \\ 1 & \text{, caso contrário} \end{cases}$$

A função Limite de Preferência aumenta a preferência linearmente até que d alcance o limite de preferência p, para $d > p$ a preferência é total. Logo:

$$F_j(d) = \begin{cases} 0 & \text{, se } d \leq 0 \\ 1 & \text{, se } d \geq p \\ d/p & \text{, caso contrário} \end{cases}$$

Na Função Pseudo-Critério, $c_j(a_x)$ não tem preferência sobre $c_j(a_y)$ quando a diferença

não excede o limite de indiferença q. Para valores entre o limite de indiferença q e o limite de preferência p, considera-se um valor de preferência igual a $1/2$. Para valores superiores a p a preferência é total.

$$F_j(d) = \begin{cases} 0 & \text{, se } d \leq q \\ 1 & \text{, se } d \geq p \\ 0.5 & \text{, caso contrário} \end{cases}$$

Na função Área de Indiferença a preferência aumenta linearmente entre os limites de indiferença q e de preferência p. Caso contrário, a função de preferência é $d/p - q$.

$$F_j(d) = \begin{cases} 0 & \text{, se } d \leq q \\ 1 & \text{, se } d \geq p \\ d/(p-q) & \text{, caso contrário} \end{cases}$$

A função Gaussiana, a preferência aumenta segundo uma distribuição normal. Definida um valor s representando a mudança de concavidade da curva, a intensidade das preferência aumenta continuamente ao longo da curva.

$$F_j(d) = \begin{cases} 0 & \text{, se } d \leq 0 \\ 1 - e^{-d^2/2s^2} & \text{, caso contrário} \end{cases}$$

Calculada o valor de preferência de cada par de funções de avaliação, obtém-se uma matriz $G(k \times n \times n)$ no qual o item da matriz $g_{j,x,y}$ representa o valor de preferência entre as alternativas a_x e a_y segundo critério j ($F_j(d(a_x, a_y))$).

$$G[j] = \begin{bmatrix} --- & F_j(d(a_1,a_2)) & F_j(d(a_1,a_3)) & \cdots & F_j(d(a_1,a_n)) \\ F_j(d(a_2,a_1)) & --- & F_j(d(a_2,a_3)) & \cdots & F_j(d(a_2,a_n)) \\ F_j(d(a_3,a_1)) & F_j(d(a_3,a_2)) & --- & \cdots & F_j(d(a_3,a_n)) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ F_j(d(a_n,a_1)) & F_j(d(a_n,a_2)) & F_j(d(a_n,a_3)) & \cdots & --- \end{bmatrix}$$

O próximo passo é calcular os valores de preferência globais a partir dos valores de preferências obtidos com a função $F_j(d)$ levando em consideração o conjunto W de pesos $(w = w_1, w_2, ..., w_k)$ atribuídos a cada critério. Os pesos indicam a importância de cada um deles e, portanto, a escolha de valores deve ser feita pelo decisor. Se os critérios tiverem a mesmo nível de importância, então os pesos serão todos iguais (Lopes, 2005).

Os valores de preferência globais indicam o quanto uma alternativa x é melhor que uma

alternativa y sobre todos os aspectos (critérios). Os valores de preferência global de cada par de alternativas ($\pi(a_x, a_y)$) é dado por:

$$\pi(a_x, a_y) = \sum_{j=1}^{k} w_j * g(j, x, y)$$

Os valores de preferência global formam uma matriz $\Pi(n \times n)$ no qual o item $\pi(a_x, a_y)$ representa o valor de preferência global entre as alternativas a_x e a_y.

$$\Pi = \begin{bmatrix} 0 & \pi(a_1, a_2) & \pi(a_1, a_3) & \cdots & \pi(a_1, a_n) \\ \pi(a_2, a_1) & 0 & \pi(a_2, a_3) & \cdots & \pi(a_2, a_n) \\ \pi(a_3, a_1) & \pi(a_3, a_2) & 0 & \cdots & \pi(a_3, a_n) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \pi(a_n, a_1) & \pi(a_n, a_2) & \pi(a_n, a_3) & \cdots & 0 \end{bmatrix}$$

A matriz Π possui as seguintes propriedades:

$$\pi(a_x, a_y) = \begin{cases} 0 & \text{, se } x = y \\ 0 \leq \pi(a_x, a_y) \leq 1 & \text{, caso contrário} \end{cases}$$

Portanto, quando mais próximo de 0 estiver o valor de $\pi(a_x \times a_y)$ menor a preferência de a_x em relação a a_y. Caso contrário, quanto mais próximo de 1 estiver o valor de $\pi(a_x, a_y)$, maior a preferência de a_x sobre a_y.

Dado os valores de preferências globais entre todos os pares de alternativas, podemos calcular o fluxo de preferência ou importância de cada alternativa. Isso possibilita analisar uma alternativa em relação as demais. O fluxo de preferência positivo (Φ^+) de uma alternativa x, indica a preferência x sobre as demais alternativas. Quanto maior for Φ^+, melhor ela é. O fluxo de preferência negativo (Φ^-), indica o quanto ela foi inferior as demais. Quanto menor for Φ^-, melhor a alternativa.

$$\Phi^+(a_x) = \frac{1}{n-1} \sum_{y=1}^{n} \pi(a_x, a_y)$$

$$\Phi^{-}(a_x) = \frac{1}{n-1} \sum_{x=1}^{n} \pi(a_x, a_y)$$

Segundo Cavalcante e Almeida (2005), existem inúmeras versões do PROMETHEE. Este trabalho irá aplicar o PROMETHEE II para resolver problemas de apoio à decisão multicritérios.

O PROMETHEE II classifica as alternativas estabelecendo uma ordem decrescente do fluxo líquido entre o fluxo de preferência positivo e negativo (Lopes, 2005).

$$\Phi(a_x) = \Phi^{+}(a_x) - \Phi^{-}(a_x)$$

Se $\Phi(a_x) > \Phi(a_y)$ então a alternativa x é melhor que a alternativa y em relação ao conjunto de critério. Logo, podemos considerar a alternativa com o maior valor de Φ a melhor entre todas as alternativas.

4.2 Heurística Baseada em Inserção com Modelo Multicritério de Apoio a Decisão

Este capítulo apresenta algumas modificações aplicadas na Heurística Baseada em Inserção apresentada na Seção 3.6. A primeira alteração é relacionada com a inserção de novos consumidores na rota.

Em algumas situações, a inserção de um novo consumidor pode acarretar num aumento exorbitante da distância trafegada. Nesses casos o ideal seria finalizar a rota. Para solucionar esse problema são consideradas alternativas para fechar a rota e abrir uma nova.

A Heurística Baseada em Inserção com Modelo Multicritério de Apoio a Decisão (BIMAD), proposta nesse trabalho, escolhe aleatoriamente um consumidor (semente) e define uma nova rota (*depósito* → *semente* → *depósito*), assim como a Heurística Baseada em Inserção. Mas, além de considerar a inserção de cada consumidor ainda não roteado em cada posição viável da rota, ele insere alternativas para o fechamento da rota.

