UNIVERSIDADE FEDERAL DO PAMPA

Ciência da Computação

MARCELO CAGGIANI LUIZELLI

PROBLEMA DE ROTEAMENTO DE VEÍCULOS APLICADO AO PROBLEMA DE DESPACHO DE ORDENS DE SERVIÇO

Trabalho de Conclusão de Curso

Alegrete
2012

Marcelo Caggiani Luizelli

Problema de Roteamento de Veículos Aplicado ao Problema de Despacho de Ordens de Serviço

Trabalho de Conclusão de Curso apresentado como parte das atividades para obtenção do título de bacharel em Ciência da Computação na Universidade Federal do Pampa.

Orientador: Fábio Natanael Kepler

Co-orientador: Vinícius Jacques Garcia

Alegrete

2012

MARCELO CAGGIANI LUIZELLI

PROBLEMA DE ROTEAMENTO DE VEÍCULOS APLICADO AO PROBLEMA DE DESPACHO DE ORDENS DE SERVIÇO

Trabalho de Conclusão de Curso apresentado como parte das atividades para obtenção do título de bacharel em Ciência da Computação na Universidade Federal do Pampa.

Trabalho apresentado e aprovado em: 05 de Janeiro de 2012.
Banca Examinadora:

Prof. Dr. Fabio Nathanael Kepler
Orientador
Ciência da Computação - UNIPAMPA

Prof. Dr. Vinicius Jacques Garcia
Coorientador
UFSM

Prof. Dr. Cleo Zanella Billa
Ciência da Computação - UNIPAMPA

Prof. Dr. Sergio Luis Sardi Mergen
Ciência da Computação - UNIPAMPA

Este trabalho é dedicado À Deus À minha família À minha namorada Fernanda e a meus amigos.

Agradecimentos

Agradeço primeiramente a Deus pela oportunidade do desenvolvimento e aprimoramento das minhas faculdades. Aos meus pais, Paulo Renato e Maria Ester, por serem os melhores tutores neste processo de aprendizado e por serem fontes inesgotáveis de amor, compreensão, carinho e incentivo. Aos meus irmãos, Rafael e Paola, por serem os melhores amigos e companheiros desde sempre. Obrigado a vocês!

Um agradecimento especial para minha vó Jurema, para a Iaia e também para aqueles que já se foram: vô Ivo, Ieie, vó Maria e Vô Júlio.

À toda minha família pelo apoio dado para que eu chegasse aqui. É graças a vocês que alcanço essa vitória.

Ao meu amor, Fernanda, por ser extremamente companheira, amiga, compreensiva, incentivadora, carinhosa e consoladora durante todo esse período de graduação. Amor, obrigado por todos os momentos!

Ao meu orientador, prof. Vinícius Jacques Garcia, por propiciar toda a fundamentação teórica-prática ao desenvolvimento deste trabalho. Agradeço a oportunidade de ter sido orientado assim como a convivência e a troca de saberes. Também agradeço pelo grande incentivo, pela confiança, pela paciência e por ter acreditado no meu potencial. Muito obrigado!

Agradeço aos meus amigos que durante estes quatro anos de formação estiveram ajudando, contribuindo e incentivando o aperfeiçoamento do conhecimento adquirido. Em especial um obrigado ao Adriano e ao Tony por terem sido companheiros em inúmeras missões impossíveis e pela grande amizade construída. Também um especial obrigado ao Rafael e a Susi, por serem ótimos amigos, companheiros e, não poderia deixar de mencionar, pelas inúmeras noites infindáveis de jogatina e pelos litros e mais litros de cerveja consumidos. Obrigado!

Por fim, agradeço a todas as pessoas que direta ou indiretamente influenciaram positivamente no desenvolvimento deste trabalho.

O meu muito obrigado a todos vocês!

Resumo

O problema de roteamento de veículos é um problema amplamente estudado na área de otimização combinatória pois possui uma grande relevância científica e prática. Pela necessidade de otimização dos sistemas logísticos faz-se necessário o aprimoramento das técnicas de resolução principalmente quando os problemas envolvidos são reais. Pela semelhança com o problema de roteamento de veículos, o problema de despacho de ordens de serviço é abordado neste trabalho como uma de suas possíveis variações. Este trabalho analisa as principais abordagens heurísticas e meta-heurísticas à resolução do problema de roteamento bem como propõe um algoritmo de busca local aplicado a qualquer de suas variantes. Por fim, analisa-se tais abordagens com o intuito de mensurar quais poderão futuramente ser aplicadas com maior êxito ao problema de despacho de ordens de serviço.

Palavras-chave: Problema de roteamento de veículos, heurísticas, meta-heurísticas, otimização.

Abstract

The vehicle routing problem is a problem widely studied in the field of combinatorial optimization because it has a great scientific and practical relevance. The need for optimization of logistics systems is necessary to improve the resolution techniques especially when the issues involved are real. For the resemblance to the vehicle routing problem, the problem of dispatch service order is addressed in this work as one of its possible variations. This paper examines the main approaches heuristics and meta-heuristics for solving the routing problem and propose a local search algorithm applied to any of its variants. Finally, we analyze these approaches in order to measure which may one day be applied more successfully to the problem of dispatch service order.

Keywords: Vehicle routing problem, heuristics, metaheuristics, optimization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

PRV	Problema de Roteamento de Veículos
PRVC	Problema de Roteamento de Veículos Capacitado
PRVMD	Problema de Roteamento de Veículos Multi-depósito
PCV	Problema do Caixeiro Viajante
SbAS	*Saving based Ant System*
VNS	*Variable Neighborhood Search*
GRASP	*Greedy Randomized Adaptative Search Procedure*
ACO	*Ant Colony Optimization*
ANEEL	Agência Nacional de Energia Elétrica
PDOS	Problema de Despacho de Ordens de Serviços
BI	*Best Improvement*
FI	*First Improvement*
LGPL	*Lesser General Public License*

SUMÁRIO

1 Introdução

Com o final da segunda guerra mundial aparecem duas novas áreas multidisciplinares: a pesquisa operacional e a otimização combinatória (AARDAL; NEMHAUSER; WEISMANTEL, 2005), (TAHA, 2007). A pesquisa operacional é uma subárea da administração que surgiu no intuito de investigar modelos matemáticos que pudessem descrever processos do mundo real com maior exatidão. Por sua vez, a área de otimização combinatória é uma subárea da matemática aplicada que se desenvolveu no estudo da resolução de modelos oriundos da atuação de cientistas na pesquisa operacional. Relacionado com as duas áreas encontra-se a ciência da computação que se preocupa com as questões de projeto e análise de algoritmos eficientes à resolução dos modelos envolvidos.

Devido ao crescimento industrial ocorrido após a segunda guerra mundial, houve a necessidade de aprimorar os sistemas logísticos existentes até então, advindo daí toda a teoria e o apelo ao desenvolvimento dos sistemas de apoio à decisão. A primeira ocorrência evidente de preocupação com os sistemas de transporte é observada em 1959 no trabalho de Dantzig e Ramser (DANTZIG; RAMSER, 1959), no qual há a primeira formulação de um problema de roteamento de veículos para o abastecimento de postos de gasolina através de caminhões tanque.

O problema de roteamento de veículos (PRV) é um dos problemas em otimização combinatória com grande relevância científica (TOTH; VIGO, 2002). O PRV forma uma ampla classe de algoritmos que, genericamente, resolvem o problema de determinar uma sequência de atendimentos a partir de um ou mais depósitos dispondo de uma frota de veículos. O objetivo dos algoritmos para o PRV é prover soluções que estejam de acordo com as restrições inerentes a cada modelagem matemática.

Existem inúmeras variações do PRV como: o problema de roteamento de veículos capacitado, o problema de roteamento de veículos multi-depósito, o problema de roteamento de veículos com janelas de tempo, o problema de roteamento de veículos com entrega e recolhimento, entre outros (TOTH; VIGO, 2002), sendo que cada variante do PRV surgiu em um contexto

prático de aplicação. O PRV possui um número extraordinário de aplicações nos mais variados setores, como: indústria, comércio, segurança, etc. Destacam-se, ainda, a aplicabilidade em sistemas de distribuição de produtos diversos, recolhimento de lixo urbano, serviços de emergência, patrulhamento policial, roteamento de pacotes em redes de computadores, roteamento de satélites, entre outros.

Empresas dos mais variados setores investem em logística a fim de reduzir custos e aumentar o diferencial competitivo. Toth e Vigo (TOTH; VIGO, 2002) destacam que o uso de técnicas computacionais aplicadas ao planejamento logístico podem gerar economias de 5% a 20% no transporte, sendo que os custos com logística representam até 20% do total do bem/serviço.

Neste contexto, empresas concessionárias de energia elétrica também investem em sua estratégia operacional. Normalmente, concessionárias de energia elétrica devem prover certos tipos de serviços que somente são possíveis com o deslocamento de unidades de atendimentos. Cada tarefa, seja qual for a finalidade, é intitulada de ordem de serviço (OS). Como exemplo de ordens de serviço pode-se citar: ligamento/desligamento de energia elétrica, manutenção da rede elétrica, serviços de emergência, entre outros.

Tais ordens de serviços são geograficamente distribuídas nos centros urbanos e cabe a concessionária estabelecer a ordem dos atendimentos de tal forma que se cumpra os prazos estabelecidos pela agência reguladora. Em um grande centro, onde há uma grande quantidade de ordens a ser executadas, o emprego de métodos computacionais é de extrema relevância para que os recursos sejam utilizados da melhor maneira possível bem como sejam cumpridos todos os prazos de atendimento.

Advindo da semelhança de características presentes entre o problema de roteamento de veículos e o problema de despacho de ordens de serviço, este trabalho se propõe ao estudo dos métodos já desenvolvidos para o problema de roteamento a fim de analisar o potencial emprego das técnicas ao problema de despacho de ordens de serviço.

1.1 Objetivos

Este trabalho de pesquisa tem o objetivo geral de analisar a possível aplicabilidade de métodos de resolução do problema de roteamento de veículos no problema de despacho de ordens de serviço. Como objetivos específicos destam-se: analisar heurísticas e meta-heurísticas desenvolvidas para o problema de roteamento quanto à qualidade, desempenho computacional e características, desenvolver um método heurístico que contemple características não observadas nos métodos clássicos e avaliar quais abordagens possuem maior potencial de serem aplicadas

ao problema de despacho de ordens de serviço.

1.2 Justificativa

Segundo Lenstra et al. (LENSTRA; KAN, 1981), o problema de roteamento de veículos é um dos problemas $NP-dificil$ conhecidos. Advindo do fato de ainda não ser conhecido algoritmo determinístico em tempo polinomial para estes problemas, existe um grande apelo científico no estudo desta classe de problemas.

A resolução deste tipo de problema pelo método da enumeração das soluções é um procedimento inviável em termos computacionais já que o número de soluções para este tipo de problema é em geral exponencial em relação ao tamanho da entrada do problema. Para o problema do caixeiro viajante, que é um problema correlato ao de roteamento e com menor complexidade, em um exemplo com 20 cidades, tem-se um número extremamente grande de soluções viáveis: 6.10^{16}. Para um computador que avalia cada solução em 10^{-8} segundos, desprenderia 19 anos para a total enumeração das soluções. Diante deste fato, a enumeração somente é utilizada quando o problema é extremamente pequeno, o que não condiz com problemas reais, como o problema de despacho de ordens de serviço.

Para tanto, nas últimas décadas houve um grande avanço no aperfeiçoamento das técnicas de resolução para estes problemas. Foram desenvolvidos métodos inteligentes que contemplam estratégias de busca de soluções que evitam a enumeração explícita. Além disto, foram desenvolvidos inúmeros métodos heurísticos e meta-heurísticos que agregam informações inerentes a estes problemas no intuito de resolvê-los com uma certa qualidade em um tempo aceitável.

Diante destes argumentos, a análise e o desenvolvimento de métodos computacionais inteligentes são de grande relevância para garantir um equilíbrio entre qualidade e custo computacional das soluções quando se trata de problemas reais.

1.3 Estrutura da Monografia

O restante desta monografia está dividido da seguinte maneira:

- No capítulo 2 são apresentadas algumas terminologias e conceitos primordiais ao entendimento do restante do texto. Apresenta-se a revisão bibliográfica acerca do problema de roteamento de veículos e do problema de despacho de ordens de serviço. Este capítulo tem o intuito de descrever matematicamente o problema de roteamento bem como

o problema de despacho de ordens de serviço. Além disto, são apresentados os principais métodos de resolução encontrados na literatura para a resolução do problema de roteamento de veículos capacitado.

- No capítulo 3 são apresentados os algoritmos implementados e o algoritmo proposto para o problema de roteamento de veículos. Este capítulo tem o objetivo de descrever textualmente e apresentar as particularidades da implementação de cada um dos métodos analisados.
- No capítulo 4 os resultados obtidos são apresentados e discutidos.
- No capítulo 5 são realizadas as considerações finais.

2 *Revisão Bibliográfica*

Este capítulo tem a finalidade de explanar sobre os conceitos em otimização combinatória relevantes ao entendimento do restante do texto. Além disto, apresentam-se as variações do problema de roteamento de veículos que são mais adequadas de serem aplicadas ao problema de despacho de ordens de serviço, os modelos matemáticos envolvidos e os tipos de métodos de resolução utilizados neste tipo de problema combinatório.

2.1 Conceitos em Otimização Combinatória

O correto entendimento das partes constituintes de um modelo matemático em otimização combinatória exige o conhecimento de algumas terminologias como: função objetivo, restrições, variáveis de decisão, factibilidade de soluções, entre outras.

A otimização consiste na análise de problemas complexos de decisão ou de alocação de recursos (LUENBERGER; YE, 2008). Uma solução para um problema complexo de decisão envolve selecionar valores para um conjunto de variáveis inter-relacionadas, tendo como objetivo aumentar a qualidade da solução. A atribuição de valores a essas variáveis é limitada pelas restrições do modelo matemático.

Para manter a clareza das próximas seções, explana-se brevemente sobre conceitos fundamentais ao entendimento dos modelos matemáticos e algoritmos envolvidos.

2.1.1 Função Objetivo

A função objetivo é definida por Goldbarg e Luna (GOLDBARG; LUNA, 2005) como uma função $F : R^n \longrightarrow R$ contínua e geralmente diferenciável. Dada uma solução S para um problema de otimização, o custo associado a S é o valor correspondente a função objetivo. Um problema de otimização pode ser entendido como o desejo de obter um conjunto $S^* \in F$, onde

F é definido como o conjunto de todas as soluções possíveis, satisfazendo:

$$Custo(S^*) \geq Custo(S), \forall S \in F \tag{2.1}$$

ou

$$Custo(S^*) \leq Custo(S), \forall S \in F \tag{2.2}$$

onde a equação 2.1 se refere a função objetivo de um problema de maximização, a qual se deseja obter o maior valor para a função objetivo; a equação 2.2 se refere a um problema de minimização o qual se deseja obter o menor valor associado a função objetivo.

O custo de uma solução $Custo(S)$ é calculado a partir de fatores inerentes ao problema em questão. Para o PRV, pode-se desejar a minimização do custo total do transporte, a minimização do número de veículos para o atendimento, a minimização do tempo de rota, a minimização de penalidades associadas a serviços parciais, entre outros (TOTH; VIGO, 2002).

2.1.2 Restrições

Em um modelo matemático, as restrições são limites definidos por hiperplanos em um espaço em R^n para um dado problema. Como mencionado, as restrições limitam a escolha de valores para as variáveis em sistema de decisão ou alocação de recursos (LUENBERGER; YE, 2008).

Restrições no PRV podem ter relação com a: necessidade de retornar ao depósito a cada término de rota, visitar cada cliente apenas uma vez, com o respeito à capacidade dos veículos na medida em que o somatório das demandas dos clientes visitados é igual ou inferior a esta capacidade, entre outras (TOTH; VIGO, 2002).

As restrições dos problemas são modeladas como equações ou inequações que necessariamente devem ser satisfeitas na resolução dos problemas.

2.1.3 Variáveis de Decisão

Variáveis de decisão são incógnitas a serem determinadas pela resolução dos modelos matemáticos.

Em problemas de otimização combinatória as variáveis podem ser: inteiras, reais ou mistas.

De acordo com o problema envolvido, o tipo de variável de decisão mais adequado deve ser definido. Para cada problema que envolva algum tipo de variáveis de decisão existem métodos matemáticos específicos para a resolução. Especificamente, no PRV, as variáveis de decisão são inteiras e particularmente binárias. Tais variáveis indicam a presença ou não de segmentos de rotas presentes em uma solução.

2.1.4 Factibilidade das soluções

Uma solução é factível se respeita as restrições do modelo matemático, ou seja, se a solução está contida no poliedro formado pelo conjunto de equações ou inequações que definem as restrições do problema. Caso contrário a solução é dita ser infactível ou inviável. Uma solução ótima é uma solução factível especial onde o valor da função objetivo está otimizado. Em problemas de minimização, uma solução ótima é aquela que apresenta menor custo possível associado a função objetivo. Já em problemas de maximização, uma solução ótima é aquela que apresenta maior custo possível associado à função objetivo.

2.1.5 Espaço de soluções

Toda a atribuição de valores válidos as variáveis de decisão geram soluções. O espaço de soluções diz respeito ao conjunto de possíveis valores que geram soluções dentro da factibilidade.

2.1.6 Vizinhança

Dada uma solução factível, representada pela atribuição de valores as variáveis de decisão, a modificação de valores em um conjunto destas variáveis acarreta em um solução vizinha, desde que a nova solução esteja contida dentro do espaço de soluções. Uma solução vizinha está nas proximidades da solução atual e caracteriza-se por ser semelhante.

2.2 Variantes do Problema de Roteamento de Veículos

Desde que Dantzig e Ramser (DANTZIG; RAMSER, 1959) propuseram o primeiro modelo de um PRV em 1959, surgiram inúmeras outras variantes correlatas. Neste trabalho, estuda-se as variantes capacitada (PRVC) e multi-depósito (PRVMD) pois possuem semelhanças com o problema de despacho de ordens de serviço e portanto podem ser aplicadas sem muitas alter-

ações. Para tanto, apresenta-se as definições formais dos problemas bem como suas formulações matemáticas.

2.2.1 Problema do caixeiro viajante (PCV)

O problema do caixeiro viajante (PCV) é um dos trabalhos mais tradicionais e conhecidos em otimização combinatória. O problema remete-se aos antigos caixeiros que visitavam as cidades vendendo os seus produtos.