Após gerar uma rota inicial com uma semente e o depósito, a heurística proposta gera um conjunto de alternativas $A = \{a_1, a_2, a_3, ..., a_n\}$. Cada uma representa a inserção de um consumidor em uma posição viável da rota. Além disso, são criadas alternativas para fechamento da rota, geradas da seguinte forma: para cada consumidor não roteado i é criada uma alternativa considerando a criação de uma nova rota *depósito* → i → *depósito*.

A cada passo, a heurística considera o custo de cada critério para cada alternativa. O problema de selecionar a melhor das alternativas pode ser visto como um problema de decisão multicritério (MCDA, do inglês *MultiCriteria Decision Aid*). Ferramentas MCDA, segundo Vincke (1992), são métodos designados a resolver estes tipos de problema. O PROMETHEE é uma ferramenta MCDA utilizada para fazer decisões com base em um conjunto de critérios (Brans e Vincke, 1985).

A segunda alteração feita na heurística baseada em inserção é no cálculo do custo de inserção de um consumidor na rota. Dethloff (2001) faz os cálculos através da soma ponderada dos valores dados por cada critério. Por serem critérios muito distintos seria adequado a utilização de um método de apoio a decisão multicritérios como o PROMETHEE. Isso torna a Heurística Baseada em Inserção com Modelo Multicritério de Apoio a Decisão menos gulosa e mais global.

Os critérios utilizados são os mesmos da heurística baseada em inserção, ou seja, distância trafegada (DT), capacidade residual (CR) e raio de abrangência (RA) ($C = \{DT, CR, RA\}$).

A partir dos conjuntos A e C é possível preencher a matriz M com os valores de $c_j(a_i)$ mostrados na Seção 4.1.

Os pesos $W = \{w_{DT}, w_{CR}, w_{RA}\}$ foram inicializados dando maior preferência para os critérios CR e RA e a medida que os nós são inseridos na rota, a heurística passa a ficar cada vez mais gulosa dando cada vez mais prioridade ao critério DT.

Quanto a função de preferência, a função Limite de Preferência F_j foi atribuída para todos os critérios. Como foi mostrado na Seção 4.1, esta função possui um limite de preferência p. Esse limite foi definido como a diferença entre o custo máximo e mínimo de cada critério. Ou seja, o valor de p da função de preferência do critério j será:

$$p_j = Max\{c_j(a_i)\} - Min\{c_j(a_i)\}, \quad \forall \quad i = 1...n$$

Após definidos todos esses parâmetros a ferramenta PROMETHEE irá, então avaliar a preferência de todo par de soluções com base em cada um dos critérios. Um valor de preferência entre cada par é calculado dada a função de preferência F_j e o peso w_j de cada critério. Então, é definido um "fluxo global"de preferência de cada alternativa e a classificação dessas.

O PROMETHEE então retorna a alternativa melhor classificada. A alternativa indica qual consumidor deve ser inserido na rota e a sua posição ou o fechamento da rota atual e a inicialização de outra. O Algoritmo 4.1 apresenta a Heurística Baseada em Inserção com Modelo Multicritério de Apoio a Decisão. O Algoritmo 4.2 apresenta os passos da ferramenta PROMETHEE.

Algoritmo 4.1: Heurística Baseada em Inserção com Modelo Multicritério de Apoio a Decisão

```
1  BIMAD(G {Grafo completo com os consumidores e depósito},
2        Q {Capacidade dos veículos},
3        D {Demandas dos consumidores})
4
5     Enquanto existir consumidor fora de alguma rota
6        semente = Selecionar um consumidor aleatoriamente
7        r = Criar uma nova rota (depósito - semente - depósito)
8        Para cada consumidor c não inserido em alguma rota
9
10                {Criar alternativa para finalizar a rota e iniciar uma
11                nova com o consumidor c (depósito - c - depósito)}
12
13                novaRota = Criar rota (depósito - c - depósito)
14                distância = Calcular distância da novaRota
15                cr = Calcular capacidade residual da novaRota
16                raio = Calcular raio de abrangência da novaRota
17                alternativa = CriarAlternativa(fecharRota, consumidor c,
18                                                distância,cr,raio)
19                Inserir alternativa no conjunto A de alternativas
20
21                {Criar alternativa para inserir o consumidor em cada
22                posição viável da rota}
23
24                Para cada posição i da rota
25                        Se consumidor c puder ser inserido na posição i
26                            distância = Calcular distância da rota após
27                                        inserir consumidor c na posição i
28                            cr = Calcular capacidade residual da rota
29                                 após inserir consumidor c na posição i
30                            raio = Calcular raio de agrangência após inserir
31                                   consumidor c na posição i
32                            alternativa = CriarAlternativa(posição i,
                                  consumidor c,
33                                        distância,cr,raio)
34                            Inserir alternativa no conjunto A de alternativas
35
36          {Definindo o limite "p" da função de preferência para cada um dos
               critérios}
37          pDistância = maior distância -  menor distância
38          pCr = maior cr -  menor cr
39          pRaio = maior raio -  menor raio
40
41          Definir Funções de preferência para cada critério com
42             os parâmetros(pDistancia, pCr, pRaio)
43
44          alternativaMelhor = Promethee(conjunto A de alternativas,
45                                        Funções de Preferência,
46                                        Pesos);
47          Reduzir o peso dos critérios cr e raio e aumentar o peso do critério
```

```
            distância
48
49          Se alternativaMelhor é para fechar rota
50              Fechar rota r e inserí-la no conjunto R
51              semente = Selecionar um consumidor aleatoriamente
52              r = Criar uma nova rota (depósito - semente - depósito)
53          Senão
54              Insere consumidor na posição especificada da alternativa
55
56      Retornar R
```

Algoritmo 4.2: Algoritmo PROMETHEE

```
1   Promethee(A {Conjunto de Alternativas},
2             F {Conjunto de Funções de Preferências},
3             W {Conjunto de Pesos})
4
5       M = criar matriz com a avaliação de cada alternativa por cada critério
6       G = criar matriz com o valor de preferência de cada par
7           utilizando as funções de preferências definidas para cada critério
8       PI = criar matriz com os valores de preferência global utilizando o pesos
9            de cada critério
10      FP+ = Calcular fluxo de preferência positivo
11      FP- = Calcular fluxo de preferência negativo
12      FP = Calcular fluxo de preferência líquido
13
14      retornar maior FP
```

4.3 Resultados Computacionais

O algoritmo proposto foi testado em 14 instâncias utilizadas por Salhi e Nagy (1999). Os resultados são apresentados na Tabela 4.1. As duas primeiras colunas mostram o nome e o tamanho de cada instância. As duas colunas seguintes mostram o número de veículos (V) e a distância trafegada (D) da melhor solução obtida por Dethloff (2001) (IBH). As três últimas colunas mostram os mesmos dados para a heurística proposta (BIMAD) e a porcentagem de melhoria da distância total trafegada (δ^*).

Para comparar os resultados com a Heurística Baseada em Inserção apresentada na Seção 3.6, a Heurística Baseada em Inserção com Modelo Multicritério de Apoio a Decisão foi executada 121 vezes para cada instância alternando os pesos da Capacidade Residual (w_{CR}) entre 0.05 a 0.15. Os pesos do Raio de Abrangência (w_{RA}) variaram entre 0.35 a 0.45. O peso do critério guloso (distância trafegada), w_{DT}, é definido como:

$$w_{DT} = 1 - (w_{CR} + w_{RA})$$

Os resultados mostraram a superioridade da heurística BIMAD em relação a heurística gulosa (IBH). As soluções apresentadas pela heurística proposta foram melhores em 78,57% das instâncias. A média total de melhoria da qualidade da solução é de 3,04%. Os resultados mostraram um tempo de execução satisfatório (poucos segundos) nos testes feitos com 50 a 200 consumidores.

Comparando os resultados da heurística BIMAD com os resultados da heurística Divisão com Kruskal (Tabela 3.5), a primeira reduz a média da distância total trafegada 7,06% (*(Média DIV_KRUSKAL - Média BIMAD)*100/Média DIV_KRUSKAL*) e apresenta os melhores resultados em 92% das instâncias analisadas.

		IBH		BIMAD		
Instancia	N	V	D	V	D	δ^*
CMT01_A	51	3	510	3	502	1.52
CMT01_B	51	3	523	3	512	2.11
CMT02_A	76	7	809	7	776	4.14
CMT02_B	76	7	792	6	750	5.34
CMT03_A	101	5	861	5	838	2.68
CMT03_B	101	5	822	5	810	1.54
CMT04_A	151	7	1095	8	1041	4.89
CMT04_B	151	7	1073	8	1091	-1.76
CMT05_A	200	11	1325	11	1289	2.13
CMT05_B	200	11	1327	11	1300	2.11
CMT11_A	121	4	1019	5	988	2.61
CMT11_B	121	4	1057	5	981	6.89
CMT12_A	101	6	755	6	734	1.71
CMT12_B	101	5	794	6	743	6.53
MEDIA	114	6,07	912	6,36	882	3,04

Tabela 4.1: Resultados numéricos da heuristica MCDA

Capítulo 5

Heurística GRASP

5.1 Introdução

O GRASP (do inglês, *Greedy Randomized Adaptive Search Procedure*) foi proposto por Feo e Resende (1995). Ele é composto por duas fases: uma fase de construção e uma fase de busca local.

Na primeira fase do GRASP uma solução inicial é gerada. Esta solução será alterada na fase de busca local, visando melhorar os valores da função objetivo. As buscas locais fazem uma análise da vizinhança da solução inicial e os melhores movimentos são executados. Esse processo é interativo e a melhor solução encontrada é retornada.

As duas fases do GRASP são apresentadas nas seções seguintes. A Seção 5.2 apresenta as heurísticas utilizadas nesse trabalho para encontrar uma solução inicial do GRASP. Todas elas são adaptações das heurísticas apresentadas no Capítulo 3.

A Seção 5.3 apresenta as buscas locais implementadas que irão percorrer a vizinhança da solução inicial. A Seção 5.4 apresenta uma visão geral do GRASP, incluindo detalhes de implementação, e os resultados obtidos são mostrados na Seção 5.5.

5.2 Fase de Construção

Para encontrar uma solução inicial na fase de construção foram utilizadas três heurísticas descritas no Capítulo 3 e a heurística proposta no Capítulo 4. São elas:

1. Rotear e Dividir
2. Divisão com Prim e União de Árvores

3. Divisão com Kruskal

4. Heurística Baseada em Inserção com Modelo de Apoio a Decisão Multicritério (BIMAD)

A primeira foi selecionada por retornar bons resultados e por ser extremamente rápida. As demais pela qualidade da solução encontrada. Todas as heurísticas utilizadas tiveram que sofrer pequenas alterações afim de torná-las não determinísticas. A seguir iremos descrever como cada heurística foi alterada.

5.2.1 Heurística Rotear e Dividir Não Determinística

A heurística Rotear e Dividir sofreu alterações na primeira fase do algoritmo (fase de roteamento). O algoritmo inicia definindo um trajeto com todos os nós (consumidores e depósito) utilizando uma solução do TSP. Neste caso foi aplicada a heurística Vizinho Mais Próximo descrita na Subseção 3.1.1.

O algoritmo, durante a criação desse trajeto inicial, utiliza a heurística Vizinho Mais Próximo. Este algoritmo irá selecionar alguns vértices mais próximos do último vértice da rota e aleatoriamente escolhe um deles para ser o próximo vértice da rota.

Posteriormente, seguindo o trajeto obtido pela solução do TSP, os nós vão sendo inseridos em uma rota. Caso a inserção de um nó extrapole a capacidade do veículo, uma nova rota é criada. O processo é finalizado quando o último nó da solução do TSP é alcançado.

5.2.2 Heurística Divisão com Prim e União de Árvores Não Determinística

Na heurística Divisão com Prim e União de Árvores (DIV_PUA) são criadas, inicialmente, n árvores, uma para cada consumidor. Ou seja, cada árvore será composta por um único nó.

A cada passo do algoritmo, uma árvore é selecionada aleatoriamente para ser expandida seguindo os princípios de Prim. Os passos seguintes são conforme descrito na Subseção 3.5.2.

5.2.3 Divisão com Kruskal Não Determinística

Na heurística Divisão com Kruskal (DIV_KRUSKAL) são criadas, inicialmente, n árvores geradoras, sendo n o número de consumidores. Posteriormente, as distâncias entre os consumidores são ordenadas de forma crescente (Subseção 3.5.3).

A aleatoriedade entra na fase de expansão das árvores. Ao invés de seguir a ordenação das distâncias entre os consumidores, as i primeiras distâncias dessa ordem são selecionadas e uma delas é aleatoriamente escolhida para que os nós adjacentes sejam unidos. Antes da união, deve-se verificar sempre a capacidade do veículo segundo sua MAXLOAD.

Após formados os grupos, a segunda fase do algoritmo efetua o roteamento utilizando um algoritmo para o TSP.

5.2.4 Heurística Baseada em Inserção com Modelo Multicritério de Apoio a Decisão

Conforme descrito na Seção 4.2, a heurística baseada em inserção com modelo multicritério de apoio a decisão aplica a ferramenta PROMETHEE para encontrar a melhor alternativa dentre várias.

Diferenciando da heurística apresentada na Seção 4.2, o PROMETHEE retorna algumas alternativas que foram melhores classificadas, e não mais a melhor alternativa, e aleatoriamente uma dessas alternativas é selecionada.

5.3 Buscas Locais

Foram definidas várias buscas locais para serem aplicadas a solução inicial encontrada na fase de construção. Quatro buscas locais são inter-rotas e uma intra-rota. As Subseções 5.3.1 a 5.3.5 descrevem cada uma das buscas locais utilizadas neste trabalho.

5.3.1 Eliminação de Rotas

A eliminação de rotas visa excluir algumas rotas da solução encontrada na fase de construção, incluindo seus vértices nas demais rotas buscando minimizar o custo. Todas as rotas são candidatas a serem eliminadas.