O PCV é definido formalmente por Goldbarg e Luna (GOLDBARG; LUNA, 2005) como: dado um grafo $G = (N, E)$, onde N é o conjunto de cidades e E o conjunto de arestas com os respectivos custos associados a quaisquer dois pontos $n_1, n_2 \in N$, deseja-se determinar uma sequência de cidades $T = \{n_1, n_2, n_3, \ldots, n_n\}$ que possua um custo associado mínimo. O problema matemático consiste, segundo Dantzig (DANTZIG; FULKERSON; JOHNSON, 1954), em determinar o melhor itinerário a cumprir, sendo que o caixeiro deve partir de uma cidade qualquer, visitar todas as demais uma única vez e retornar à cidade de partida.

Este itinerário de visitas é conhecido como circuito hamiltoniano (CORMEN et al., 2002). Neste caso, busca-se um circuito hamiltoniano de custo mínimo.

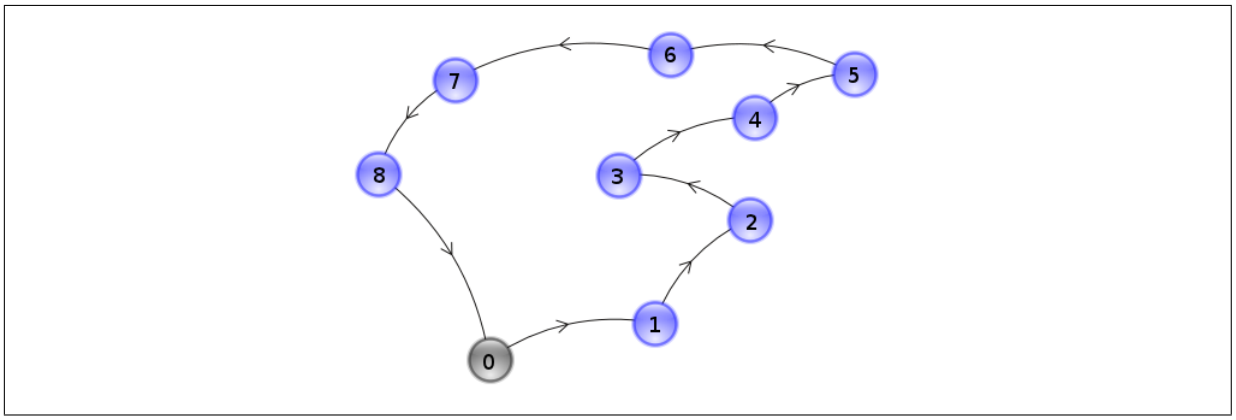

FIGURA 2.1: Exemplo de um circuito hamiltoniano. Cada ponto na figura ilustra uma cidade. Cada seta indica a direção do caminho a percorrer pelo caixeiro viajante.

2.2.2 Problema de roteamento de veículos capacitado (PRVC)

Segundo Toth e Vigo (TOTH; VIGO, 2002), o problema de roteamento de veículos é uma extensão do problema do caixeiro viajante. O PRVC objetiva encontrar um ou mais circuitos hamiltonianos de custo mínimo, sendo que cada um representa um veículo e todos eles possuem a mesma origem, o depósito. Neste caso, cada cidade possui uma certa demanda a qual deve ser suprida por apenas um único veículo. Deve-se respeitar as restrições de capacidade de cada

veículo e a atomicidade do atendimento. O PRVC é composto por apenas um único depósito que dispõe de veículos homogêneos, ou seja, com capacidades idênticas.

Formalmente, o PRVC é formulado por Laporte (LAPORTE, 1992) como: dado um grafo $G = (V, E)$, onde $V = \{v_1, v_2, v_3, \ldots, v_n\}$ corresponde aos pontos de demanda e $E = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ o conjunto de arestas entre os pontos de demanda.

A cada cliente $v_i \in V$ existe uma demanda $q_i \in Q$ associada que deve ser suprida por algum veículo. Ao conjunto E está associado uma matriz simétrica de custos não negativos C que representa o custo necessário a percorrer dois nós i e j , sendo $c_{ij} = c_{ji}$. Para tanto se devem respeitar as seguintes restrições:

- cada cidade $v \in V - 1$ deve ser visitada exatamente uma única vez.
- todas as rotas devem iniciar e terminar no depósito.
- atender as restrições de capacidade dos veículos.

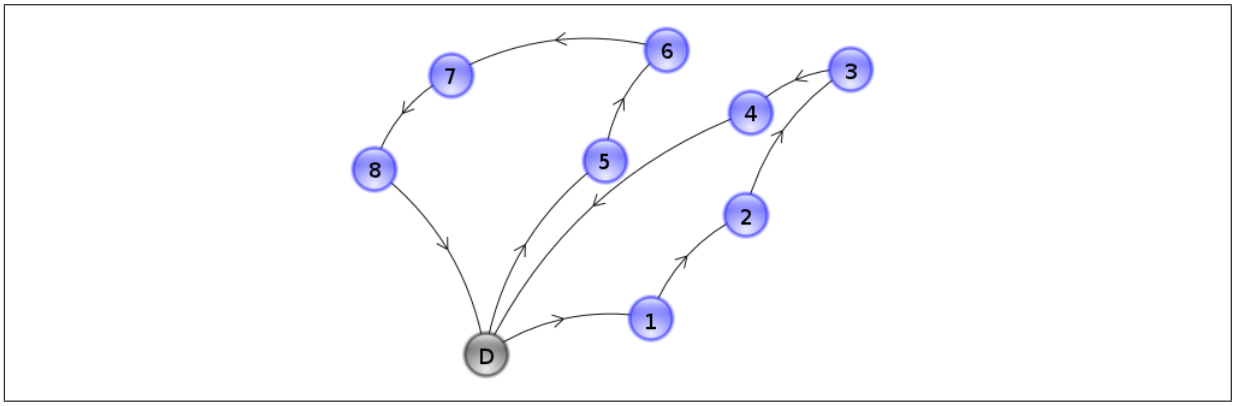

FIGURA 2.2: Exemplo de uma solução para o PRVC. Nota-se a presença de duas rotas ou dois circuitos hamiltonianos.

Modelo matemático

Existem inúmeras modelagens matemáticas para um mesmo problema de otimização. Geralmente, o modelo matemático adotado ou desenvolvido considera questões referentes ao modo de resolução do problema em questão. O modelo adotado neste trabalho para o PRVC é o modelo desenvolvido por Fisher e Jaikumar (FISHER; JAIKUMAR, 1981). Tal modelo contempla as restrições elementares do problema de roteamento capacitado. Segue abaixo a modelagem.

$$Min \sum_{i,j} \sum_{k} c_{ij} x_{ijk} \tag{2.3}$$

Sujeito a:

$$\sum_{k} y_{ik} = 1, i = 2 \ldots n \tag{2.4}$$

$$\sum_{k} y_{ik} = mi = 1 \tag{2.5}$$

$$\sum_{i} q_i y_{ik} \leq Q_k k = 1, \ldots, m \tag{2.6}$$

$$\sum_{j} x_{ijk} = \sum_{j} x_{jik} = y_{ik} i = 1, \ldots, n \, k = 1, \ldots, m \tag{2.7}$$

$$\sum_{j} x_{ijk} \leq |S| - 1 \forall S \subseteq \{2, \ldots, n\}, k = 1, \ldots, m \tag{2.8}$$

$$y_{ik} \in \{0,1\} \, i = 1, \ldots, n \, k = 1, \ldots, m \tag{2.9}$$

$$x_{ijk} \in \{0,1\} \, i, j = 1, \ldots, n \, k = 1, \ldots, m \tag{2.10}$$

Onde:

- $x_{ijk} \equiv$ variável binária que assume valor 1 quando o veículo k visita o cliente j imediatamente após o cliente i, 0 caso contrário.
- $y_{ik} \equiv$ variável binária que assume valor 1 se o cliente i é visitado pelo veículo k, 0 caso contrário.
- $q_i \equiv$ corresponde a demanda do cliente i.
- $Q_k \equiv$ corresponde a capacidade do veiculo k.
- $c_{ij} \equiv$ corresponde ao custo de percorrer o trecho que vai do cliente i ao cliente j.

A função objetivo 2.3 minimiza a distância total da solução, calculando a soma das distâncias entre pontos de demanda presentes na solução. A restrição 2.4 garante que cada ponto de demanda seja visitado exclusivamente por um veículo. A restrição 2.5 garante que o depósito receba uma visita de todos os veículos. A restrição 2.6 garante que a capacidade dos veículos não seja excedida. A restrição 2.7 garante que uma rota tenha término sempre no depósito e não em um ponto de demanda. A restrição 2.8 garante a não ocorrência de *subtours* nas rotas. As restrições 2.9 e 2.10 são as restrições de domínio das variáveis de decisão do problema.

2.2.3 Problema de roteamento de veículos multi-depósito (PRVMD)

O problema de roteamento de veículos multi-depósito pode ser definido como uma extensão do PRVC. Segundo Cordeau et al. (CORDEAU; GENDREAU; LAPORTE, 1997), o PRVMD apresenta mais de um depósito e a cada depósito estão vinculados veículos que realizam os atendimentos aos pontos de demanda.

Ainda segundo Cordeau et al. (CORDEAU; GENDREAU; LAPORTE, 1997), o PRVMD pode ser formulado como: dado um grafo $G = (V,E)$, onde $V = v_0, v_1, ..., v_n$ é o conjunto de vértices e $E = (v_i, v_j)^{k,l}$ o conjunto de arestas. As variáveis k,l referem-se ao índice do veículo e ao depósito associado, respectivamente. Cada aresta $(v_i, v_j)^{k,l}$ é associada a um custo não negativo c_{ijkl} proporcional ao tempo de rota do veículo k que percorre o consumidor i até o consumidor j partindo do depósito l. Define-se as variáveis binárias a_{rl}, x_{ijkl} e y_ir. A variável a_{rl} assume valor igual a 1 se e somente se uma rota r está associada a um depósito l. A variável x_{ijkl} é igual a 1 se e somente se o veículo k visita o consumidor j imediatamente depois do consumidor j e está atribuído ao depósito l. A variável y_{ir} é igual a 1 se e somente se um consumidor está atribuído a uma rota $r \in C_i$.

Neste modelo, todos os veículos possuem a mesma capacidade Q. O objetivo do PRVMD é servir todos os consumidores com um custo mínimo de distância satisfazendo as seguintes restrições:

- Cada consumidor é atendido exclusivamente por um único veículo.
- Cada veículo inicia e termina uma rota no mesmo depósito.
- A demanda acumulada em cada rota deve ser inferior ou igual a capacidade do veículo.
- Tempo de rota de cada veículo deve ser inferior ou igual ao tempo máximo de rota.

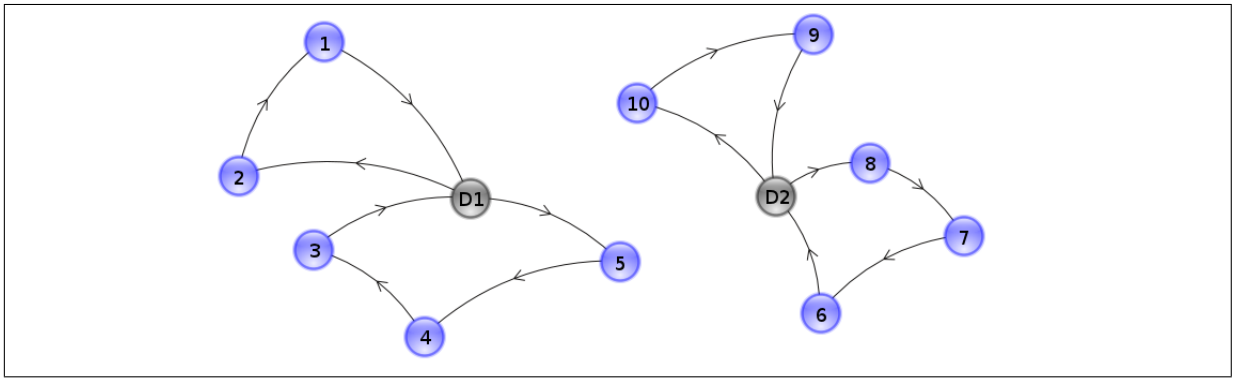

FIGURA 2.3: Exemplo de uma solução para o PRVMD. Nota-se a presença de dois depósitos (representados como D1 e D2) e a cada depósito vinculado duas rotas.

Modelo matemático

O modelo matemático utilizado para descrever o PRVMD é o formulado por Cordeau et al. (CORDEAU; GENDREAU; LAPORTE, 1997). Tal modelo pode ser generalizado para outras variações do PRV como, o problema de roteamento de veículos periódico.

$$Min \sum_{i}^{n}\sum_{j}^{n}\sum_{k}^{m}\sum_{l}^{t} c_{ijkl}x_{ijkl} \tag{2.11}$$

Sujeito a:

$$\sum_{r\in C_i} y_{ir} = 1 \qquad (i = 1\ldots n); \tag{2.12}$$

$$\sum_{j=0}^{n}\sum_{k=1}^{m} x_{ijkl} - \sum_{r\in C_i} a_{rl}y_{ir} = 0, \qquad (i = 1\ldots n; l = 1\ldots t); \tag{2.13}$$

$$\sum_{i=0}^{n} x_{ihkl} - \sum_{j=0}^{n} x_{hjkl} = 0, \qquad (h = 0\ldots n; k = 1\ldots m; l = 0\ldots t); \tag{2.14}$$

$$\sum_{j=1}^{n} x_{0jkl} \leq 1 \qquad (k = 1\ldots m; l = 1\ldots t); \tag{2.15}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} q_i x_{ijkl} \leq Q_k \qquad (k = 1\ldots m; l = 1\ldots t); \tag{2.16}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} (c_{ijkl} + d_i)x_{ijkl} \leq D_k \qquad (k = 1\ldots m; l = 1\ldots t); \tag{2.17}$$

$$\sum_{v_i\in S}\sum_{v_j\in S} x_{ijkl} \leq |S| - 1 \qquad (k = 1\ldots m; l = 1\ldots t; S \subseteq V\ 0; |S| \geq 2); \tag{2.18}$$

$$x_{ijkl} \in 0,1 \qquad (i = 0\ldots n; j = 1\ldots n; k = 1\ldots m; l = 1\ldots t); \tag{2.19}$$

$$y_{ir} \in 0,1 \qquad (i = 0\ldots n; r \in C_i); \tag{2.20}$$

A função objetivo 2.11 minimiza a distância total da solução, calculando a soma das distâncias entre pontos de demanda presentes na solução. A restrição 2.12 garante que cada consumidor seja atribuído a uma rota. Restrição 2.13 garante que cada consumidor seja visitado somente por um veículo de um depósito. A restrição 2.14 garante a rota não termine em um consumidor. Restrição 2.15 especifica que cada veículo é utilizado no máximo em um depósito. Limites de capacidade e de duração de rota são impostas pelas restrições 2.16 e 2.17. A restrição

2.18 elimina os clássicos *subtours*. As restrições 2.19 e 2.20 definem o domínio das variáveis de decisão.

2.3 Problema de despacho de ordens de serviço (PDOS)

O problema de despacho de ordens de serviço é oriundo dos serviços prestados pelas concessionárias de energia elétrica. Tais concessionárias devem manter um alto grau de disponibilidade no atendimento de ordens de serviço e garantir que o maior número de ordens seja atendido nos seus respectivos prazos. No Brasil, a Agência Nacional de Energia Elétrica (ANEEL) regulamenta a prestação de serviço por parte das concessionárias. Existem basicamente quatro tipos de ordens diferentes: ordens reguladas, ordens comerciais, ordens técnicas e ordens emergenciais. As ordens reguladas são aquelas que possuem data e tempo de execução predefinido; as ordens comerciais são aquelas relacionadas à conexão e desconexão de consumidores à rede de distribuição; as ordens técnicas são aquelas caracterizadas por atividades de manutenção e inspeção na rede de distribuição; e as ordens emergenciais são aquelas onde envolvem qualquer tipo de serviço com tempo restrito de atendimento. A ANEEL estabelece tempos máximos para a execução de ordens reguladas com possíveis sansões em caso de descumprimento.

Para manter um equilíbrio entre os recursos disponíveis e a demanda requerida de tempo de execução entre as diferentes ordens, faz-se necessária a utilização de métodos algorítmicos para auxiliar o processo de gerenciamento e alocação de recursos. Com o gerenciamento eficiente dos recursos, neste caso o tempo disponível de atendimento, atende-se as exigências da ANEEL e os custos com a logística são minimizados.

Equipes dispostas em bases estratégicas suprem a cadeia de atendimentos. Tais equipes podem possuir especialização em determinados tipos de atendimentos, como atendimentos às ordens técnicas. Cada equipe também possui uma jornada máxima de trabalho, a qual não deve ser excedida. Cada ordem de serviço possui algum requisito funcional como, por exemplo, possuir atributo emergencial.

Formalmente, o PDOS é definido como um grafo $G = (N, E)$, onde:

- N é o conjunto de nós que representam as ordens de serviço e as equipes.
- $E = \{(i, j) : i, j \in N\}$ corresponde as arestas. A distância $c_{i,j}$ associada a cada aresta $(i, j) \in N$ é definida pela distância euclidiana entre cada $(i, j) \in E$.

As seguintes restrições são assumidas:

- cada ordem de serviço deve ser atendida por somente uma equipe;
- toda equipe deve ter sua carga-horária de trabalho respeitada;
- cada rota inicia e termina no depósito;
- o tempo de viagem de uma rota não pode exceder o limite D;
- a demanda total de um rota não pode exceder a capacidade Q do veículo;

Diante das características e restrições envolvidas no problema de despacho de ordens de serviço, uma das possíves maneiras de abordá-lo de forma algorítmica é fazendo uso de modelos de problemas correlatos como, o PRVMD ou o PRVC. O objetivo principal do PDOS é a minimização do custo envolvido com a logística sem descumprir os prazos máximos de atendimento.