A busca local seleciona aleatoriamente uma rota da solução inicial. Então, a inserção de seus vértices é analisada em cada posição das demais rotas visando minimizar a distância trafegada. Após analisar a exclusão de todas as rotas, aquela que acarretar em uma menor distância trafegada será efetivada. Este processo é repetido afim fazer outras eliminações.

A Figura 5.1 mostra um exemplo da aplicação da busca local eliminação de rotas. No caso apresentado, a rota dois é eliminada tendo seus vértices inseridos na rota 1 e 3.

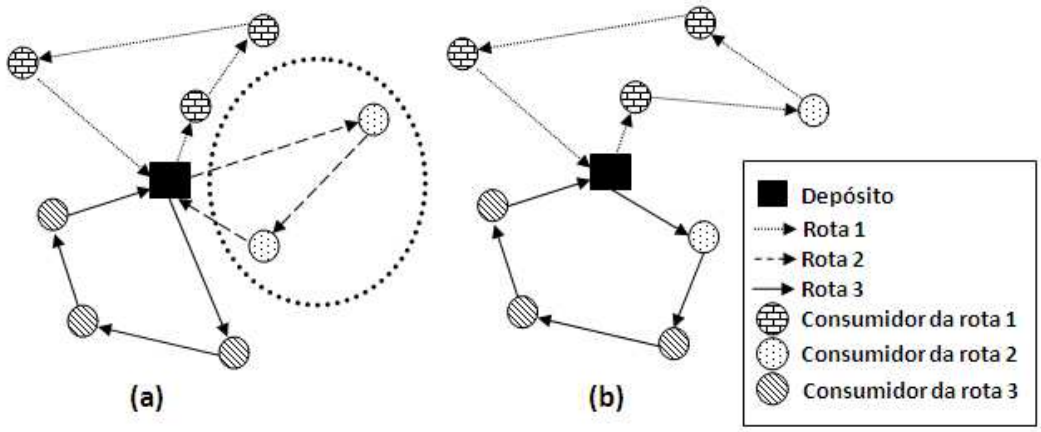


Figura 5.1: Exemplo da busca local eliminação de rotas

5.3.2 Realocação

A busca local realocação remove aleatoriamente os vértices da rota na tentativa de encontrar uma posição viável de menor custo em outra rota. Todos os vértices são candidatos a mudar de rota.

A Figura 5.2 mostra um exemplo desta busca local. Na figura um dos nós da rota 2 é retirado e inserido na rota 3. Todas as trocas serão analisadas e somente aquela que mais reduz os custos de transporte será mantida.

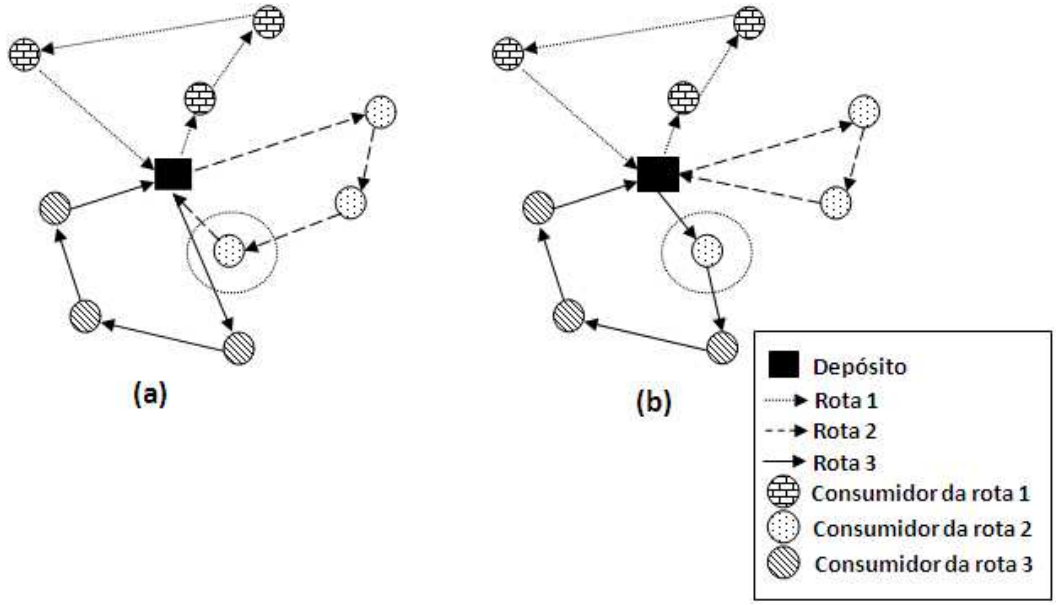


Figura 5.2: Exemplo da busca local realocação

5.3.3 Intercâmbio

A busca local intercâmbio visa trocar de lugar dois vértices situados em rotas distintas. Estes vértices serão inseridos na melhor posição viável, ou seja, posição que reduz os custos de transporte sem extrapolar a capacidade dos veículos. A busca local faz a troca entre todos os possíveis pares de vértices situados em rotas distintas. Após analisar todas as trocas, aquela que mais reduzir os custos será efetuada.

A Figura 5.3 mostra um problema com três rotas definidas previamente. Neste exemplo, dois vértices pertencentes a rotas distintas (rota 2 e 3) são trocadas.

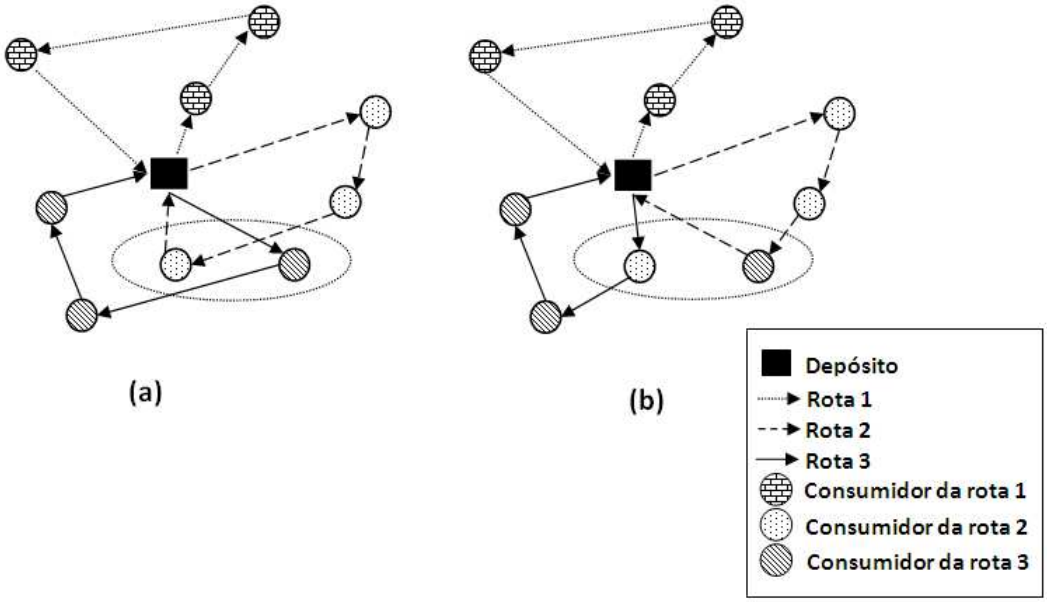


Figura 5.3: Exemplo da busca local intercâmbio

5.3.4 Cruzamento

A busca local cruzamento seleciona dois nós aleatoriamente (x_i e y_j) contidos em rotas distintas $R_x = x_1, x_2, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n$ e $R_y = y_1, y_2, ..., y_{j-1}, y_j, y_{j+1}, ..., y_m$. Posteriormente o cruzamento dessas rotas é feito, tendo como parâmetros os nós x_i e y_j. O resultados do cruzamento são duas rotas constituídas pelos nós $R_x = x_1, x_2, ..., x_{i-1}, x_i, y_{j+1}, ..., y_m$ e $R_y = y_1, y_2, ..., y_{j-1}, y_j, x_{i+1}, ..., x_n$

A Figura 5.4 mostra um exemplo onde o cruzamento é feito entre as rotas 2 e 3. Todas os cruzamentos viáveis são analisados e aqueles que mais reduzirem os custos são aplicados.