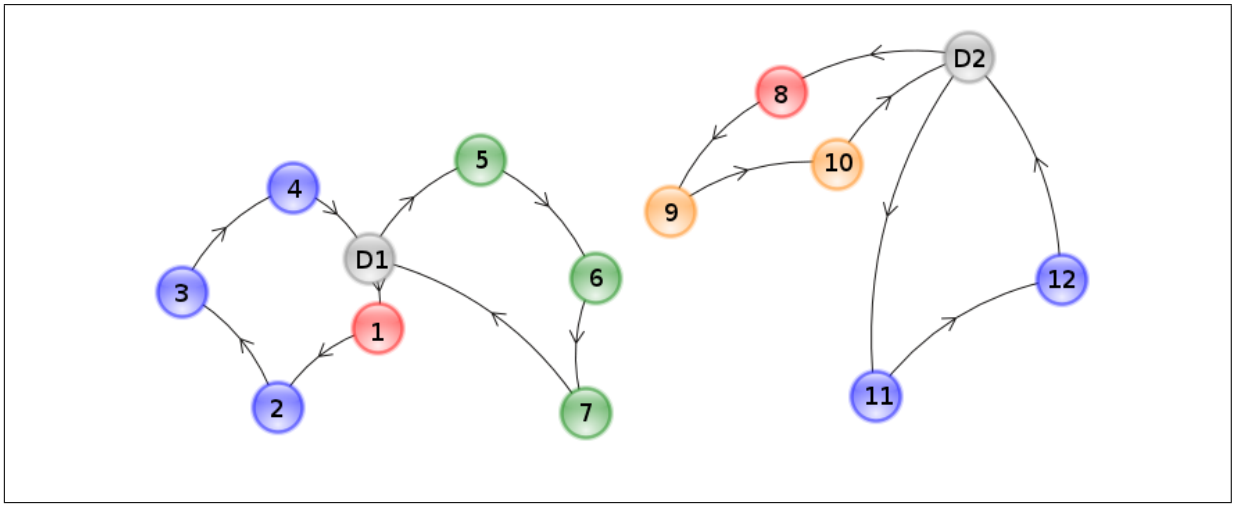

FIGURA 2.4: Exemplo de uma solução para um PDOS hipotético. Neste exemplo nota-se que existem duas bases geograficamente distribuídas (representadas por D1 e D2) de onde partem as equipes . Cada ponto representa uma ordem de serviço. Cada cor representa um tipo de ordem de serviço.

2.3.1 Modelo matemático

Neste trabalho, o problema de despacho de ordens de serviço é modelado como um PRVC com restrição de tempo de rota. As restrições do problema original não são consideradas neste modelo como, as características de multi-depósito (multi-equipes). Neste modelo a capacidade Q do veículo representa a disponibilidade em realizar um atendimento e a variável D é o limitante de tempo de rota que, neste caso, pode ser visualizado como a carga-horária máxima de

trabalho de uma equipe. Ainda, considera-se que todas as ordens necessariamente devam ser atendidas pela equipe.

$$Min \sum_{k \in K} \sum_{i,j \in E} c_{ij} x_{ijk} \tag{2.21}$$

Sujeito a:

$$\sum_{k \in K} \sum_{j \in N} x_{ijk} = 1, \forall i \in C \tag{2.22}$$

$$\sum_{i \in C} d_i \sum_{j \in N} x_{ijk} \leq Q, \forall k \in K \tag{2.23}$$

$$\sum_{i \in N} \sum_{j \in N} t_{ij} x_{ijk} \leq D, \forall k \in K \tag{2.24}$$

$$\sum_{j \in N} x_{0jk} = 1, \forall k \in K \tag{2.25}$$

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0, \forall h \in C, \forall k \in K \tag{2.26}$$

$$\sum_{i \in N} x_{i,n+1,k} = 1, \forall k \in K \tag{2.27}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1, S \subseteq C, 2 \leq |S| \leq \frac{n}{2}, \forall k \in K \tag{2.28}$$

$$x_{ijk} \in \{0,1\} \tag{2.29}$$

Onde:

- $x_{ijk} \equiv$ variável binária que assume valor 1 quando o veículo k visita o cliente j imediatamente após o cliente i, 0 caso contrário.
- $t_{ij} \equiv$ corresponde a um tempo de viagem de um arco i para j.
- $d_i \equiv$ corresponde a demanda do cliente i.

- $Q \equiv$ corresponde a capacidade do veiculo k.
- $D \equiv$ corresponde a duração máxima de cada rota.
- $c_{ij} \equiv$ corresponde ao custo de percorrer o trecho que vai do cliente i ao cliente j.

A função objetivo 2.21 minimiza a distância total da solução, calculando a soma das distâncias entre pontos de demanda presentes na solução. A restrição 2.22 garante que cada ordem seja atribuído a uma rota. Restrição 2.23 garante que a capacidate de atendimento da equipe não seja extrapolado. A restrição 2.24 garante que o tempo de rota não seja superior ao limitante D e, para isto, cada trecho de rota tem um tempo estimado representado por t_{ij}. A restrição 2.25 garante que cada rota inicie no depósito. Restrição 2.26 é uma restrição de fluxo que garante a atomicidade do atendimento. A restrição 2.27 garante que haja apenas uma aresta (ligação) entre a ordem corrente as próximas ordens. A restrição 2.28 é a restrição de clássica de eliminação de *subtours*.

2.4 Métodos de resolução para o problema de roteamento de veículos

2.4.1 Métodos exatos

As abordagens exatas para a resolução do PRV propõe uma forma de enumerar as soluções viáveis a fim de garantir que a melhor solução seja encontrada. A enumeração total geralmente é dispendiosa e impraticável em termos computacionais, o que remete ao estudo de métodos exatos com um certo grau de inteligência para evitar a busca de soluções em regiões do espaço de soluções onde a qualidade não é atrativa. Para isto, utilizam-se técnicas como: *branch and bound* e *branch and cut*.

Segundo Laporte e Nobert (LAPORTE; NOBERT, 1987), os métodos exatos podem ser classificados em: métodos de busca direta em árvore, programação dinâmica e programação linear inteira.

Até o final da década de 80, os principais algoritmos exatos baseavam-se em *branch and bound* os quais fundamentam-se na definição de limites inferior e superior no espaço de busca. Para tanto, utilizam-se limites como: *Assignment Problem*, *Degree-constrained* e *Shortest Spanning Tree*.

Como trabalhos relacionados aos métodos de busca direta em árvore e *branch and bound*, destacam-se os trabalhos de Fisher (FISHER, 1994), o qual descreve o método *k-tree*. Este

método é um dos mais atrativos na resolução exata de problemas de roteamento de veículos, tendo sucesso em resolver instâncias com até 71 pontos de demanda. Segundo Renauld e Boctor (RENAULD; BOCTOR, 2002), raramente algoritmos exatos conseguem encontrar a solução ótima para instâncias com tamanho superior a 50 pontos de demanda em um tempo aceitável.

Recentemente, novos limites mais sofisticados foram propostos para o algoritmo *branch-and-bound*, os quais aumentam o poder de resolução dos problemas combinatórios. Destacam-se os limites propostos por Fischetti et al. (FISCHETTI; TOTH; VIGO, 1994) com *additive bounding* e os trabalhos de Fisher (FISHER, 1994) e Miller (MILLER, 1995) sobre restrições lagrangeanas.

Algoritmos *branch-and-cut* são extensões dos clássicos algoritmos *branch-and-bound* onde são aplicados planos de corte no espaço de busca (GOLDBARG; LUNA, 2005). Tais planos de corte são, em geral, definidos por heurísticas. A aplicabilidade destes algoritmos ao PRV ainda é limitada, mas um grande esforço tem sido dedicado nas últimas décadas no desenvolvimento de planos de corte eficientes (TOTH; VIGO, 2002). Os resultados mais relevantes da aplicação de *branch-and-cut* ao PRVC podem ser econtrado nos trabalhos de Augerat et al. (AUGERAT et al., 1995) e Blasum e Hochstättler (BLASUM; HOCHSTÄTTLER, 2000).

2.4.2 Método heurísticos

Decorrente da dificuldade computacional de resolver problemas tipicamente combinatórios de forma exata, como o PRV, surgiu algoritmos heurísticos com a finalidade de prover soluções com uma boa qualidade em um tempo hábil.

Métodos heurísticos são em geral algoritmos específicos para um determinado problema e geralmente não se aplicam a problemas diferentes, pois tiram proveito de características inerentes a cada problema.

Cada método heurístico provê uma alternativa restrita de busca por soluções no espaço de soluções. Por este motivo, os métodos heurísticos não possuem garantia de otimalidade das soluções encontradas nem uma forma de garantir o quão próximos podem estar da solução ótima.

Um dos problemas encontrados na busca de soluções utilizando métodos heurísticos é os mínimos ou máximos locais. Mínimos ou máximos locais são regiões do espaço de busca definidos em vales ou picos, ou seja, regiões onde o custo da função objetivo é o maior (maximização) ou o menor (minimização) dentre as soluções possíveis na vizinhança analisada.

Segundo Toth e Vigo (TOTH; VIGO, 2002), as heurísticas para o PRV podem ser classificadas em três classes:

- heurísticas construtivas.
- heurísticas de duas fases.
- heurísticas de melhoramentos.

As heurísticas construtivas são métodos algorítmicos que geram soluções de forma gradual visando a melhoria da função objetivo. Como relatado por Toth e Vigo (TOTH; VIGO, 2002) tais métodos caracterizam-se por realizar uma limitada exploração no espaço de busca e por geralmente produzirem boas soluções dentro de um custo computacional baixo. Como exemplos de heurísticas podem-se citar: a heurística do vizinho mais próximo (LAWLER et al., 1985), a heurística da inserção mais barata (ROSENKRANTZ; STEARNS; LEWIS, 1977), heurística das economias (CLARKE; WRIGHT, 1964), entre outras.

Nas heurísticas de duas fases o problema é decomposto em duas etapas: o agrupamento (*clustering*) dos vértices e o roteamento (*routing*). As heurísticas de duas fases podem ainda ser divididas em duas possíveis abordagens: *cluster-first, route-second* ou *route-first, cluster-second*. Na primeira abordagem, os pontos de demanda são agrupados e, sobre estes grupos, são construídas as rotas válidas. Na segunda abordagem, cria-se primeiramente uma grande rota que envolve todos os pontos de demanda, sem considerar algumas das restrição do problema, e, na segunda etapa, esta grande rota é decomposta em rotas menores que tornem a solução factível. Esta abordagem foi proposta por Beasley (BEASLEY, 1983) o qual observou que a segunda fase desta abordagem é um caso específico do problema de caminho mais curto e, portanto, pode ser resolvido com complexidade $O(n^2)$ se for resolvido, por exemplo, com o algoritmo de Dijkstra. O desempenho destas abordagens está estritamente relacionado com a escolha dos métodos utilizados nas etapas de agrupamento e roteamento.

Por fim, heurísticas de melhoramento são procedimentos que visam aumentar a qualidade de uma solução já construída. Para tanto, utilizam-se trocas simples de vértices ou arcos dentro de uma mesma rota ou entre rotas. As heurísticas de melhoramento possivelmente finalizam a busca por soluções em um mínimo ou máximo local, dificilmente sendo em um máximo ou mínimo global.

Na literatura, encontram-se uma infinidade de heurísticas desenvolvidas ao longo das últimas décadas aplicadas ao PRV. Destaca-se a clássica heurística proposta por Clarke e Wright (CLARKE; WRIGHT, 1964) denominada *savings heuristic* (heurística das economias) . Este

método é baseado na expansão gradativa das rotas e no conceito de economia. A heurística da inserção sequencial de Mole e Jameson (MOLE; JAMESON, 1976) explora e expande os conceitos introduzidos por Clarke e Wright (CLARKE; WRIGHT, 1964).

O algoritmo de duas fases de Gillet e Miller (GILLET; MILLER, 1974) é denominado *Sweep Algorithm* (algoritmo de varredura). Este método, em sua primeira etapa, visa agrupar os pontos de demanda segundo um critério angular e, na segunda etapa, roteá-los. Outro importante algoritmo é o *Petal Algorithm* desenvolvido por Ryan et al. (RYAN; HJORRING; GLOVER, 1993). Este método é uma extensão dos conceitos empregados no algoritmo de Gillet e Miller.

Como importante contribuição acerca de heurísticas de melhoramento, cita-se o trabalho de Lin (LIN, 1965), que propõe um dos algoritmos mais eficientes na busca por soluções vizinhas baseado em trocas de arcos, o $k-opt$. Tal método foi desenvolvido inicialmente para o problema do caixeiro viajante mas os conceitos são aplicados naturalmente aos problemas de roteamento de veículos. O método $\lambda - interchange$ desenvolvido por Osman e Christofides em 1994 (OSMAN; CHRISTOFIDES, 1994) para o problema de agrupamento capacitado e, posteriormente, utilizado no PRV como um método heurístico de melhoramento inter-rota. Este método se baseia na troca de pontos de demanda entre conjuntos de rotas (grupos).

Para avaliar os movimentos gerados pelas heurísticas de melhoramento, consideram-se duas abordagens: *First Improvement* e *Best Improvement*. A primeira baseia-se na avaliação da vizinhança até que encontre uma solução com custo mais atrativo que a solução incumbente (melhor solução encontrada até o momento); a outra se baseia na avaliação de todas as possíveis soluções vizinhas. A escolha de qual abordagem a utilizar tem impacto direto no desempenho dos algoritmos de resolução.

2.4.3 Métodos meta-heurísticos

Meta-heurísticas são procedimentos genéricos aplicados geralmente a problemas de otimização. Estes procedimentos definem metodologias de busca que procuram conciliar melhoras locais com estratégias capazes de evitar a permanência das soluções em ótimos locais (TOTH; VIGO, 2002). Diferentemente das heurísticas convencionais, os procedimentos meta-heurísticos podem fazer uso combinado de escolhas aleatórias, uso de memória e conhecimento adquirido. Estas características combinadas guiam o procedimento de busca para regiões mais atrativas do espaço de soluções. A qualidade das soluções produzidas por estas metodologias são muito maiores do que as obtidas pelas simples heurísticas. Apesar de possuírem qualidade superior, tais procedimentos são computacionalmente mais caros, isto é, requerem um tempo maior de

execução.

De acordo com Glover e Kochenberger (GLOVER; KOCHENBERGER, 2003), as meta-heurísticas são processos iterativos que geram uma única solução ou uma população de soluções.

Ao longo dos últimos anos, inúmeras meta-heurísticas surgiram, entre elas é possível citar: *Simulated Annealing* (S. Kirkpatrick, 1983), *Tabu Search* (GLOVER; LAGUNA, 1997), GRASP (RESENDE, 1995), *Ant Colony* (DORIGO; GAMBARDELLA, 1997), VNS, *Genetic Algorithm* (GOLDBERG, 1989), entre outras.

Um dos resultados mais promissores obtidos para o PRVC foi obtido no trabalho de Taillard (TAILLARD, 2003) com a meta-heurística *Tabu Search*.

3 *Algoritmos desenvolvidos para o PRVC*

Para o embasamento teórico-prático deste trabalho e para criar os subsídios necessários ao entendimento dos métodos de resolução do PRV foram selecionadas algumas heurísticas da literatura que possuem características relevantes e aplicáveis ao PDOS para serem estudadas e reproduzidas algoritmicamente. O PDOS é um problema que contempla um cenário de aplicação onde é necessário o roteamento e o agrupamento de ordens já que estas devem ser roteadas a partir de um conjunto de equipes dispostas em bases estrategicamente distribuídas.

Com o intuito de evitar uma visão restrita do problema de despacho e tratá-lo tão somente como um problema de roteamento ou de agrupamento, analisa-se além de heurísticas clássicas para o roteamento, heurísticas que primem também pelos conceitos de agrupamento. Neste sentido, primeiramente analisa-se a clássica heurística das economias, proposta por Clarke e Wright (CLARKE; WRIGHT, 1964), que explora com simplicidade os aspectos de roteamento sem, no entanto, perder a robustez necessária à resolução do problema. Para contemplar o estudo de uma abordagem que prime pelos aspectos de agrupamento, estuda-se a heurística clássica de Gillet e Miller (GILLET; MILLER, 1974), que explora os conceitos de agrupamento e roteamento em etapas distintas de um mesmo procedimento.

Por fim, estuda-se os principais métodos de busca local presentes na literatura para o PRVC que exploram movimentos de melhoria intra-rotas e inter-rotas. Estuda-se o algoritmo $k-opt$ por ser um método eficiente de melhoria intra-rota e o algoritmo $\lambda-interchange$ como procedimento de melhoria inter-rota. Tais procedimentos são responsáveis pela melhoria das soluções construídas pelos métodos construtivos, como os propostos por Clarke e Wright (CLARKE; WRIGHT, 1964) e Gillet e Miller (GILLET; MILLER, 1974).

Na segunda sessão deste capítulo, apresentam-se as contribuições científicas deste trabalho: o desenvolvimento de uma heurística de busca local denominada *JoinRoutes* que explora alguns conceitos e formas de movimentações ainda não explorados; a descrição da implementação da meta-heurística $D-Ants$ (REIMANN; DOERNER; HARTL, 2004) com algumas alterações realizadas neste trabalho.

3.1 Algoritmos clássicos desenvolvidos para o PRVC

3.1.1 Algoritmo das economias

O algoritmo das economias ou *Savings Algorithm* é uma das mais tradicionais heurísticas construtivas para o PRV. O algoritmo foi proposto em 1964 por Clarke e Wright em (CLARKE; WRIGHT, 1964) e baseia-se na expansão gradativa das rotas de modo a minimizar a função objetivo, a partir de simples reorganizações sobre as rotas já construídas.

O algoritmo parte de uma solução trivial, onde todos os pontos de demanda são atendidos por rotas únicas. A partir desta solução são calculadas as economias. Estas economias representam o impacto da inclusão de um ponto a uma outra rota, isto é, a atratividade da união de pontos de demanda sob a mesma rota.

Dados duas ligações do tipo $(i-d-i)$ e $(j-d-j)$, onde i e j representam pontos de demanda e d o depósito, remove-se duas ligações e acrescenta-se uma outra entre os pontos de demanda. O cálculo quantitativo destas mudanças é realizado da seguinte forma:

$$economia = c_{di} + c_{jd} - c_{ij} \tag{3.1}$$

Onde c_{di} representa o custo de deslocamento do depósito d ao nó i , c_{jd} representa o custo de deslocamento do nó j ao depósito d e c_{ij} o custo de ir do nó i ao nó j.

O procedimento de calcular as economias pode ser observado na ilustração abaixo.

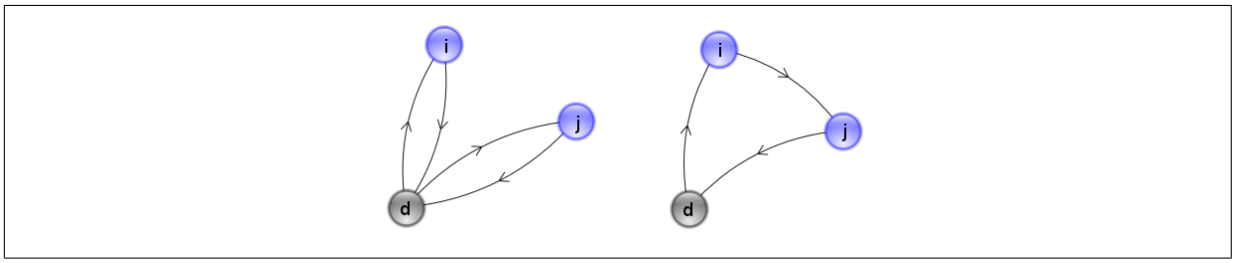

FIGURA 3.1: Exemplo do procedimento de economia. À esquerda, antes da aplicação do procedimento de economia. À direita, após a aplicação do procedimento de economia.

O algoritmo 1 apresenta os detalhes da implementação do método. Tal algoritmo segue as definições propostas por Goldbarg e Luna (GOLDBARG; LUNA, 2005).

ALGORITMO 1 Algoritmo das economias

Require: $instancia = G(V,A)$

1: $sol = InicializarRota()$ { Gerar rotas triviais do tipo (PontoDemanda1-Depósito-PontoDemanda1), onde cada rota atende apenas um ponto de demanda}
2: $ListlistaEconomia = CalcularEconomias(sol)$ { O cálcula das economias é dado segundo a equação 3.1}
3: $OrdenarDec(listaEconomia)$ {Ordem decrescente}
4: **while** $Existirconomias$ **do**
5: Determinar primeira economia da lista que pode ser utilizada para ser acrescida em um dos dois extremos da Rota
6: Remova da lista esta economia
7: Se Rota não pode ser expandida então escolher a primeira ligação viável da lista
8: **end while**

3.1.2 Algoritmo de varredura

O algoritmo de varredura ou *Sweep Algorithm* foi proposto por Gillet e Miller (GILLET; MILLER, 1974). Esta heurística é uma das tantas existentes desenvolvidas que se baseiam no conceito de heurísticas de duas fases conforme a abordagem *cluster-first*, *route-second*, ou primeiro agrupar e depois rotear.

O método baseia-se na premissa de que as rotas são desenvolvidas preferencialmente entre os pontos de demanda mais próximos (vizinhos) e destaca-se por apresentar uma enorme eficiência e simplicidade na fase de agrupamento (GOLDBARG; LUNA, 2005).

Inicialmente calcula-se as coordenadas polares para todos os pontos de demanda. As coordenadas polares são calculadas em relação ao depósito. Ao final deste procedimento, os pontos de demanda (nós) são ordenados em ordem crescente de acordo com os valores angulares de suas coordenadas. Seleciona-se o ponto de demanda com menor angulagem e iterativamente os demais são interconectados até que alguma das restrições seja excedida. Quando uma restrição excede, o procedimento é reiniciado e novamente os pontos de demanda restantes são ligados ao depósito. Desta forma os pontos são agrupados segundo o critério angular. Como ocorre a conexão dos pontos de demanda a cada iteração, ao final do procedimento tem-se uma solução inicial.

Uma possível abordagem é não conectar os pontos de demanda durante a etapa de *clustering*. Neste caso, a segunda etapa do algoritmo deverá necessariamente ser a aplicação de um método que gere soluções sobre o grupo de pontos. Existem ainda abordagens que utilizam na segunda etapa, etapa de *routing*, algoritmo exatos.

Após ter construído a solução inicial para o problema, aplica-se procedimentos de busca

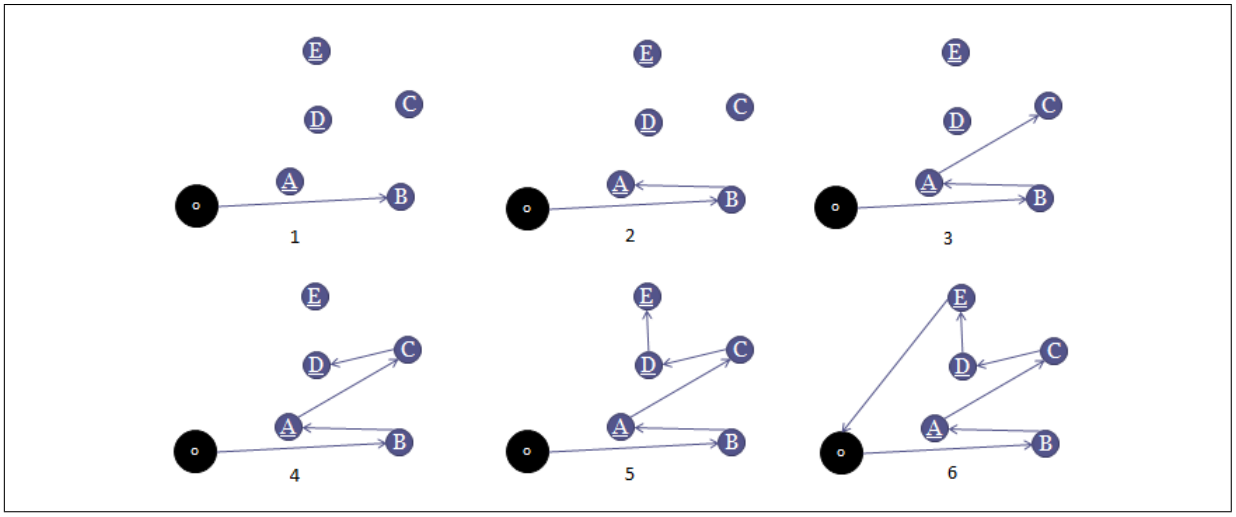

FIGURA 3.2: Exemplo do funcionamento da etapa construtiva da heurística de varredura.

local no intuito de intensificar e refinar a qualidade da solução gerada. Nesta implementação, utiliza-se o procedimento $k-opt$, utilizando valor de k igual a 2.

A representação do procedimento pode ser observado no algoritmo 2. Tal representação é adaptada de Goldabarg e Luna (GOLDBARG; LUNA, 2005).

ALGORITMO 2 Algoritmo de varredura

Ler Grafo $G = (N,A), c_{i,j}; n1 o de psito central do roteamento$
Obter as coordenadas polares dos clientes em relação ao depósito;
Ordená-las em ordem crescente do seu valor angular;
$F = N\ x_1$;
$Ponta_Rota = x_1$;
$i = 1$;
while $F \neq \emptyset$ **do**
  $Rota_i = \{x_1\}$;
  while $\exists x_s \in F$ atendendo as condições de viabilidade para: $Rota_i$ **do**
    Encontrar o vértice $x_s \in F$ com menor coord. polar;
    $Rota_i = Rota_i \cup x_s$;
    $Ponta_Rota = x_s$;
    $F = F\ x_s$;
  end while
  $Rota_i = Rota_i \cup \{x_1\}$;
  Aplicar procedimento de busca local em $Rota_i$;
  $i = i+1$;
  $PontaRota = x_1$;
end while

Algoritmo de Varredura com Variação do Ponto Inical

A implementação acima descrita representa uma maneira simples e determinística de implementar os conceitos de *clustering* desenvolvidos por Gillet e Miller (GILLET; MILLER, 1974). O algoritmo de varredura original possui algumas variações em sua implementação.

Gillet e Miller (GILLET; MILLER, 1974) expõem em seu trabalho que uma maneira de implementar os conceitos de *cluster* desenvolvidos em seu trabalho é utilizando a variação do ponto inicial (*seed*). O ponto inicial da construção do *cluster* refere-se ao ponto com uma certa angulagem em relação ao depósito. Na implementação proposta por Goldbarg e Luna (GOLDBARG; LUNA, 2005), o ponto inical é sempre aquele com o menor valor angular. Na implementação original, Gillet e Miller (GILLET; MILLER, 1974) propõe que o algoritmo seja executado por um número de vezes ou por um tempo determinado, onde a cada iteração o ponto inicial ou *seed* seja selecionado de forma aleatória. Ao final do processo, retorna-se a melhor solução obtida ao longo das iterações.

Desta forma, evita-se que a formação dos *clusters* seja determinística.

3.1.3 Algoritmo $k-opt$

A heurística $k-opt$ foi proposta por Lin em 1965 (LIN, 1965) e foi desenvolvida inicialmente para o problema do caixeiro viajante (PCV) mas pode ser aplicada naturalmente ao PRV.

Existem inúmeras implementações desta heurística, como em (LIN; KERNIGHAN, 1973). Lin e Kernighan descrevem uma forma efetiva de implementar tal método aplicado ao PCV. Tal método é um dos clássicos algoritmos de busca local para o PCV e baseia-se na troca de arcos em soluções inicialmente criadas (através de heurísticas construtivas), onde o número de arcos a serem trocados depende de k . À medida que o parâmetro k aumenta, o procedimento aproxima-se da enumeração total de soluções vizinhas (GOLDBARG; LUNA, 2005).

O procedimento pode ser descrito da seguinte forma: remover k arcos e reinserí-los de todas as maneiras possíveis. Destas reinserções permanece a que possuir o melhor custo associado à função objetivo. Caso não possua custo melhor que o original antes da remoção, escolhe-se outros k arcos para serem removidos. O procedimento é reiniciado até que todas as possibilidades de remoções dos k arcos sejam realizadas. Segue na figura 3.3 um exemplo de uma etapa do funcionamento do algoritmo k-opt.

A abordagem de busca e escolha de soluções vizinhas utilizada foi a *Best Improvement* (BI). Nesta abordagem a vizinhança de soluções é totalmente explorada a partir dos movimentos

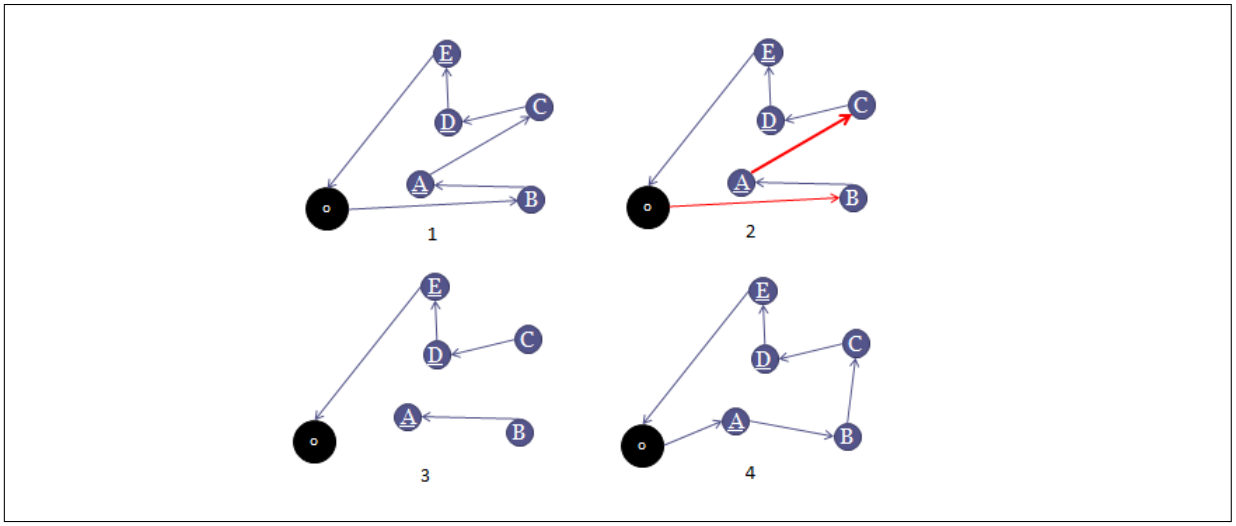

FIGURA 3.3: Exemplo do funcionamento da heurística de melhoramento $k-opt$ com $k=2$.

de remoção/inserção de arestas e é escolhido o melhor de todos os possíveis movimentos de melhora para ser realizado. Concluída a alteração na solução, o procedimento é reiniciado onde são novamente exploradas as soluções vizinhas a partir da nova solução incumbente. Como condição de parada do algoritmo utiliza-se a primeira iteração onde não há melhoria de custo da solução.

No algoritmo 3 pode ser visualizado os detalhes da implementação do procedimento $2-opt$.

ALGORITMO 3 Algoritmo k-opt

Require: Grafo $sol \leftarrow G(N,A)$; {nó 1 é o depósito central do roteamento}

Require: Matriz $custo \leftarrow c_{i,j}$; {Matriz de custos}

1: $melhorTroca \leftarrow 0$
2: $iMelhor, jMelhor$
3: $houverMelhora \leftarrow true$
4: **for all** $Rotas \in ConjuntoRotas$ **do**
5: $houverMelhora \leftarrow false$
6: **while** $houverMelhora$ **do**
7: **for all** $i \in N$ **do**
8: **for all** $j \in N$ **do**
9: **if** $Dist(anterior(i), j) + Dist(i, proximo(j)) < Dist(anterior(i), i) + Dist(j, proximo(j))$ **then**
10: **if** $Dist(anterior(i), j) + Dist(i, proximo(j)) > melhorTroca$ **then**
11: $melhorTroca = Dist(anterior(i), j) + Dist(i, proximo(j))$
12: $iMelhor \leftarrow i$
13: $jMelhor \leftarrow j$
14: **end if**
15: **end if**
16: **end for**
17: **end for**
18: **if** $melhorTroca \neq \emptyset$ **then**
19: InserirAresta(anterior(iMelhor), jMelhor);
20: InserirAresta(iMelhor, proximo(jMelhor));
21: RemoverAresta(anterior(iMelhor), iMelhor);
22: RemoverAresta(jMelhor, proximo(jMelhor));
23: $melhorTroca \leftarrow 0$
24: $houverMelhora \leftarrow true$
25: **end if**
26: **end while**
27: **end for**

3.1.4 Algoritmo $\lambda - interchange$

O método de busca local $\lambda - interchange$ foi introduzido por Osman e Christofides em 1994 (OSMAN; CHRISTOFIDES, 1994) para o problema de agrupamento capacitado. Posteriormente foi utilizado no PRV como um método heurístico de melhoramento inter-rota. Este método se baseia na troca de pontos de demanda entre conjuntos de rotas (grupos).

Dado uma solução para o PRV representada por um conjunto de rotas $S = \{R_1, R_2, R_3, \ldots, R_n\}$, onde cada rota contem um conjunto de clientes/pontos de demanda, a aplicação do método $\lambda - interchange$ entre um par de rotas $R_p e R_q$ é definido como a troca de um conjunto de $clientes S_1 \subseteq R_p$ de tamanho máximo igual a $|S_1| \leq \lambda$ por um outro subconjunto $S_2 \subseteq R_q$ de tamanho máximo igual a $|S_2| \leq \lambda$ a fim de obter duas novas rotas: $(R'_p - S_1) \cup S_2 e (R'_q - S_2) \cup S_1$. A vizinhança $N_\lambda(S)$ é dada pelo conjunto de todas as permutações entre as rotas e um dado valor de λ. Segue, na figura 3.4, uma representação de uma etapa de troca de pontos de demanda entre duas rotas.

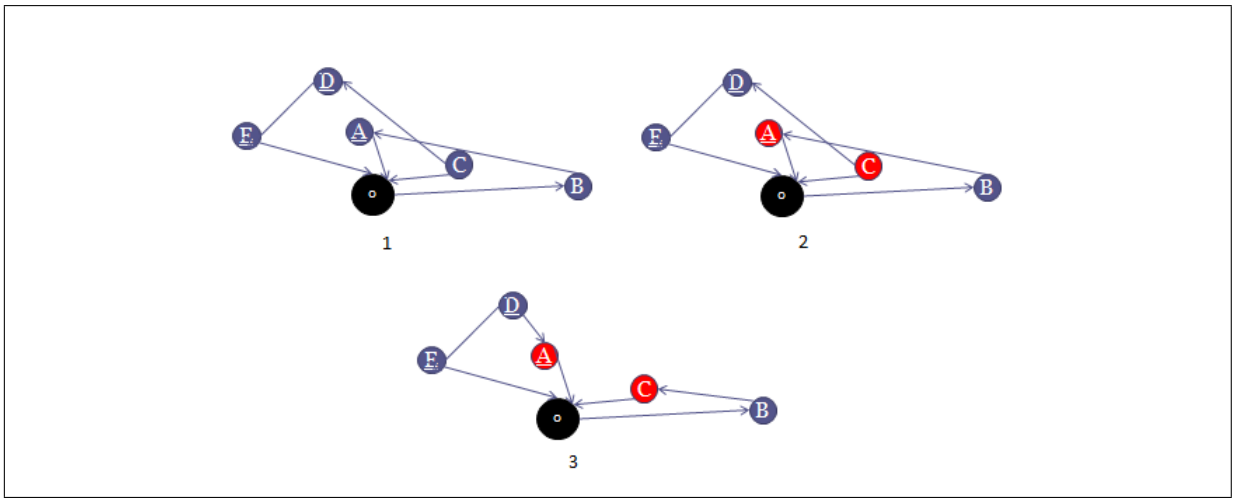

FIGURA 3.4: Exemplo do funcionamento da heurística de melhoramento $\lambda - interchange$. Neste caso, $|\lambda| = 1$.

A abordagem de busca na vizinhança utilizada é a *Best Improvement*. Neste caso, todas as permutas de pontos de demanda entre os conjuntos de rotas analisadas são avaliadas e desses possíveis movimentos o melhor é realizado. Como critério de parada do algoritmo utiliza-se a primeira iteração sem melhoria.

No algoritmo 4 pode ser visualizado os detalhes da implementação do procedimento.

ALGORITMO 4 Algoritmo $\lambda - interchange$

Require: Grafo S, λ; {criada pela heurística construtiva}

1: $melhorCusto \leftarrow Custo(S)$;
2: $houverMelhora \leftarrow true$;
3: **while** $houverMelhora$ **do**
4: $houverMelhora \leftarrow false$
5: **for all** par de rotas $R_i, R_j \in S$ **do**
6: Definir todos os subconjuntos $S_i \subseteq R_i, S_j \subseteq R_j$, sendo $|S_i| \leq \lambda, |S_j| \leq \lambda$
7: $S' = S - R_i - R_j$;
8: $R'_i = (R_i - S_i) \cup S_j$;
9: $R'_j = (R_j - S_j) \cup S_i$;
10: $S' \cup R'_i \cup R'_j$;
11: **if** $Custo(S') < melhorCusto$ **then**
12: $S'' \leftarrow S'$;
13: $melhorCusto \leftarrow Custo(S')$;
14: $houverMelhora \leftarrow true$;
15: **end if**
16: **end for**
17: **if** $houverMelhora$ **then**
18: $S \leftarrow S''$;
19: **end if**
20: **end while**

3.2 Algoritmos Propostos

Esta sessão descreve as principais contribuições científicas deste trabalho de pesquisa. A partir dos métodos de resolução analisados e reproduzidos na sessão anterior foi possível agregar conhecimento teórico necessário para propor uma heurística de busca local. Tal procedimento contempla algumas características ainda não exploradas na literatura e é aplicável a qualquer variante do PRV.