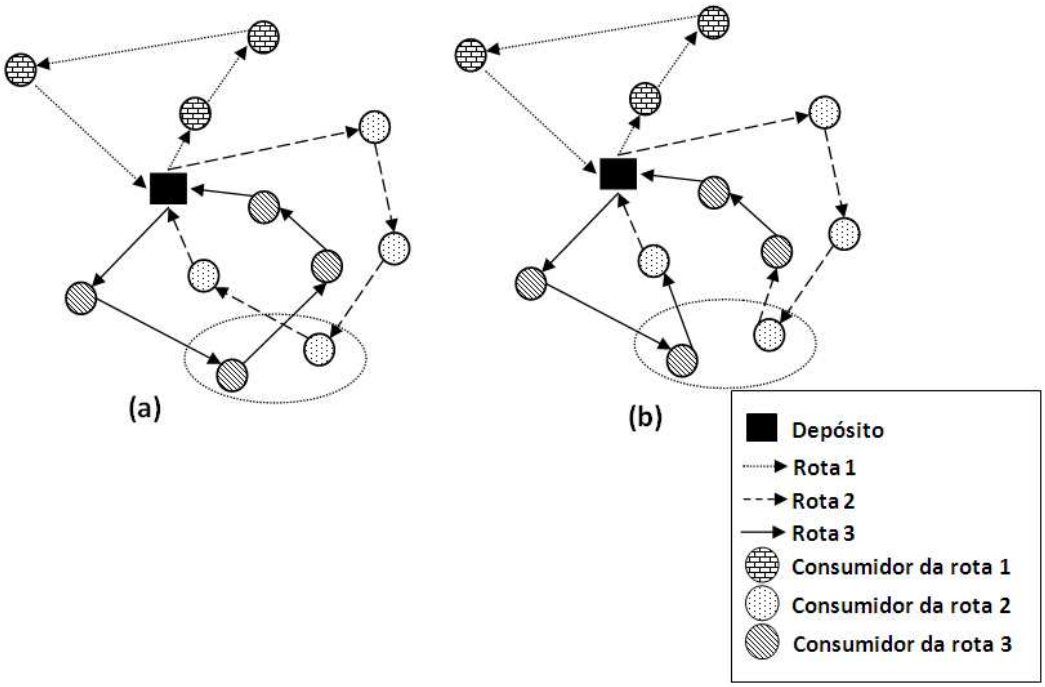


Figura 5.4: Exemplo da busca local cruzamento

5.3.5 2-OPT

A busca local 2-Opt é intra-rota, ou seja, as movimentações são feitas internamente a rota. Todas as demais buscas locais citadas anteriormente são inter-rotas, pois as trocas são feitas entre rotas distintas.

A busca local 2-Opt tem como objetivo reduzir os custo de transporte de determinada rota alterando a ordem como os nós são visitados. Cada movimento representa a remoção de duas arestas e a inclusão de outras duas. Os movimentos que apresentam uma redução na distância percorrida são efetivados. Todos os nós da rota são testados em todas as posições viáveis na mesma, e as melhores trocas são efetuadas. A Figura 5.5 mostra um exemplo desta busca.

5.4 Aplicação do Grasp

O GRASP inicialmente busca uma solução inicial utilizando uma das heurísticas descritas na Seção 5.2. Posteriormente as buscas locais são aplicadas a essa solução afim de reduzir os custos.

Neste trabalho optamos por utilizar inicialmente a busca local eliminação de rota. Todas as heurísticas apresentadas na Seção 5.2 efetuam a divisão dos nós de acordo com a maior

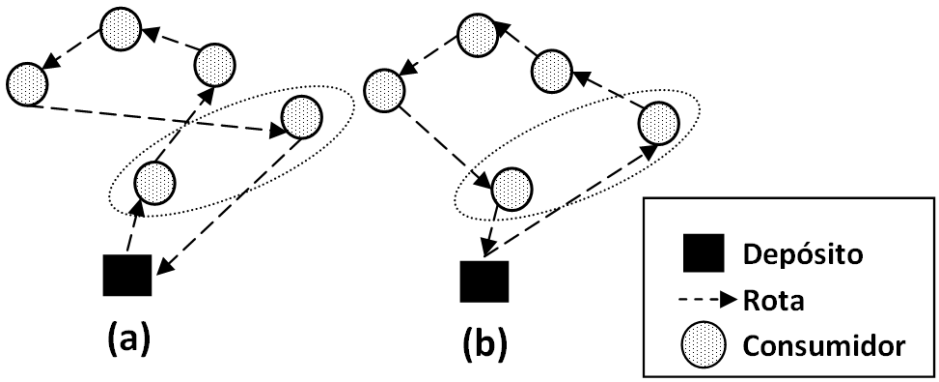


Figura 5.5: Exemplo da busca local 2-Opt.

carga possível (MAXLOAD) que sempre considera o pior caso, ou seja, visitar inicialmente todos os consumidores com demanda líquida positiva e em seguida os demais. Essa forma de divisão garante que qualquer trajeto definido após a divisão seja viável. Em contrapartida, o método finaliza rotas que comportam outros consumidores a mais. Portanto, a eliminação de rota irá amenizar a desvantagem de antecipar a finalização de rotas.

Após a eliminação de rotas, as demais buscas locais inter-rotas são aplicadas. Por fim, a busca local 2Opt é aplicada em cada rota.

As buscas locais recebem como parâmetro uma solução na qual se pretende reduzir o seu custo de transporte. O retorno das buscas locais é uma solução otimizada, caso consiga alguma melhoria com a busca local, o procedimento é repetido. A fase de busca local é executada enquanto apresentar melhoria na solução.

O Algoritmo 5.1 ilustra o procedimento GRASP para o problema de roteamento de veículos com coleta e entrega simultâneas utilizando na fase de construção uma das heurísticas apresentadas na seção 5.2 e as buscas locais apresentadas na seção 5.3.