Além disto, tal estudo resultou na análise e implementação do procedimento meta-heurístico $D - Ants$, proposto por Reimann et al. (REIMANN; DOERNER; HARTL, 2004). Tal método contempla uma abordagem onde são utilizadas ambas as heurísticas construtivas estudadas na sessão anterior (*Savings Algorithm* e *Sweep Algorithm*). Tal procedimento prima por soluções que contemplem características de agrupamento e roteamento e para isto utiliza sistematica-

mente as abordagens estudadas em conjunto. Como tal método apresenta robustez nos resultados alcançados quando aplicado a instâncias de tamanho real (REIMANN; DOERNER; HARTL, 2004), torna-se atrativo a análise de aplicabilidade ao PDOS pois este também apresenta instâncias de grande tamanho.

Nas próximas subseções são apresentadas os detalhes de implementações dos algoritmos $D-Ants$ e $JoinRoutes$.

3.2.1 Algortimo D-Ants

O algoritmo D-Ants é um procedimento meta-heurístico inspirado em *Ant Colony*. Foi proposto em 2004 por Reimann et al. (REIMANN; DOERNER; HARTL, 2004) e é um procedimento eficiente e efetivo para resolução de problemas de roteamento com instâncias de tamanho real.

O D-Ants é um melhoramento da meta-heurística *Saving Based Ant System* (SbAS) proposta anteriormente também por Reimann et al. (REIMANN; DOERNER; HARTL, 2004). O procedimento SbAS é um procedimento que utiliza a meta-heurística *Ant Colony* em conjunto com o algoritmo das economias. Já o D-Ants acrescenta uma etapa ao procedimento SbAS a qual é inspirada no procedimento heurístico de varredura (*Sweep Algorithm*). Tal etapa tem o objetivo de resolver partes do problema por meio de uma estratégia de divisão e conquista.

A meta-heurística *Ant Colony* foi proposta por Dorigo (DORIGO; GAMBARDELLA, 1997) e é um procedimento bio-inspirado no comportamento social das formigas. Mais precisamente, quando as formigas saem à procura de alimento, cada formiga deixa em seu percurso uma quantidade de feromônio. O feromônio é uma substância química produzida por certos animais que permitem o reconhecimento mútuo e sexual de indivíduos de uma mesma espécie. Quanto mais formigas utilizarem o mesmo caminho, maior será a concentração de feromônio e consequentemente mais formigas serão atraídas a utilizar este caminho.

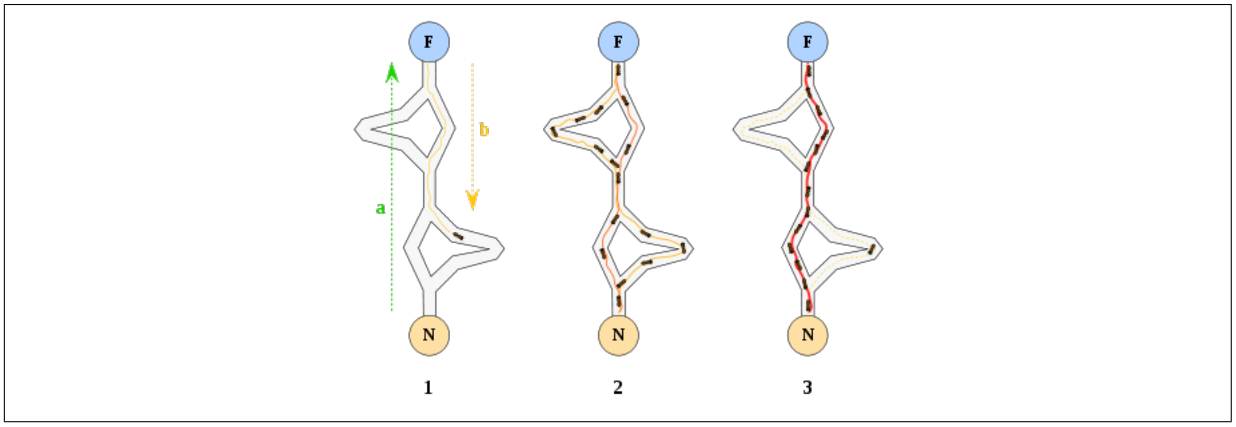

FIGURA 3.5: Princípio do funcionamento da meta-heurística Ant Colony. 1 - A primeira formiga procura a comida, representada por F, através de qualquer caminho. Retorna ao formigueiro deixando uma certa quantia de feromônio. 2 - Formigas seguem indiscriminadamente os 4 possíveis caminhos e aos poucos o melhor caminho se torna atraente pela alta quantia de feromônio. 3 - A grande maioria das formigas seguem a melhor rota até a comida. **Fonte**: Wikipédia

Analogamente ao processo biológico, os sistemas computacionais simulam o comportamento das formigas com agentes artificias induzindo um processo correlato ao mundo natural a fim de solucionar inúmeros problemas de otimização. Neste caso, utiliza-se o *Ant Colony* para prover soluções para o problema de roteamento de veículos. Basicamente, o procedimento consiste na iteração de três etapas:

- Geração de soluções pelas formigas artificias utilizando a concentração de feromônio.
- Aplicação de procedimentos de busca local
- Atualização do feromônio.

O algoritmo D-Ants inclui uma etapa adicional no algoritmo SbAS chamada de *Augmentation Saving List*. Esta etapa compreende um conjunto de procedimentos que visam aumentar a qualidade através da decomposição da solução.

O critério de parada do procedimento pode ser definido através de um número estático de iteração, tempo máximo de execução, número de iterações sem melhora, entre outras. Nesta implementação, utiliza-se um tempo limite de execução de 600 segundos ou a primeira iteração sem melhora.

ALGORITMO 5 Algoritmo genérico para a meta-heurística Ant Colony

```
1: sol ← LerInstancia();
2: while ¬CondicaoParada do
3:    sol ← gerarSolucao(); {Gera solucoes segundo alguma heuristica}
4:    sol ← aplicarBuscaLoca(); {Aplica-se os procedimentos de busca local}
5:    atualizarFeromonio();
6:    if Custo(sol) < Custo(melhorSol) then
7:       melhorSol ← sol; {Atualiza-se a solucao incumbente}
8:    end if
9: end while
```

Saving based Ant System (SbAS)

Antes de apresentar o algoritmo D-Ants, faz-se necessário uma breve descrição sobre o funcionamento do procedimento SbAS, já que este é utilizado pelo D-Ants. O SbAS é uma variação do algoritmo tradicional Ant Colony (DORIGO; GAMBARDELLA, 1997) e utiliza em sua etapa construtiva uma variação do algoritmo das economias (CLARKE; WRIGHT, 1964). Para isto, serão descritas as três etapas do algoritmo: geração da solução, aplicação de procedimentos de busca local e atualização do feromônio, análogas ao Ant Colony.

1 - Geração da solução

Os mecanismos de geração de solução utilizados no Ant Colony são baseados na heurística do vizinho mais próximo (BULLNHEIMER; HARTL; STRAUSS, 1999). Em oposto a tradicional abordagem, utiliza-se no SbAS uma versão aleatória do algoritmo das economias (CLARKE; WRIGHT, 1964). Para isto, o algoritmo das economias é modificado no intuito de suportar a utilização da concentração de feromônio nas etapas de decisão.

Inicialmente, gera-se uma lista de atratividade em ordem decrescente para cada par de vértices. Esta lista de atratividade contém as informações das economias quanto da concentração de feromônio no arco envolvido. Assim, a lista é construída da seguinte maneira:

$$\xi_{ij} = [s_{ij}]^{\beta}[\tau_{ij}]^{\alpha} \tag{3.2}$$

Onde τ_{ij} representa a concentração de feromônio no arco conectando os pontos de demanda i e j, e α e β representam a relativa influência da quantidade de feromônio e s_{ij} o valor da economia. A concentração τ_{ij} indica o quão atrativo foi a combinação dos pontos i e j em

iterações anteriores.

Em cada etapa de decisão de uma formiga, considera-se as k melhores combinações possíveis da lista de atratividade, onde k é o parâmetro do algoritmo que se refere a vizinhança.

Dado Ω_k ser o conjunto dos k melhores vizinhos, ou seja, os k vizinhos com maior valor de atratividade, toda etapa de decisão é baseada na seguinte equação:

$$p_{ij} = \begin{cases} \frac{\xi_{ij}}{\sum_{(h,l)\in\Omega_k} \xi_{hl}} & \text{se} \quad \xi_{ij} \in \Omega_k \\ \\ 0 & \text{caso contrário} \end{cases} \tag{3.3}$$

Onde p_{ij} representa a probabilidade de escolher os arcos i e j. A etapa construtiva para quando não existirem mais combinações factíveis.

2 - Busca local

Após a construção das soluções pelas formigas artificiais, a cada uma destas soluções são aplicados procedimentos de busca local. Primeiramente, aplica-se o algoritmo *swap* (denominação dada para o algoritmo $\lambda - interchange$ quando $|\lambda| = 1$) a qual objetiva a melhora dos clusters constituídos de pontos de demanda. Tal movimento caracteriza-se pela troca do ponto de demanda $i \in k$ por $j \in l$, onde k, l são rotas. Seguido desta aplicação inicial, aplica-se o algoritmo $k - opt$ com $k = 2$. Tal algoritmo realiza iterativamente, como descrito anteriormente, trocas de arestas objetivando a melhoria intra-rota.

3 - Atualização do feromônio

A última etapa caracteriza-se pela atualização das informações referentes ao feromônio presente nas arestas que ligam os pontos de demanda. Como em um processo natural, a quantidade de feromônio evapora assim como são depositados novas quantias sobre os trajetos. A evaporação baseia-se em uma constante de persistência definida *a priori*. Para o depósito de feromônio são consideradas as melhores solução geradas pelas formigas artificiais. Para tanto, utiliza-se um modelo de *rank* no qual um número de soluções de elite, representado por n_e, são utilizadas para efetuar o cálculo de atualização. Estas operações são descritas pela equação 3.4.

$$\tau_{ij} = \rho\,\tau_{ij} + \sum_{r=1}^{n_e-1} \Delta\tau_{ij}^{r} + n_e \Delta\tau_{ij}^{*} \tag{3.4}$$

Onde $0 \leq \rho \leq 1$ representa a taxa de persistência do feromônio e n_e o número de soluções elite. Utilizando a equação 3.4, são consideradas duas maneiras de depositar o feromônio. A

primeira refere-se ao montante depositado pela melhor solução encontrada, a segunda pelas soluções de elite. Atualiza-se a melhor solução como se n_e formigas tivessem percorrido o melhor caminho encontrado. Na equação 3.4, isto é representado por $\Delta\tau_{ij}^{*} = 1/L^{*}$ onde L^{*} representa o custo da função objetivo da melhor solução. A segunda forma de depósito é realizada utilizando as $n_e - 1$ soluções restantes. Neste caso, $\Delta\tau_{ij}^{r} = (n_e - r)/L^{r}$ onde a quantidade de feromônio depositado depende diretamente do *rank r* bem como da qualidade L^{r} de cada solução.

Augmentation Saving List

Esta procedimento compreende a quarta etapa adicionada ao SbAS pelo algoritmo D-Ants. Segundo Reimann et al. (REIMANN; DOERNER; HARTL, 2004), a estratégia utilizada é baseada na premissa de que o PRV é uma generalização do problema do caixeiro viajante (PCV). Assim, realizada a atribuição de cada ponto de demanda a um veículo (clusterização), um PRV se reduz a resolver um conjuntos de TSP (um para cada rota). O princípio fundamental deste etapa é decompor a região constituída pelas rotas em conjuntos menores geograficamente distintos. Este princípio baseia-se na ideia de que as rotas, geralmente, cobrem regiões geograficamente distintas. Assim, pretende-se criar pequenos *clusters* de pontos de demanda. Cada subproblema, constituído por *clusters* de pontos, é resolvido pela aplicação do SbAS.

Inicialmente, o problema é resolvido em sua totalidade pela aplicação do SbAS. Dada a melhor solução encontrada pela aplicação do SbAS, calcula-se os centros de gravidade para cada rota pertencente a esta solução e então as rotas são agrupadas utilizando o algoritmo de varredura. Para cada *cluster* de rotas aplica-se o algoritmo SbAS para um dado número de iterações. Após todos os subproblemas terem sido resolvidos pelo SbAS, os subproblemas são remontados juntos a solução global. Por último, realiza-se a atualização do feromônio.

Abaixo segue o algoritmo em pseudo-código adaptado de Reimman et al. (REIMANN; DOERNER; HARTL, 2004). A etapa (*Augmentation Saving List*) é descrita no algoritmo 6 pelos passos II, III, IV e V.

ALGORITMO 6 D-Ants

```
1: Ler instância
2: Inicializar algoritmo {Inicializar parâmetros, matrizes de distância e feromônio}
3: while ¬CondicaoParada do
4:    for Pré-especificado número de iterações do
5:       Resolver o problema utilizando o algoritmo SbAS (Saving Based Ant System); {Etapa
         I}
6:    end for
7:    for Melhor solução encontrada até agora do
8:       Computar o centro de gravidade para cada rota; {Etapa II}
9:    end for
10:   Decompor a melhor solução em um número pré-estabelecido de subproblemas pela apli-
      cação do algoritmo de varredura; {Etapa III}
11:   for Cada subproblema do
12:      for Pré-especificado número de iterações do
13:         Resolver o subproblema utilizando o algoritmo SbAS utilizando as informações de
            feromônio globais. {Etapa IV}
14:         Se aplicável, atualizar melhor solução; {Etapa V}
15:      end for
16:   end for
17: end while
```

Reimman et al. (REIMANN; DOERNER; HARTL, 2004) argumentam que a utilização de uma abordagem baseada na decomposição reduz substancialmente o tempo de execução dos algoritmos de resolução. Explica ainda que quanto maior o problema, mais iterações serão necessária para resolvê-lo com um nível de qualidade utilizando a abordagem clássica do Ant Colony ou mesmo do método SbAS. Contudo, em sua abordagem, Reimman et al.(REIMANN; DOERNER; HARTL, 2004) propõem resolver o problema em poucas iterações para posteriormente explorar as informações de ferômonio na resolução dos subproblemas (Passo IV do algoritmo D-Ants).

Calcular centro de gravidade - Etapa II

Dada a melhor solução encontrada pelo algoritmo SbAS, computa-se o centro de gravidade

para cada rota. O cômputo dos centros de gravidade é realizado segundo a equação 3.5.

$$centroGravidade = \frac{\sum demanda_i * coord_i}{\sum demanda_i} \quad \forall i \in Rota_i, Rota_i \in S \tag{3.5}$$

Onde S representa a solução para um problema.

Decompor a solução - Etapa III

Tendo computado os centros de gravidade para cada rota, aplica-se o algoritmo de varredura sobre os centros computados. Ignoram-se os pontos de demanda pertencentes a cada rota e apenas consideram-se os centros de gravidades.

Segundo Reimman et al. (REIMANN; DOERNER; HARTL, 2004) o número de *clusters* é pré-definido pela escolha do número de subproblemas. Dado este número, calcula-se o número de rotas a serem atribuídas a cada *cluster*. Formalmente, seja n_t o número de rotas na melhor solução encontrada (Etapa I) e n_s o número de subproblemas (pré-definido), o calculo de rotas por *cluster* é realizado segundo a equação 3.6.

$$n_{s,t} = \left\lceil \frac{n_t}{n_s} \right\rceil \tag{3.6}$$

Calculado o valor da variável $n_{s,t}$, que indica o número de rotas por cluster, aplica-se o algoritmo de varredura. Primeiramente, calculam-se as coordenadas polares dos centros de gravidade (calculados na etapa II). Tais centros são ordenados em ordem crescente, em relação ao ângulo polar. Diferente da implementação da seção 3.1.2, onde seleciona-se o primeiro ponto de demanda com a menor angulagem em relação ao depósito, nesta abordagem seleciona-se um ponto pertencente as rotas de forma aleatória. A varredura então é iniciada pelo primeiro centro de gravidade com angulagem superior ao ponto selecionado. Este centro de gravidade é o nó *seed*. A partir deste centro de gravidade, os demais vão sendo unidos em conjuntos de $n_{s,t}$ rotas.

Cada *cluster* é formado pelos pontos de demanda pertencentes as rotas representadas pelos centros de gravidade. O procedimento pode ser vizualido na figura 3.6.

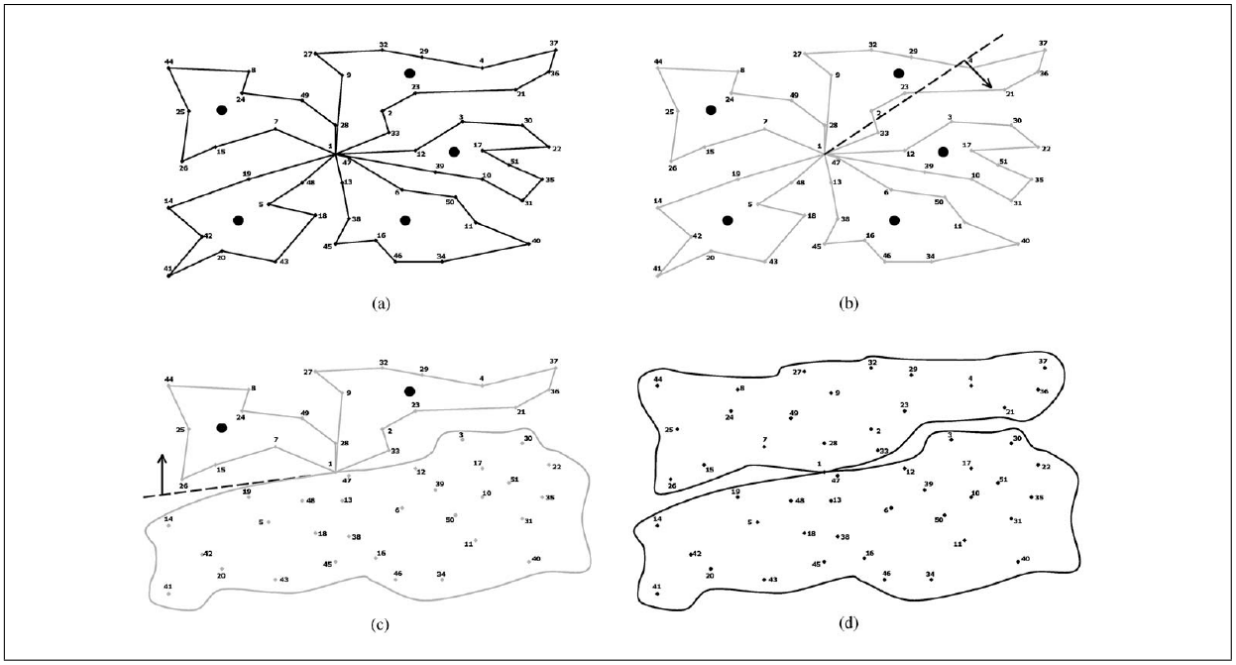

FIGURA 3.6: Exemplo da etapa de *clustering* do procedimento D-Ants. Em (a) são calculados os centros de gravidade para cada rota. Em (b) seleciona-se um ponto de demanda aleatório. Em (c) inicia-se o procedimento de *clustering* a partir primeiro centro de gravidade após o nó selecionado. Neste exemplo considera-se que $n_{s,t} = 3$. **Fonte:** Reimann et al. (REIMANN; DOERNER; HARTL, 2004)

Segundo Reimman et al. (REIMANN; DOERNER; HARTL, 2004), a abordagem utilizada de selecionar aleatoriamente o nó *seed* evita o determinismo na etapa de *clustering* e provê maior otimização.

Solução dos subproblemas - Etapa IV

A solução dos subproblemas se dá pela aplicação do algortimo SbAS ao conjunto de nós de cada *cluster*. Esta etapa utiliza as informações geradas na etapa I referentes a quantidade de feromônio presente nos arcos. Decorrente do menor número de pontos em cada *cluster*, o algoritmo SbAS pode executar em um número maior de iterações.

Atualização - Etapa V

Um dos principais objetivos das etapas II, III e IV é melhorar a qualidade das solução global por meio da melhora de partes desta solução (aplicação do algoritmo SbAS aos *clusters*). Na etapa IV, as informações de feromônio da solução global são utilizadas na resolução dos subproblemas e nesta etapa, etapa V, partes das informações do feromônio global são reforçados. Este reforço ocorre quando a resolução destes subproblemas resulta em melhoras locais, isto é, quando comparado o resultado *cluster* com o resultado da solução global. Tal reforço ocorre da mesma forma que na atualização do feromônio do SbAS e garante que haja uma comunicação

entre a solução global e as soluções dos *clusters*.

Reimann et al. (REIMANN; DOERNER; HARTL, 2004) ainda coloca que a utilização deste procedimento pode convergir para boas soluções em pouco tempo de execução.

3.2.2 Algoritmo JoinRoutes

O algoritmo JoinRoutes é um método de busca local proposto neste trabalho. Esta heurística baseia-se no princípio *route-first, cluster-second* e tem como objetivo explorar regiões de mínimos locais a fim de movimentar a solução incumbente para uma região mais atrativa do espaço de soluções. Para tanto, o método utiliza uma maneira diferenciada de explorar o espaço de soluções.

Ao contrário dos métodos tradicionais de busca local que exploram somente o espaço de soluções factíveis, este método busca explorar regiões que são atingíveis a partir de regiões infactíveis do espaço de soluções. Os movimentos que possibilitam esta busca em regiões infactíveis são baseados na relaxação da restrição de capacidade e na modificação da cardinalidade das rotas e dos elementos pertencentes a estas.

Dada uma solução inicial construída, o algoritmo une N rotas e, posteriormente, as particiona em conjuntos de no máximo N rotas com o intuito de minimizar o custo da solução. Desta forma, pretende-se modificar a cardinalidade de elementos pertencentes as rotas e a cardinalidade de rotas da solução.

Os movimentos são gerados a partir das combinações entre as possíveis junções. O tamanho máximo de rotas a unir é definido como um parâmetro do algoritmo. Realizada a construção da grande rota (composta pela união das N rotas), esta é particionada em conjuntos de 1 a N sem considerar, neste momento, a restrição de capacidade do veículo. Cada partição gerada desta grande rota é um possível movimento.

Todos os movimentos gerados são ordenados segundo uma função que avalia a redução do custo oriundo deste movimento. Em um processo iterativo entre todos estes movimentos gerados, tenta-se realizar primeiramente aqueles movimentos que possuem maior atratividade. Se o movimento selecionado é factível, então este apenas é efetivado e o processo é reiniciado. Caso este movimento tenho rotas infactíveis, então estas deverão ser viabilizadas. Para isto, utiliza-se o conceito de movimentos compostos. Dado uma rota infactível, tenta-se factibilizá-la pelo deslocamento de pontos desta para as rotas vizinhas a ela até que esta se torne viável. Caso à rota vizinha a rota anteriormente infactível tenha se tornado também infactível, o mesmo procedimento é aplicado a ela. Para evitar possíveis *loops* de movimentos compostos, estes

movimentos são limitados por uma constante que, nesta implementação, é igual ao número de rotas. Se possível for a factibilização deste movimento, avalia-se a perturbação gerada pelos movimentos compostos e se a solução ainda é atrativa. Caso for, efetiva-se o movimento da solução e reinicia-se o processo. Caso o movimento não seja mais atrativo ou não seja possível a factibilização, outro movimento é selecionado.

A abordagem de busca utilizada neste algoritmo é a *best improvement* e a condição de parada utilizada é a primeira iteração sem melhoria.

Segue abaixo, em pseudo-código, uma descrição do algoritmo JoinRoutes.

ALGORITMO 7 JoinRoutes

```
1:  Ler sol, N {Solução gerada pela heurística construtiva; N - Número máximo de rotas a
    unir}
2:  Inicializar parâmetros {Inicializar matriz de distância}
3:  while ¬CondicaoParada do
4:     for all Rota r ∈ sol do
5:        Computar o centro de gravidade desta rota;
6:     end for
7:     gerarMovimentos();
8:     avaliarMovimentos();
9:     ordenarMovimentos();
10:    for Cada movimento do
11:       if movimenot factivel then
12:          realizarMovimento();
13:          break; {Reinicia o procedimento}
14:       else
15:          Factibilização();
16:          if movimento tornou-se factivel then
17:             if movimento ainda é atrativo then
18:                realizarMovimento();
19:             end if
20:          end if
21:       end if
22:    end for
23: end while
```

Centro de Gravidade

A etapa inicial do algoritmo *JoinRoutes* é unir rotas em uma grande e única rota. No intuito de reduzir a complexidade computacional em gerar todos os possíveis conjuntos de rotas de cardinalidade N, apenas as rotas com uma certa proximidade são unidas.

Para avaliar a proximidade das rotas utiliza-se a diferença angular do centro de gravidade de cada rota. Para isto, calcula-se para todas as rotas os seus centros de gravidade. O referencial para cada rota é seu centro de gravidade.

O centro de gravidade é calculado segundo a equação 3.7.

$$centroGravidadeRota = \frac{\sum demanda_i * coord_i}{\sum demanda_i} \quad \forall i \in Rota_i, Rota_i \in Solucao \tag{3.7}$$

Geração dos Movimentos

A geração dos movimentos é composta das etapas: seleção das rotas, união das rotas e segregação da grande rota resultante em conjuntos de tamanhos variados.

1 - Seleção das rotas

A seleção das rotas à união se dá através da aplicação do algoritmo de varredura sob os centros de gravidade de cada rota. Este procedimento é inspirado na etapa de decomposição do algoritmo D-Ants (REIMANN; DOERNER; HARTL, 2004).

Para isto, os centros de gravidade devem estar ordenados em ordem crescente em relação à angulagem. O procedimento consiste em selecionar N centros de gravidade a partir do primeiro com menor ângulo.

ALGORITMO 8 Seleção das Rotas

1: Ler sol, N
2: **for** $i = 0 \rightarrow (NumDeRotas - N)$ **do**
3: Criar conjunto de rotas com indice $i \rightarrow (i+N)$ {Criar conjuntos de rotas com indices de i até i + N}
4: $i \leftarrow i + 1$
5: **end for**

2 - União das rotas

Tendo constituído os conjuntos de rotas na etapa anterior (etapa de seleção), faz-se necessário a criação de uma grande rota a partir das rotas pertencentes a estes conjuntos.

Quando as rotas de uma solução são bem distribuídas geograficamente, a união destas rotas é trivial. Simplesmente podem-se unir os pontos extremos das rotas centrais.

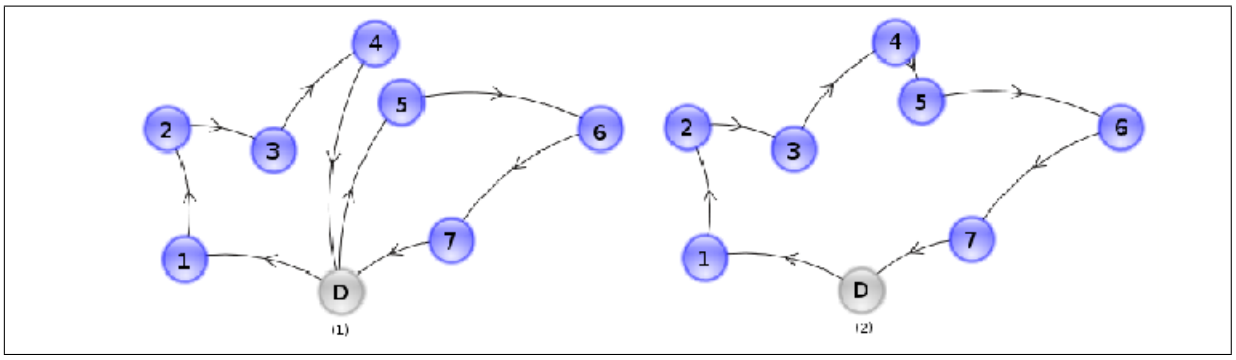

FIGURA 3.7: Exemplo do procedimento de união. À esquerda (1), antes da união das rotas. À direita (2), após a união.

Porém nem todas as soluções possuem rotas bem distribuídas. Na ilustração abaixo, nota-se a presença de rotas que ocupam a mesma região. Neste caso, aplicar o procedimento anterior de união nem sempre é uma boa alternativa ao passo que a grande rota pode não permanecer tão circular quanto desejada.

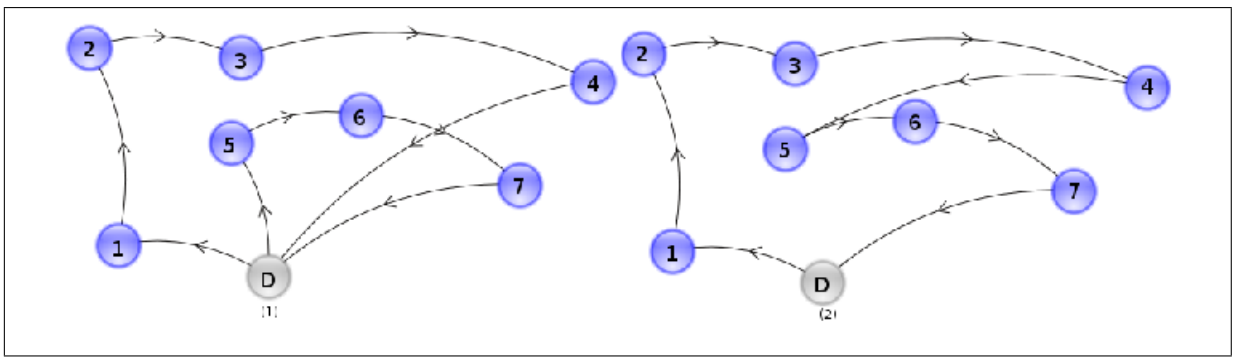

FIGURA 3.8: Exemplo do procedimento de união. À esquerda (1), antes da união das rotas. À direita (2), após a união.

Como alternativa a este caso, utiliza-se a aplicação do algoritmo de varredura aos pontos pertencentes a estas rotas. Desta forma, sem considerar nenhuma restrição do modelo matemático (capacidade do veículo, por exemplo), constrói-se uma grande rota onde o elo de ligação entre os pontos de demanda é a distância angular, vide seção 3.1.2. Abaixo uma representação deste procedimento.

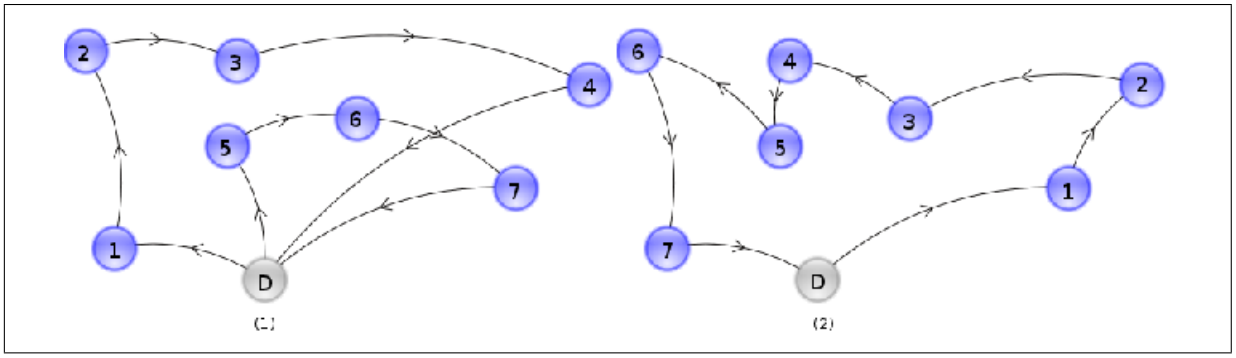

FIGURA 3.9: Exemplo do procedimento de união. À esquerda (1), antes da união das rotas. À direita (2), após a união com o procedimento de varredura.

3 - Divisão da Grande Rota

Este procedimento contempla a segregação das grandes rotas construídas na etapa anterior, isto é, a divisão destas rotas em rotas menores. A formação de novas rotas tem o intuito de minimizar o custo da solução.

O procedimento realiza cortes nas grandes rotas de tamanhos variados a fim de gerar novos conjuntos de rotas e alterar a cardinalidade de rotas da solução. Por exemplo, se a variável N (que indica o número de rotas a unir) for igual a 3, realiza-se particionamentos de 3 conjuntos, de 2 conjuntos e de 1 conjunto. Desta forma geram-se vários planos de corte desta grande rota. Cada plano de corte desta grande rota compõe um possível movimento do algoritmo.

Nesta etapa de divisão não são consideradas as restrições do problema, ou seja, divide-se a rota em novos conjuntos objetivando tão somente a minimização do custo envolvido sem, no entanto, preocupar-se com a factibilidade do movimento.

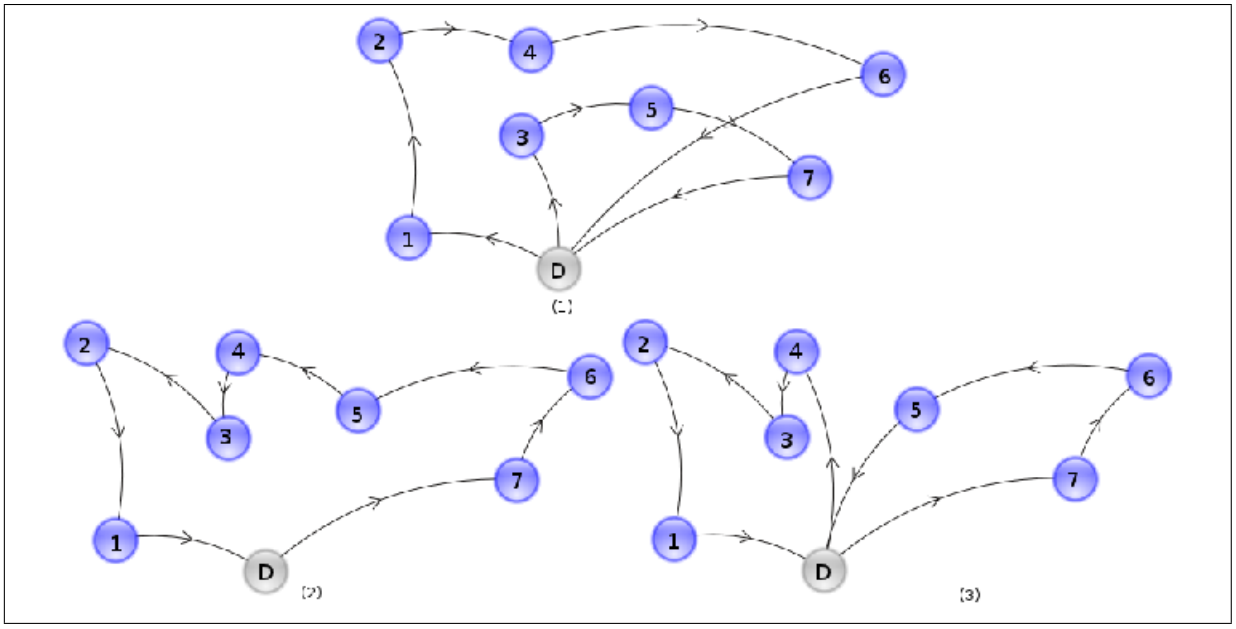

FIGURA 3.10: Exemplo do procedimento de divisão da grande rota. Em (1) a rota original; em (2) a grande rota; em (3) um possível reparticionamento.

Avaliação dos Movimentos

A etapa de avaliação compreende o procedimento de mensurar a qualidade dos movimentos. Os movimentos necessitam de mensuração para, posteriormente, serem ordenados em ordem decrescente em relação ao ganho.

O cálculo é realizado através da razão dos somatórios dos custos das novas rotas geradas pela etapa de divisão e os custos das rotas originais envolvidas.

A atratividade do movimento é representado por ξ_m na equação 3.8. Considera-se R o conjunto de rotas envolvidas no movimento m e R' o conjuntos das novas rotas pertencentes ao movimento m.

$$\xi_m = \frac{\sum_{\forall r' \in R'} Custo(Rota_{r'})}{\sum_{\forall r \in R} Custos(Rota_r)} \quad \forall m \in Movimentos \tag{3.8}$$

Desta forma, valores de $\xi_m \geq 1$ não são atrativos pois não geram ganhos para a solução.

Procedimento de Factibilização

Após a avaliação dos movimentos e a ordenação dos movimentos em relação a suas atratividades, inicia-se o processo de efetuar as mudanças na solução. Em muitos casos, os planos de cortes gerados na etapa de divisão são movimentos infactíveis. Embora reduzam o custo da solução, geram rotas que não respeitam as capacidades dos veículos, por exemplo.

Neste sentido, avalia-se a possibilidade de tornar tais movimentos factíveis permanecendo atrativos em relação ao custo da solução. Para isto, utiliza-se uma abordagem de deslocamentos de pontos entre rotas vizinhas e o conceito de movimentos compostos.

Nesta aplicação, movimentos compostos condizem com o fato de ocorrer deslocamentos entre a rota infactível e rotas vizinhas e, se necessário, também o deslocamentos de pontos de demanda entre a rota vizinha e a sua vizinhança. Tal procedimento, iterativamente, busca a factibilização da solução, porém é restrita ao número de deslocamentos entre rotas adjacentes. Tal cuidado é considerado para evitar possíveis *loops* ou que o algoritmo entre em um ciclo de deslocamentos e nunca consiga factibilizar o movimento. Nesta implementação, utiliza-se como parâmetro de deslocamento máximo o número de rotas da solução, sendo que não é possível deslocar pontos de demanda entre rotas mais vezes do que o número de rotas.