Algoritmo 5.1: Algoritmo GRASP

```
1  GRASP(G {Grafo completo com os consumidores e depósito},
2        Q {Capacidade dos veículos},
3        D {Demandas dos consumidores},
4        NUM_ITERACOES {número de iterações do GRASP})
5
6      melhorSolução = nulo;
7
8      Para i=0 até NUM_ITERACOES
9
10         solução = faseConstrução(G,Q,D);
11
12         Enquanto houver melhorias na solução
13
14             Enquanto houver melhorias na solução
```

```
15                  solução = Eliminação de Rotas(solução);
16
17              Enquanto houver melhorias na solução
18                  solução = Realocação(solução);
19
20              Enquanto houver melhorias na solução
21                  solução = Intercâmbio(solução);
22
23              Enquanto houver melhorias na solução
24                  solução = Cruzamento(solução)
25
26              Enquanto houver melhorias na solução
27                  solução = 2Opt(solução)
28
29          se (melhorSolução == nulo ou
30              solução < melhorSolução)
31              melhorSolução = solução
32
33      retornar melhorSolução
```

5.5 Resultados Computacionais

O GRASP proposto neste trabalho foi testado em instâncias com 50 a 200 consumidores, propostas por Salhi e Nagy (1999). Algumas delas possuem restrição de distância máxima a ser percorrida pelos veículos. Esta restrição foi ignorada. Os algoritmos foram implementados em Java (JDK 1.4) e executados em um Pentium IV 2.40 GHz. O número de iterações do GRASP foi de 100.

A tabela 5.1 mostra os resultados do GRASP. A primeira coluna apresenta o nome da instância. As demais colunas apresentam os resultados obtidos: número de veículos (V), distância total percorrida (D) e tempo (T) [milisegundos], pelas heurísticas Grasp com Rotear e Dividir, com Prim e União de Árvores, com Kruskal e com MCDA.

Os resultados mostraram que o GRASP com Prim e União de Árvores e Kruskal são superiores ao GRASP com Rotear e Dividir em 96% das instâncias analisadas. Comparando o GRASP com Prim e União de Árvores e Grasp com Kruskal, observa-se que a segunda apresentou melhores resultados em 53%.

Os pesos da heurística MCDA aplicada na fase de construção do GRASP, foram ajustados da seguinte forma: o peso do critério Capacidade Residual foi fixado em 0.5 e o peso do critério Raio de Abrangência em 0.32. Estes pesos mostraram bons resultados independente da instância utilizada.

A heurística MCDA não apresentou bons resultados pela sua necessidade de ajustes de

	GRD			GPUA			GK			GMCDA		
Problema	V	D	T	V	D	T	V	D	T	V	D	T
CMT01_A	3	**464**	106	3	474	219	3	465	284	3	478	571
CMT01_B	3	493	111	3	484	214	3	**468**	283	3	472	555
CMT02_A	6	674	295	6	690	765	5	**648**	971	6	698	824
CMT02_B	6	719	333	6	685	790	6	**634**	993	6	688	791
CMT03_A	5	777	389	4	730	1039	5	**728**	1625	5	755	6332
CMT03_B	4	766	455	4	743	1023	4	**717**	1648	5	744	5791
CMT04_A	6	972	1613	6	**885**	21344	7	897	30105	6	918	24638
CMT04_B	6	960	1732	7	916	1868	6	**903**	7043	6	933	47448
CMT05_A	9	1145	4825	10	1083	7408	9	**1075**	21034	8	1133	48681
CMT05_B	8	1192	8845	9	**1056**	9656	9	1132	23460	8	1108	40413
CMT06_A	3	492	106	3	484	212	3	**460**	288	3	467	535
CMT06_B	3	472	114	3	474	225	3	**468**	292	3	477	655
CMT07_A	5	723	281	7	689	755	5	**634**	941	5	679	740
CMT07_B	5	708	296	6	683	780	6	**656**	971	5	683	808
CMT08_A	4	727	428	5	**714**	1047	5	737	1741	4	761	5458
CMT08_B	4	741	457	3	731	1069	4	**729**	1697	4	752	6700
CMT09_A	7	919	19470	6	906	7847	6	**879**	2969	7	959	1926
CMT09_B	7	928	20137	5	**894**	7963	7	900	3153	7	986	2535
CMT10_A	9	1165	35574	9	1100	23052	9	**1062**	9405	9	1185	5773
CMT10_B	8	1183	8154	9	**1072**	9626	9	1142	34829	10	1125	26654
CMT11_A	4	848	938	4	**797**	1218	4	815	1901	3	822	25108
CMT11_B	4	861	823	4	**794**	1318	5	804	1778	4	831	16672
CMT12_A	5	720	480	5	**666**	750	5	685	1834	6	697	3474
CMT12_B	6	695	2098	5	676	879	4	**671**	1865	6	685	3521
CMT13_A	4	854	966	3	**785**	947	3	789	1887	3	832	15010
CMT13_B	4	823	723	4	**790**	1047	4	807	1766	3	825	24142
CMT14_A	6	718	515	6	**683**	764	6	701	1972	6	694	3465
CMT14_B	6	746	533	5	**671**	765	5	678	1854	5	712	2914
MÉDIA	5,4	803	–	5,4	763	–	5,4	760	–	5,3	789	–

Tabela 5.1: Resultados do Grasp utilizando as heurísticas Rotear e Dividir (GRD), Divisão com Prim e União de Árvores (GPUA), Divisão com Kruskal (GK) e Heurística Baseada em Inserção com MCDA (GMCDA) na fase de construção.

pesos. Apesar dos pesos fixados apresentarem bons resultados no geral, a heurística é mais eficiente se estes pesos forem ajustados de acordo com a instância utilizada. A última linha da tabela 5.1 mostra, que na média, a qualidade das soluções obtidas é superior à da heurística GRASP com Rotear e Dividir. Em nenhuma instância analisada esta heurística apresentou os melhores resultados.

No geral, a heurística GRASP Divisão com Kruskal (GK) apresentou as melhores soluções, considerando a distância trafegada total. Portanto, a tabela 5.2 compara estes resultados com os resultados obtidos com a heurística Busca Tabu (BT), desenvolvida por Montané e Galvão (2006). A quarta coluna da tabela mostra a porcentagem de melhoria das soluções (*((Solução BT - Solução GK)/Solução BT)*100*) obtidas pela heurística GRASP Divisão com Kruskal em relação aos resultados da heurística Busca Tabu. A quinta coluna apresenta um limite inferior (Montané e Galvão, 2006) para o problema e a sexta coluna o %Gap (*((Solução*

Problema	GK	BT	% Melhoria GK	LB	%Gap
CMT01_A	**465**	472	1,51	454,68	2,27
CMT01_B	**468**	470	0,43	455,52	2,73
CMT02_A	**648**	695	7,25	617,01	5,02
CMT02_B	**634**	700	10,41	617,64	2,65
CMT03_A	728	**721**	-0,96	646,73	12,57
CMT03_B	**717**	719	0,28	648,04	10,64
CMT04_A	897	**880**	-1,90	714,18	25,60
CMT04_B	903	**878**	-2,77	715,67	26,17
CMT05_A	**1075**	1089	1,30	858,14	25,28
CMT05_B	1132	**1083**	-4,33	856,59	32,15
CMT11_A	**815**	900	10,43	663,38	22,85
CMT11_B	**804**	910	13,18	662,84	21,30
CMT12_A	685	**675**	-1,46	568,79	20,43
CMT12_B	**671**	689	2,68	573,53	16,99
MÉDIA	760	777	2,58	646,62	16,19

Tabela 5.2: Comparação entre heurística GRASP Divisão com Kruskal e Heurística Busca Tabu (Montané e Galvão, 2006)

*DIV_KRUSKAL - limite inferior)/Solução DIV_KRUSKAL)*100*) da heurística GRASP Divisão com Kruskal.