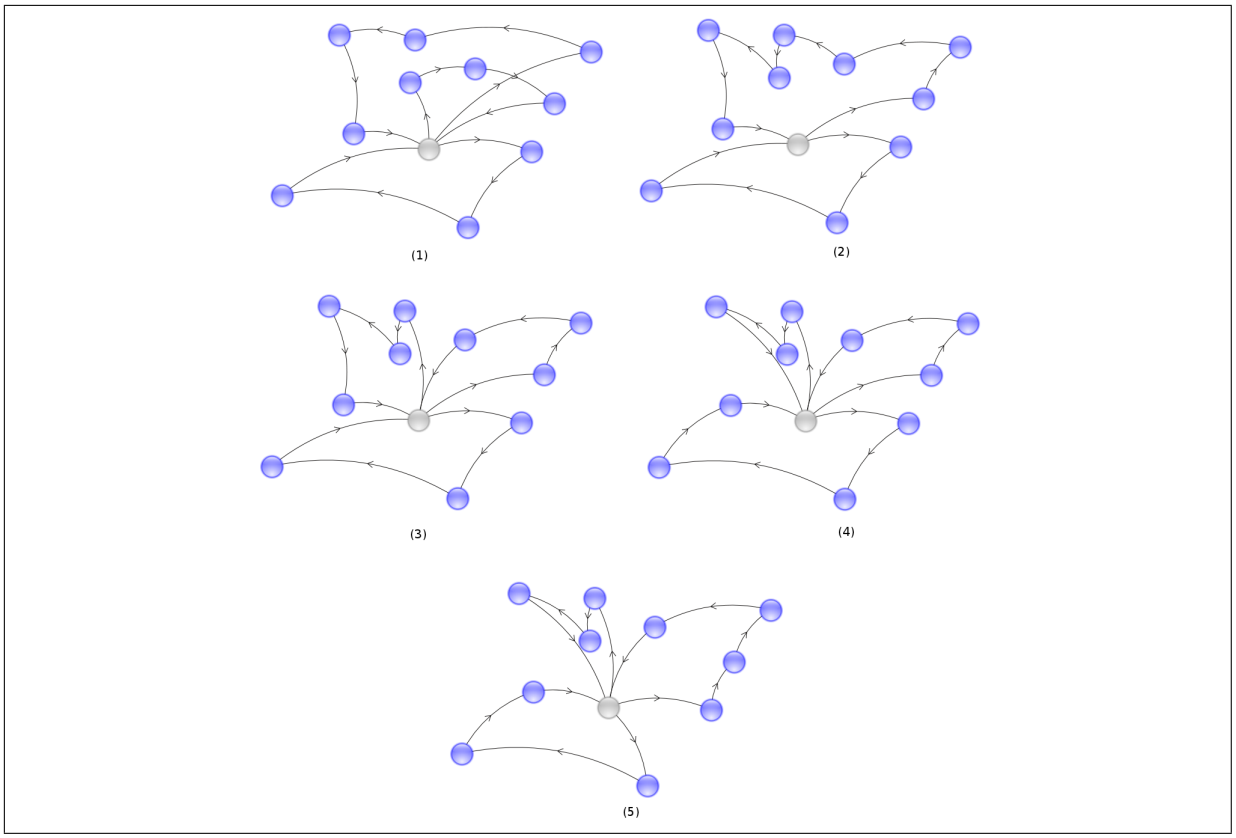

FIGURA 3.11: Exemplo de um processo de factibilização. Em (1) a rota original. Em (2) a junções de 2 rotas em uma grande rota. Em (3) mostra o novo particionamento desta grande rota. Em (4) e (5) são mostrado movimentos compostos (deslocamentos) entre rotas para tornar viável a solução.

4 *Análise de resultados*

A partir da implementação dos métodos propostos e da aplicação destes a um conjunto de instâncias presentes em Christofides et al. (CHRISTOFIDES; MINGOZZI; TOTH, 1979) é possível avaliar a qualidade das soluções obtidas. Tais instâncias são amplamente utilizadas na literatura como referência para os métodos de resolução aplicados ao PRVC.

Apesar de existirem inúmeros conjuntos de instâncias na literatura, o conjunto proposto por Christofides et al. apresenta algumas particularidades na disposição dos consumidores. Em certas instâncias, a disposição dos consumidores se dá em grupos geograficamente distribuídos o que instiga a aplicação de métodos construtivos e de melhoramento diferentes para a avaliação dos comportamentos.

Pelas características de agrupamento e roteamento das instâncias, tal conjunto é bastante relevante no estudo dos possíveis métodos a serem aplicados ao problema de despacho de ordens de serviço pois as ordens também possuem tais características. Tal análise investiga uma combinação de heurísticas que possuam um bom comportamento nas situações de roteamento e roteamento com agrupamento.

Busca-se, com isso, a mensuração da qualidade obtida pelos métodos estudados e prover embasamento estatístico para auxiliar na tomada das decisões futuras quanto à utilização de abordagens heurísticas para a resolução do PDOS.

O conjunto de instâncias proposto em (CHRISTOFIDES; MINGOZZI; TOTH, 1979) apresenta custos ótimos conhecidos e descritos na tabela 4.1. Tais custos são utilizados para calcular o desvio dos resultados obtidos nas heurísticas avaliadas. No restante do texto, o desvio é referenciado como *GAP* e representa o quociente entre o custo da função objetivo obtido e o seu custo ótimo correspondente. As instâncias vrpnc11 e vrpnc12 apresentam as características de roteamento e agrupamento com maior intensidade.

Utilizou-se a linguagem de programação C++ com o compilador do GNU (g++), o utilitário QMake como gerador de arquivos *Makefile* e o *Framework* Qt, desenvolvido e disponibilizado por Nokia sob a licença LGPL. Todos os algoritmos foram compilados com a *flag* de otimização

-O3 (otimização de código). Todos os testes foram realizados utilizando um único *core* de um Intel Core i5 M40 2.4 Ghz e 4GB de RAM, com índice de BOGOMIPS de 4787,00, sob o sistema operacional Linux Ubuntu 10.04 X64.

TABELA 4.1: Custos ótimos para as instâncias propostas em (CHRISTOFIDES; MINGOZZI; TOTH, 1979).

Instâncias	Número de pontos	Custos ótimos conhecidos
vrpnc1	50	524.61
vrpnc2	75	835.26
vrpnc3	100	826.14
vrpnc4	150	1028.42
vrpnc5	199	1291.29
vrpnc11	120	1042.11
vrpnc12	100	819.56

4.1 Análise dos métodos construtivos

Esta seção tem a finalidade de analisar os resultados obtidos através das implementações das heurísticas construtivas descritas no capítulo 3. Analisa-se o desempenho computacional mensurado através do tempo de execução no ambiente acima descrito e a qualidade das soluções através do valor final da função objetivo.

A implementação do algoritmo das economias não requer a definição de parâmetros ao seu funcionamento e, portanto, a implementação do algoritmo é tal qual descrita no capítulo 3. Já o algoritmo de varredura, analisa-se somente a etapa de *clustering* por considerar-se que esta etapa compreende a etapa construtiva nas implementações desenvolvidas. Na implementação do algoritmo de varredura com variação do ponto inicial, são executadas $N/2$ vezes o mesmo procedimento, onde N é igual ao número de pontos de demanda (consumidores). A cada execução, o ponto inicial é escolhido aleatoriamente. Na implementação proposta por Goldbarg e Luna (GOLDBARG; LUNA, 2005) é executado o procedimento apenas uma única vez e nesta é selecionado como ponto inicial aquele com menor ângulo polar.

Abaixo, as tabelas com os custos da função objetivo para cada instância de teste além do tempo requerido para a execução.

TABELA 4.2: Resultados para a aplicação da heurística das economias (CLARKE; WRIGHT, 1964) ao conjunto de instâncias proposto em (CHRISTOFIDES; MINGOZZI; TOTH, 1979)

Instância	Custo	Tempo(s)	GAP
vrpnc1	584,63	0,001	11,44%
vrpnc2	900,24	0,002	7,77%
vrpnc3	886,82	0,004	7,38%
vrpnc4	1133.43	0,011	10,21%
vrpnc5	1395,74	0,027	8,09%
vrpnc11	1071,07	0,007	2,78%
vrpnc12	833,50	0,010	1,70%
		Média	7,05%
		Mediana	7,77%
		Desvio-padrão	3,60%

TABELA 4.3: Resultados para a aplicação da etapa construtiva da heurística de varredura (GILLET; MILLER, 1974) ao conjunto de instâncias proposto em (CHRISTOFIDES; MINGOZZI; TOTH, 1979). Ponto inicial: ponto com menor ângulo polar

Instância	Custo	Tempo(s)	GAP
vrpnc1	856,69	0,0005	63,30%
vrpnc2	1306,90	0,001	56,45%
vrpnc3	1433,31	0,002	73,49%
vrpnc4	2199,03	0.0025	113,83%
vrpnc5	2612,01	0,003	102,28%
vrpnc11	3511,22	0,001	236,93%
vrpnc12	1179,19	0,001	43,88%
		Média	98,59%
		Mediana	73,49%
		Desvio-padrão	65,86%

TABELA 4.4: Resultados para a aplicação da etapa construtiva da heurística de varredura com variação do ponto inicial (GILLET; MILLER, 1974) ao conjunto de instâncias proposto em (CHRISTOFIDES; MINGOZZI; TOTH, 1979)

Instância	Custo	Tempo(s)	GAP
vrpnc1	812,832	0,001	54,94%
vrpnc2	1233,45	0,005	47,65%
vrpnc3	1426,31	0,002	72,65%
vrpnc4	2038,96	0.0025	98,26%
vrpnc5	2607,70	0,003	101,95%
vrpnc11	3044,64	0,005	192,16%
vrpnc12	1103,92	0,019	34,70%
		Média	86,04%
		Mediana	72,65%
		Desvio-padrão	53,10%

Pela análise dos dados, nota-se que o algoritmo das economias apresenta melhores resultados em relação a etapa construtiva do algoritmo de varredura de Gillet e Miller. Em média, o algoritmo das economias apresenta disparidade de 7,05% em relação ao ótimo conhecido. Já o algoritmo de varredura apresenta disparidade extremamente alta, 98,5% acima do ótimo conhecido. A razão para tamanha disparidade entre as duas abordagens advém da abordagem utilizada pelo algoritmo de varredura. Tal algoritmo é planejado para executar a etapa de *clustering* e depois a etapa de *routing*. Nesta análise, verifica-se o desempenho somente da etapa de *clustering*. O procedimento de varredura com aleatoriedade apresenta resultados superiores que a abordagem determinística do mesmo procedimento. Na figura 4.1, mostra-se a relação entre as instâncias e seus desvios em relação aos ótimos conhecidos (GAP).

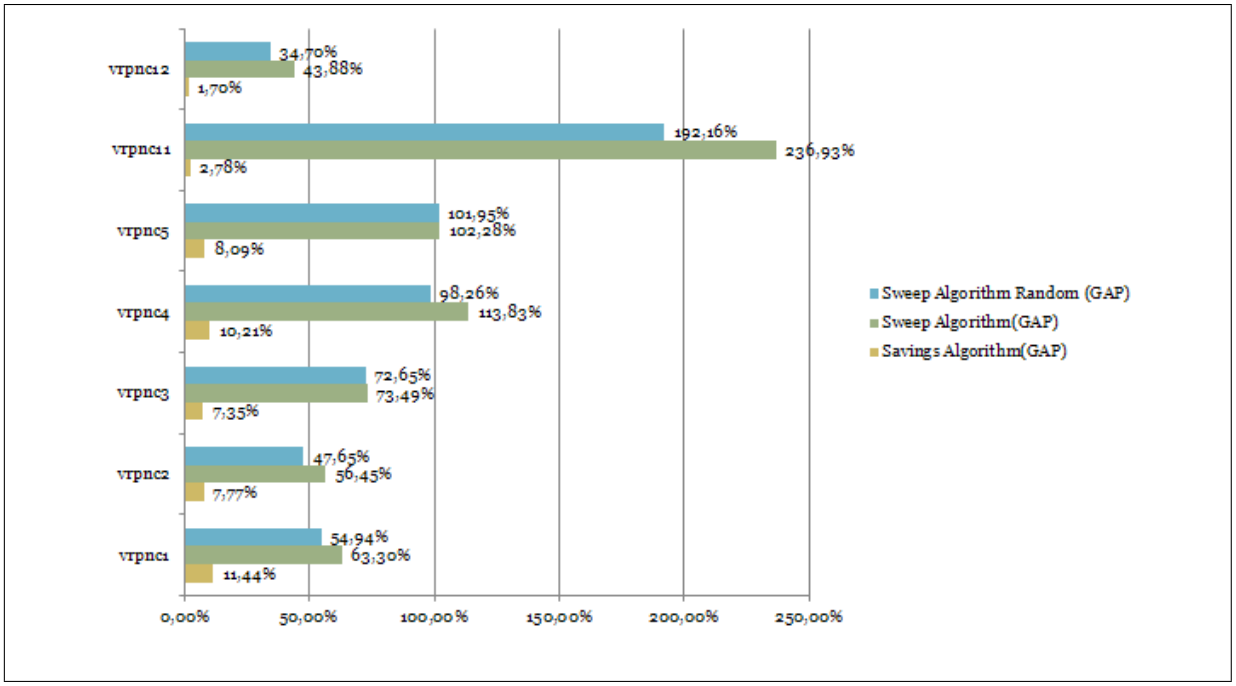

FIGURA 4.1: Comparativo entre os GAPs atingidos com a aplicação dos métodos construtivos ao conjunto de instâncias.

Na tabela 4.5, compara-se as três abordagens construtivas: algoritmos das economias, algoritmo de varredura determinístico e algoritmo de varredura aleatório. Relaciona-se o método construtivo, as instâncias, o custo da função objetivo e o tempo de execução.

TABELA 4.5: Comparativo entre as abordagens construtivas para o PRVC.

		Algoritmo das economias		Algoritmo de varredura		Algoritmo de varredura (aleatório)	
Instancias	n	Custo	Tempo(s)	Custo	Tempo(s)	Custo	Tempo(s)
vrpnc1	50	584,63	0,001	856,69	0,0005	812,83	0,005
vrpnc2	75	900,24	0,002	1306,90	0,001	1233,45	0,002
vrpnc3	100	886,82	0,004	1433,31	0,002	1426,31	0,0025
vrpnc4	120	1133,43	0,011	2199,31	0,0025	2038,96	0,003
vrpnc5	150	1395,74	0,027	2612,01	0,003	2607,70	0,003
vrpnc11	120	1071,07	0,007	3511,22	0,001	3044,64	0,005
vrpnc12	100	833,50	0,010	1179,19	0,001	1103,92	0,019
Média GAP		7,50%		98,59%		86,04%	

4.2 Análise das buscas locais

Esta seção busca analisar o impacto no custo das soluções a partir dos algoritmos de busca local aplicados às heurísticas construtivas desenvolvidas. Para tanto, analisa-se as heurísticas de busca local $k-opt$, $\lambda-interchange$ e $JoinRoutes$ com variações em seus parâmetros aplicadas à heurística de varredura e à heurística das economias.

Além da análise dos resultados obtidos da aplicação dos métodos de busca local aos construtivos, busca-se analisar a influência e o grau de melhoria atingido quando tais heurísticas de melhoramento são combinadas e aplicadas a um método construtivo.

Tal análise tem o objetivo de mensurar quais combinações entre algoritmos construtivos e de melhoramento possuem maior atratividade em relação ao custo da solução e tempo requerido de execução.

4.2.1 Heurística de varredura

Apresenta-se, na tabela 4.6, o desempenho das heurísticas de busca local aplicadas ao algoritmo de varredura. Na implementação do $k-opt$ foi utilizado o parâmetro $k=2$ e $k=3$. Na implementação do $\lambda-interchange$ foi utilizado o parâmetro $\lambda=1$. Na implementação do algoritmo *JoinRoutes*, utilizou-se o parâmetro $N=2$ e $N=3$. Em todas as implementações foi utilizado a abordagem de seleção de vizinhos *best improvement*. Segue, a tabela com os custos da função objetivo para cada instância, o tempo requerido para a execução e o percentual de melhoria em relação ao método construtivo. Representa-se o custo da função objetivo como C, o tempo de execução como t(s) e a melhoria em relação ao método construtivo como G.entre

TABELA 4.6: Algoritmos de busca local aplicados a heurísticas de varredura.

Métodos		Instâncias						
		vrpnc1	vrpnc2	vrpnc3	vrpnc4	vrpnc5	vrpnc11	vrpnc12
	C	590,12	906,39	857,41	1104,48	1448,43	1385,40	991,74
2-opt	t(s)	0,001	0,001	0,001	0,002	0,004	0,003	0,001
	G	31,12%	30,64%	40,18%	49,77%	44,55%	60,54%	15,90%
	C	608,52	940,37	880,24	1152,91	1457,76	1397,46	1004,97
3-opt	t(s)	0,015	0,027	0,054	0,120	0,200	0,182	0,130
	G	28,97%	28,05%	38,59%	47,57%	44,19%	60,20%	14,77%
	C	603.59	965.95	881.17	1203.47	1526.84	1497.84	1089.15
1-interchange	t(s)	0.005	0,011	0,019	0,089	0,171	0,058	0,010
	G	29,54%	26,09%	38,52%	45,27%	41,55%	57,34%	7,64%
	C	716.38	952.63	1051,83	1226,24	1629,05	1302,03	1089,14
JoinRoutes-2	t(s)	1,011	1,002	1,008	2,021	3,024	1,016	1,005
	G	16,38%	27,11%	26,62%	44,24%	37,63%	62,92%	7,64%
	C	591,18	993,15	991,27	1283,39	1590,38	1243,07	1014,78
JoinRoutes-3	t(s)	2,001	2,003	2,011	3,022	4,037	2,022	2,037
	G	30,99%	24,01%	30,84%	41,64%	39,11%	64,60%	13,94%

Após a aplicação dos algoritmos de busca local na heurística de varredura, nota-se o refinamento da solução e a melhora substancial do valor da função objetivo em relação aos valores obtidos na etapa de *clustering*, tabela 4.5. A clássica heurística $2-opt$ apresentou a melhor média percentual de melhoria em relação às soluções construídas pelo algoritmo de varredura.