A heurística GRASP Divisão com Kruskal apresenta os melhores resultados em 64% das instâncias analisadas, melhorando, em média, 2,58% a qualidade das soluções obtidas pela Busca Tabu. O Gap médio foi de 16,19%.

Estes resultados mostraram que a heurística Divisão com Kruskal é uma boa alternativa para definir uma solução inicial para o problema de Roteamento de Veículos com Coleta e Entrega Simultâneas.

Capítulo 6

Conclusões

Devido à complexidade computacional do VRPSPD, o desenvolvimento e análise de heurísticas construtivas para encontrar soluções é extremamente importante. Assim, o presente trabalho apresentou um conjunto de heurísticas e ainda uma metaheurística GRASP para o problema de roteamento de veículos com coleta e entrega simultâneas.

A principal contribuição foi analisar e comparar algumas heurísticas da literatura e outras propostas nesse trabalho que até então são desconhecidas. O processo de implementação destas heurísticas se fez de forma gradativa. Iniciando por heurísticas mais simples e finalizando com heurísticas mais elaboradas, visando a redução do número de veículos e da distância trafegada pelos veículos.

Ao término dos experimentos, foi possível perceber que heurísticas simples como RD apresentaram bons resultados. Inicialmente, a heurística passa uma falsa impressão de retornar um grande número de rotas. Na prática isso não ocorre e o número de veículos utilizado não difere muito das demais heurísticas. No entanto, as heurísticas como Rotear e Dividir Máximo, Dividir e Rotear com Carga Mínima e Dividir e Rotear com Carga Máxima que priorizavam a redução do número de veículos não demonstraram muita efetividade. O número de veículos entre todas heurísticas não teve uma variação significativa. Por outro lado, a redução da distância trafegada foi significativa nas heurísticas Divisão com Kruskal e Heurística Baseada em Inserção. Isso comprovou que, mesmo visando unicamente a minimização da distância trafegada, a redução do número de veículos ocorre em conjunto.

Apesar dos bons resultados da heurística Baseada em Inserção, deve-se considerar que ela define outros critérios além do guloso para criar as rotas. Comparando com a heurística Divisão com Kruskal, o método ganha na redução dos custos, mas perde em praticidade, pois requer o ajustes de pesos para os pesos dos critérios.

Alguns dos problemas da Heurística Baseada em Inserção foram corrigidos na heurística Heurística Baseada em Inserção com Modelo Multicritério de Apoio a Decisão tendo um

aumento na qualidade da solução de 3,04%.

Por fim, as melhores heurísticas apresentadas foram utilizadas na fase de construção da heurística GRASP. Os resultados melhoraram significativamente os resultados da literatura. Em 64% das instâncias analisadas, a heurística proposta reduziu a distância trafegada total, acarretando em uma melhoria da qualidade das melhores soluções da literatura em 2,58%. Os resultados também comprovaram que a utilização de uma boa heurística nesta fase melhora a qualidade dos resultados finais da heurística baseada no GRASP.

6.1 Trabalhos Futuros

As heurísticas implementadas neste trabalho para solucionar o TSP são extremamente simples, existindo na literatura métodos mais sofisticados que poderiam ser utilizados, como o Concorde (Cook, 2005). Muitos métodos aqui apresentados utilizam a solução do TSP para definir o roteamento dos nós uma vez que eles já foram separados em grupos.

Outra questão que poderia ser aperfeiçoada é a heurística com modelo de decisão multicritério. Neste caso, a ferramenta utilizada neste trabalho (PROMETHEE) pode ser considerada como uma das mais simples entre as demais. Portanto, a aplicação e análise de outras ferramentas deve ser feita.

Outra extensão deste trabalho pode incluir na metaheurística GRASP: filtros, outras buscas locais e ainda aplicá-la em conjunto com outros métodos como a Busca Tabu, melhorando consideravelmente os resultados e o tempo computacional.

Além de melhorias nas abordagens apresentadas neste trabalho, a aplicação de métodos exatos, como o Branch-and-cut, pode ser uma alternativa vantajosa na solução de Problemas de Roteamento de Veículos com Coleta e Entrega Simultâneas. Existem poucas referências na literatura com esse tipo de aplicação.

Outra questão que pode ser explorada é a adaptação e aplicação das heurísticas e metaheurística apresentadas neste trabalho em Problemas Dinâmicos de Roteamento de Veículos com Coleta e Entrega Simultâneas. Estudos envolvendo este problema tem crescido nos últimos anos.

Referências Bibliográficas

Alvarenga, G. B. (2005). *Um Algoritmo Híbrido para o Problema de Roteamento de Veículos Estático e Dinâmico com Janela de Tempo*. PhD thesis, Universidade Federal de Minas Gerais.

Alvarenga, G. B.; de Abreu Silva, R. M. e Mateus, G. R. (2005). A hybrid approach for the dynamic vehicle routing problem with time windows. In *HIS '05: Proceedings of the Fifth International Conference on Hybrid Intelligent Systems*, pp. 61–67. IEEE Computer Society.

Angelelli, E. e Mansini, R. (2002). *The Vehicle Routing Problem with Time Windows and Simultaneous Pick-up and Delivery*, volume 519 of *Lecture Notes in Economics and Mathematical Systems*. Springer. Editado por Andreas Klose and M. Gracia Speranza and Luk N. Van Wassenhove.

Associação Nacional de Petróleo (2006). Associacão nacional de petróleo. www.anp.gov.br. Acessado em Janeiro-2006.

Çatay, B. (2006). An ant based algortihm for the vehicle routing problem with simultaneous delivery and pick-up. In *INFORMS Meeting*.

Beasley, J. E. (1983). Route first - cluster second methods for vehicle routing. *Omega*, 11(4):403–408.

Belfiore, P. P. e Yoshizaki, H. T. Y. (2006). Scatter Search para Problemas de Roteirização de Veículos com Frota Heterogênea, Janelas de Tempo e Entregas Fracionadas. *Produção*, 16:455 – 469.

Bergvinsdottir, K. B. (2004). The genetic algorithm for solving the dial-a-ride problem. Master's thesis, Technical University of Denmark.

Bianchessi, N. (2003). Approximation algorithms for the vehicle routing problem with simultaneous picku-up and delivery. In *Operations Research Peripatetic Postgraduate Programme*.

Bianchessi, N. (2004). Algoritmi di ricerca locale per il vehicle routing problem with simultaneous pick-up and delivery. Master's thesis, Technical University of Denmark.

Bianchessi, N. e Righini, G. (2007). Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers and Operations Research*, 34(2):578–594.

Brans, J.; Vincke, P. e Mareschal, B. (1986). How to select and how to rank projects: The promethee method. *European Journal of Operational Research*, 24:228–238.

Brans, J. P. e Vincke, P. (1985). A preference ranking organization method: The PROMETHEE method for multiple criteria decision-making. *Management Science*, 31(6):647–656.

Bräysy, O.; Dullaert, W. e Gendreau, M. (2004). Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics*, 10(6):587–611.

Campos, V. e Mota, E. (2000). Heuristic procedures for the capacitated vehicle routing problem. *Computational Optimization and Applications*, 16(3):265–277.

Caricato, P.; Ghiani, G.; Grieco, A. e Guerriero, E. (2003). Parallel tabu search for a pickup and delivery problem under track contention. *Parallel Computing*, 29:631–639.

Cavalcante, C. A. V. e Almeida, A. T. (2005). Modelo multicritério de apoio a decisão para o planejamento de manutenção preventiva utilizando PROMETHEE II em situações de incerteza. *Pesquisa Operacional*, 25(2):279–296.

Chin, A. J.; Kit, H. W. e Lim, A. (1999). A new ga approach for the vehicle routing problem. In *ICTAI '99: Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, p. 307. IEEE Computer Society.

Choi, E. e Tcha, D.-W. (2007). A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers and Operations Research*, 34(7):2080–2095.

Confederação Nacional de Transportes (2006). Confederação nacional de transportes. www.cnt.org.br. Acessado em Janeiro-2006.

Cook, W. (2005). Concorde TSP solver. www.tsp.gatech.edu/concorde/index.html. Acessado em Junho-2007.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(6):573–586.

Cordeau, J.-F. e Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594.

Cormen, T. H.; Stein, C.; Rivest, R. L. e Leiserson, C. E. (2001). *Introduction to Algorithms*. McGraw-Hill Higher Education.

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

Dell'Amico, M.; Righini, G. e Salani, M. (2005). A branch-and-price algorithm for the vehicle routing problem with simultaneous delivery and collection. *Transportation Science*, 40:235–247.

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

Fan, J.; Wang, X. e Chen, Q. (2007). A heuristic algorithm for multiple depots vehicle routing problem with backhauls. In *Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, volume 3, pp. 421–425. IEEE Computer Society.

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

Gelogullari, C. A. (2004). An exact algorithm for the vehicle routing problem with backhauls. *Computers and Operations Research*, 33:595–619.

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

Ghaziri, H. e Osman, I. H. (2003). A neural network algorithm for the traveling salesman problem with backhauls. *Computers and Industrial Engineering*, 44(2):267–281.

Gillett, B. E. e Miller, L. R. (1974). A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 22:340–349.

Goetschalckx, M. e Jacobs-Blecha, C. (1993). The vehicle routing problem with backhauls: Properties and solution algorithms. Technical Report MHRC-TR-88-l3, Georgia Institute of Technology.

Goldbarg, M. C. e Luna, H. P. L. (2000). *Otimização Combinatória e Programação Linear*. Ed. Campus.

Hernádez-Pérez, H. e Salazar-Gonzáles, J.-J. (2004). A branch-and-cut algorithm for the traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145(1):126–139.

Ho, S. C. e Gendreau, M. (2006). Path relinking for the vehicle routing problem. *Journal of Heuristics*, 12(1-2):55–72.

Ho, S. C. e Haugland, D. (2004). Tabu search heuristics for the probabilistic dial-a-ride problem. Technical Report 286, University of Bergen - Department of Informatics.

Lau, H. e Liang, Z. (2001). Pickup and delivery problem with time windows: Algorithms and test case generation. In *13th IEEE International Conference on Tools with Artificial Intelligence*, pp. 333–340.

Lawler, E. L.; Lenstra, J. K.; Kan, A. H. G. R. e Shmoys, D. B. (1985). *The Traveling Salesman Problem*. John Wiley & Sons, New York.

Li, H. e Lim, A. (2001). A metaheuristic for the pickup and delivery problem with time windows. In *Proceedings of the 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01)*, p. 160, Washington, DC, USA. IEEE Computer Society.

Lopes, M. R. C. M. (2005). Uso das metodologias PROMETHEE e F-PROMETHEE na avaliação de clientes. Master's thesis, Universidade Federal do Paraná.

Lu, Q. e Dessouky, M. M. (2006). A new insertion-based construction heuristic for solving the pickup and delivery problem with hard time windows. *European Journal of Operational Research*, 175:672–687.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pickup points. *Transportation Research-A*, 23A(5):377–386.

Montané, F. A. T.; Ferreira, V. J. M. F. e Galvão, R. D. (1997). Determinação de rotas para empresa de entrega expressa. *Pesquisa Operacional*, 17:107–135.

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

Montemanni, R.; Gambardella, L. M.; Rizzoli, A. E. e Donati, A. V. (2002). A new algorithm for a dynamic vehicle routing problem based on ant colony system. Technical report.

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

Mosheiov, G. (1998). Vehicle routing with pick-up and delivery: Tour-partition heuristics. *Computer and Industrial Engineering*, 34(3):669–684.

Nanry, W. P. e Barnes, W. J. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 34(2):107–121.

Ombuki, B.; Ross, B. J. e Hanshar, F. (2006). Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence*, 24(1):17–30.

Pankratz, G. (2005). A grouping genetic algorithm for the pickup and delivery problem with time windows. *OR Spectrum*, 27(1):21–41.

Parragh, S. N.; Doerner, K. F. e Hartl, R. F. (2006). A survey on pickup and delivery models. http://www.univie.ac.at/bwl/prod/research/surveyPDP/.

Potvin, J.-Y.; Duhamel, C. e Guertin, F. (1996). A genetic algorithm for vehicle routing with backhauling. *Applied Intelligence*, 6(4):345–355.

Psaraftis, H. N. (1980). A dynamic programming approach to the single vehicle, many-to-many immediate request dial-a-ride problem. *Transportation Science*, 17:351–357.

Reimann, M.; Doerner, K. e Hartl, R. F. (2002). Insertion based ants for vehicle routing problems with backhauls and time windows. In *ANTS '02: Proceedings of the Third International Workshop on Ant Algorithms*, pp. 135–148. Springer-Verlag.

Ropke, S. e Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

Rousseau, L.-M.; Gendreau, M. e Pesant, G. (2002). Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of Heuristics*, 8(1):43–58.

Ruland, K. S. e Rodin, E. Y. (1997). The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computer and Mathematics with Aplications*, 33(12):1–13.

Salhi, S. e Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of Operational Research Society*, 50:345–355.

Savelsbergh, M. e Sol, M. (1995). The general pickup and delivey problem. *Transportation Science*, 29(1):17–29.

Vincke, P. (1992). *Multicriteria Decision-aid*. John Wiley & Sons.

Vural, A. V. e Çatay, B. (2003). A ga based meta-heuristic for capacitated vehicle routing problem with simultaneous pick-up and delivery. Master's thesis, Faculty of Management, Istanbul Technical University.

Xu, H.; Chen, Z.-L.; Rajagopal, S. e Arunapuram, S. (2003). Solving a practical pickup and delivery problem. *Transportation Science*, 37(3):347–364.