As soluções iniciais construídas pelo algoritmo de varredura apresentam rotas geograficamente bem distribuídas e geralmente irregulares (não circulares). Essa não circularidade advém da etapa de *clustering* onde cada ponto de demanda é selecionado de acordo com sua angulagem em relação ao depósito mais próximo. Como a heurística $k-opt$ atua na troca de arcos da solução e as trocas primam por tornar as rotas mais circulares, tal heurística se comportou de forma a contemplar os melhores resultados.

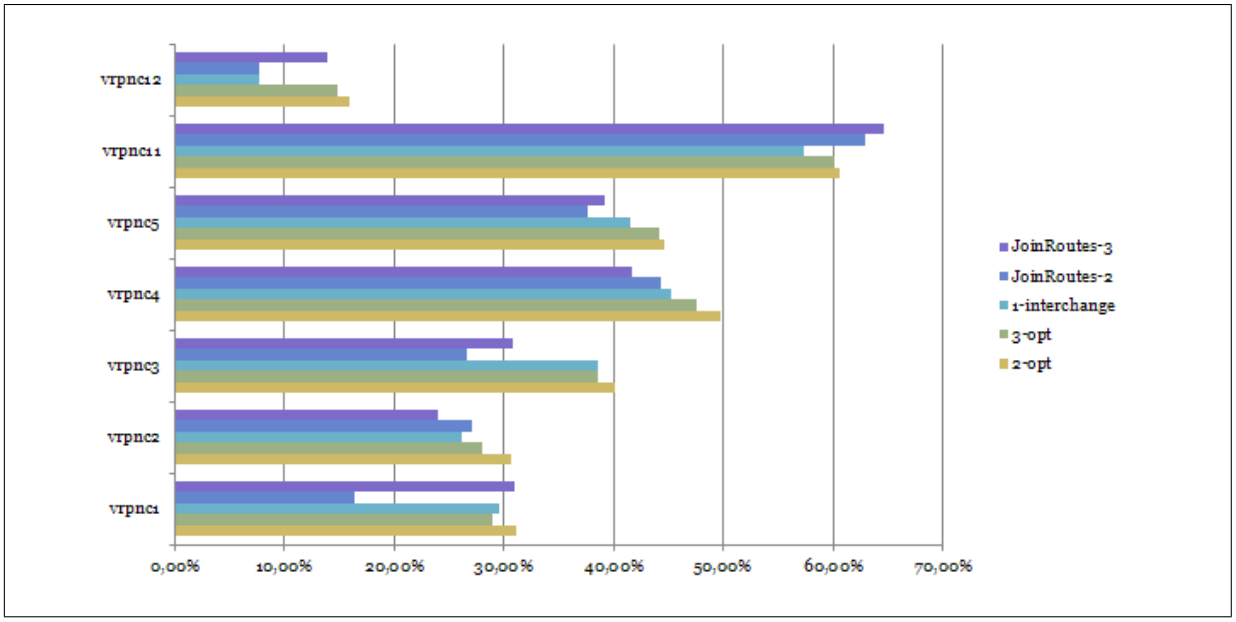

FIGURA 4.2: Gráfico comparativo entre os percentuais de melhoria atingidos com a aplicação das buscas locais ao algoritmo de varredura.

Quanto aos outros métodos, todos apresentaram uma média de melhoria acima de 30%. O algoritmo $k-opt$, quando utilizado o parâmetro $k = 3$, apresenta uma mínima diferença na média de melhoria quando da utilização de $k = 2$. Embora o esforço computacional e a enumeração da vizinhança sejam maiores, não há a garantia de melhores resultados ao fim do processo. Comparando o método *JoinRoutes* com os métodos de busca local clássicos, este mostra-se competitivo em relação aos resultados obtidos para os valores da função objetivo. Apresenta para algumas instâncias valores superiores aos atingidos pelas demais heurísticas. Quanto ao tempo requerido para a execução, o algoritmo *JoinRoutes* apresenta valores superiores que as demais abordagens.

Buscas locais combinadas

Tendo avaliado o grau de melhoria das buscas locais aplicadas ao método construtivo proposto por Gillet e Miller (GILLET; MILLER, 1974), busca-se analisar se a combinação dos métodos de busca local aplicados ao algoritmo de varredura é atrativo em termos de melhoria dos custos atingidos. Para tanto, analisa-se progressivamente a combinação dos métodos de busca local com o intuito de avaliar o quão significativo é a inclusão desse método ao procedimento.

São analisadas as seguintes combinações de buscas locais aplicadas ao algoritmo de varredura:

1. C1 = construtivo + 2opt + 3opt

2. C2 = construtivo + 2opt + 3opt + 1-interchange

3. C3 = construtivo + 2opt + 3opt + 1-interchange + JoinRoutes-2

4. C4 = construtivo + 2opt + 3opt + 1-interchange + JoinRoutes-2 + JoinRoutes-3

Na tabela , representa-se o custo da função objetivo como C, o tempo de execução como t(s) e a melhoria como G. O ganho G é sempre calculado como sendo a melhoria em relação à combinação anterior, isto é, o valor percentual da melhoria gerada a partir da inclusão de um método de busca local. Na combinação 1, o G é calculado em relação valor obtido pela aplicação do algoritmo $2-opt$ ao algoritmo de varredura.

TABELA 4.7: Combinações de heurísticas de busca local aplicadas a heurísticas de varredura.

		Instâncias						
Combinações		vrpnc1	vrpnc2	vrpnc3	vrpnc4	vrpnc5	vrpnc11	vrpnc12
	C	588,29	906,39	854,95	1096,38	1439,85	1379,66	990,80
C1	t(s)	0,40	0,23	0,39	0,13	0,44	0,20	0,22
	G	0,31%	0,00%	0,29%	0,73%	0,59%	0,41%	0,09%
	C	559,71	906,42	846,10	1096,38	1432,89	1360,88	990,80
C2	t(s)	0,42	0,30	0,45	0,21	0,60	0,29	0,24
	G	4,86%	0,00%	1,04%	0,00%	0,48%	1,36%	0,00%
	C	551,52	905,47	846,10	1091,96	1432,89	1356,25	989,02
C3	t(s)	1,21	1,39	1,31	2,12	3,87	1,16	1,31
	G	1,46%	0,11%	0,00%	0,40%	0,00%	0,34%	0,18%
	C	551,52	905,47	846,10	1091,96	1430,90	1356,25	989,02
C4	t(s)	3,05	2,89	3,51	5,23	6,18	3,42	4,02
	G	0,00%	0,00%	0,00%	0,00%	0,14%	0,00%	0,020%

Como se verifica na tabela 4.7, a combinação de métodos de busca local aplicados ao algoritmo construtivo de varredura não provê melhorias significativas no custo da função objetivo. Nota-se que o esforço computacional, neste caso, não é proporcional ao melhoramento das soluções. Tais métodos, mesmo que combinados, não conseguem escapar dos mínimos locais atingidos.

TABELA 4.8: Comparativo entre os GAPs atingidos pela aplicação dos métodos de busca local à heurística de varredura.

		Instâncias							
		vrpnc1	vrpnc2	vrpnc3	vrpnc4	vrpnc5	vrpnc11	vrpnc12	
									Média
C1	GAP	12,1%	8,5%	3,5%	6,6%	11,5%	32,4%	20,9%	13,6%
C2	GAP	6,7%	8,5%	2,4%	6,6%	11,0%	30,6%	20,9%	12,4%
C3	GAP	5,1%	8,4%	2,4%	6,2%	11,0%	30,1%	20,7%	12,0%
C4	GAP	5,1%	8,4%	2,4%	6,2%	10,8%	30,1%	20,7%	12,0%

4.2.2 Heurísticas de varredura com variação do ponto inicial

Tendo analisado o desempenho das buscas locais aplicadas ao algoritmo de varredura determinístico, esta subseção tem o objetivo de analisar e mensurar a influência das heurísticas de busca local aplicadas ao algoritmo de varredura aleatório. Neste sentido, analisam-se os mesmos algoritmos analisados no procedimento determinístico. Na implementação do $k-opt$ foi utilizado o parâmetro $k = 2$ e $k = 3$. Na implementação do $\lambda - interchange$ foi utilizado o parâmetro $\lambda = 1$. Na implementação do algoritmo *JoinRoutes*, utilizou-se o parâmetro $N = 2$ e $N = 3$. Em todas as implementações foi utilizada a abordagem de seleção de vizinhos *best improvement*. Nesta implementação do algoritmo de varredura, executa-se 10 vezes o algoritmo e a cada execução são iterados $N/2$ vezes para cada instância, onde N = número de consumidores da instância. Das 10 execuções, apresenta-se a melhor solução com seu tempo de execução.

Segue a tabela com os custos da função objetivo para cada instância, o tempo requerido para a execução e o percentual de melhoria em relação ao método construtivo. Representa-se o custo da função objetivo como C, o tempo de execução como t(s) e a melhoria em relação ao método construtivo como G.

TABELA 4.9: Algoritmos de busca local aplicados a heurísticas de varredura.

Métodos		Instâncias						
		vrpnc1	vrpnc2	vrpnc3	vrpnc4	vrpnc5	vrpnc11	vrpnc12
	C	531,90	902,77	852,41	1088,29	1420,96	1267,29	920,25
2-opt	t(s)	0,007	0,014	0,034	0,095	0,158	0,122	0,030
	G	52,82%	36,63%	67,33%	87,35%	83,52%	140,25%	19,94%
	C	546,81	909,65	870,64	1111,18	1442,10	1295,38	966,59
3-opt	t(s)	0,375	0,673	1,361	3,625	5,934	4,732	3,250
	G	48,60%	35,60%	63,80%	83,50%	80,80%	135,00%	14,20%
	C	584,81	915,36	910,59	1153,62	1489,24	1327,43	985,59
1-interchange	t(s)	0.125	0,275	0,475	2,225	4,275	1,450	2,500
	G	39,00%	34,80%	56,60%	76,70%	75,10%	129,40%	12,00%
	C	616,89	921,61	948,05	1160,89	1442,19	1302,03	1034,78
JoinRoutes-2	t(s)	50,011	50,002	50,008	2,021	3,024	1,016	1,005
	G	31,80%	31,6%	50,40%	75,60%	80,80%	133,80%	6,70%
	C	595,03	921,61	982,33	1216,38	1590,38	1243,07	1014,78
JoinRoutes-3	t(s)	2,001	2,003	2,011	3,022	4,037	2,022	2,037
	G	36,40%	33,80%	45,20%	67,50%	64,00%	144,90%	8,80%

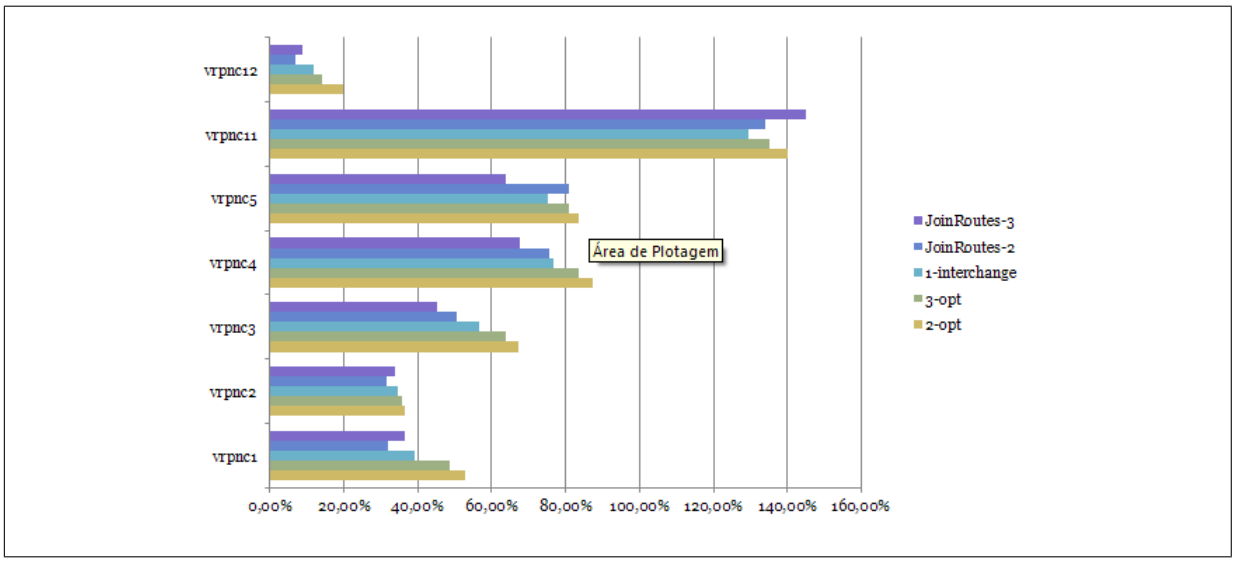

FIGURA 4.3: Gráfico comparativo entre os percentuais de melhoria atingidos com a aplicação das buscas locais ao algoritmo de varredura aleatório.

Com a abordagem aleatória do algoritmo de varredura, nota-se uma melhora nos desvios dos resultados em relação aos ótimos conhecidos. Da mesma forma que na implementação determinística, a busca local $2-opt$ foi responsável pelos melhores resultados em comparação com os outros algoritmos aplicados. A abordagem aleatória apresenta um custo computacional maior em decorrência do número de iterações executadas.

A média dos melhoramentos para todos os métodos foi acima de 50%. A melhora dos resultados em relação à abordagem determinística é atingida pois há uma maior variabilidade das soluções iniciais. Assim, os métodos de busca locais conseguem explorar regiões não atingíveis a partir da solução inicial gerada deterministicamente.

Buscas locais combinadas

Tendo avaliado o grau de melhoria das buscas locais ao método construtivo de varredura aleatório, busca-se analisar se a combinação dos métodos de busca local aplicados ao algoritmo é atrativa em termos de melhoria dos custos atingidos. Para tanto, analisa-se progressivamente a combinação dos métodos de busca local com o intuito de avaliar o quão significativa é a inclusão desse método ao procedimento aleatório.

São analisadas as seguintes combinações de buscas locais aplicadas ao algoritmo de varredura:

1. C1 = construtivo + 2opt + 3opt
2. C2 = construtivo + 2opt + 3opt + 1-interchange
3. C3 = construtivo + 2opt + 3opt + 1-interchange + JoinRoutes-2
4. C4 = construtivo + 2opt + 3opt + 1-interchange + JoinRoutes-2 + JoinRoutes-3

Na tabela 4.13, representa-se o custo da função objetivo como C, o tempo de execução como t(s) e a melhoria como G. O ganho G é sempre calculado como sendo a melhoria em relação a combinação anterior, isto é, o valor percentual da melhoria gerada a partir da inclusão de um método de busca local. Na combinação 1, o G é calculado em relação valor obtido pela aplicação do algoritmo $2-opt$ ao algoritmo de varredura aleatório.

TABELA 4.10: Combinações de heurísticas de busca local aplicadas a heurísticas de varredura aleatória.

		Instâncias						
Combinações		vrpnc1	vrpnc2	vrpnc3	vrpnc4	vrpnc5	vrpnc11	vrpnc12
	C	531,90	879,59	846,64	1083,99	1420,54	1264,10	906,76
C1	t(s)	10,05	5,75	9,75	3,25	11,00	5,00	5,50
	G	0,00%	2,57%	0,68%	0,40%	0,03%	0,25%	1,47%
	C	526,90	856,92	843,96	1068,16	1385,14	1067,39	837,48
C2	t(s)	10,50	7,50	11,25	5,25	15,00	7,25	6,03
	G	0,94%	2,58%	0,32%	1,46%	2,49%	15,56%	7,64%
	C	524,61	848,41	834,26	1068,16	1385,14	1068,11	833,72
C3	t(s)	30,25	34,75	32,75	53,00	96,75	29,00	32,75
	G	0,44%	0,99%	1,15%	0,00%	0,00%	0,40%	0,45%
	C	524,61	848,41	834,26	1068,16	1385,14	1068,11	833,72
C4	t(s)	76,25	72,25	87,75	130,75	154,45	85,50	100,50
	G	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%

Embora o custo computacional seja maior que a abordagem determinística, nota-se que a média dos desvios em relação aos ótimos conhecidos é, para certas combinações, realmente atrativa. Para as combinações C2 e C3 as médias dos GAPs são, respectivamente, 3% e 2,5%, como mostrado na figura 4.11. Para a instância vrpnc1, a combinação de buscas locais atingiu um custo da função objetivo igual ao custo ótimo conhecido para tal instância. Além disto, nota-se que o algoritmo *JoinRoutes* teve uma pequena influência na melhoria dos custos atingidos na aplicação porém contribuiu para alcançar o custo ótimo para a instância vrpnc1.

TABELA 4.11: Comparativo entre os GAPs atingidos pela aplicação dos métodos de busca local à heurística de varredura aleatória.

		Instâncias							
		vrpnc1	vrpnc2	vrpnc3	vrpnc4	vrpnc5	vrpnc11	vrpnc12	
									Média
C1	GAP	1,4%	5,3%	2,5%	5,4%	10,0%	21,3%	10,6%	8,1%
C2	GAP	0,4%	2,6%	2,2%	3,9%	7,3%	2,4%	2,2%	3,0%
C3	GAP	0,0%	1,6%	1,0%	3,9%	7,3%	2,0%	1,7%	2,5%
C4	GAP	0,0%	1,6%	1,0%	3,9%	7,3%	2,0%	1,7%	2,5%

4.2.3 Heurística das economias

Esta subseção apresenta e analisa os resultados obtidos a partir da aplicação de métodos de busca local à heurística das economias. Tais análises objetivam verificar se a aplicação dos métodos e suas combinações são eficientes quanto ao custo da função objetivo e quanto ao custo computacional requerido à execução. Os métodos de busca local analisados são: $k-opt$, $\lambda-interchange$ e $JoinRoutes$ com variações em seus parâmetros.

Segue a tabela com os custos da função objetivo para cada instância, o tempo requerido para a execução e o percentual de melhoria em relação ao método construtivo. Representa-se o custo da função objetivo como C, o tempo de execução como t(s) e a melhoria em relação ao método construtivo como G.

TABELA 4.12: Algoritmos de busca local aplicados a heurísticas das economias.

Métodos		Instâncias						
		vrpnc1	vrpnc2	vrpnc3	vrpnc4	vrpnc5	vrpnc11	vrpnc12
	C	584,63	888,08	882,04	1129,70	1389,60	1052,99	825,67
2-opt	t(s)	0,001	0,002	0,008	0,021	0,024	0,016	0,005
	G	0,00%	1,40%	0,50%	0,30%	0,33%	1,00%	0,95%
	C	584,63	888,08	882,04	1129,70	1389,60	1052,99	825,67
3-opt	t(s)	0,010	0,015	0,032	0,045	0,051	0,024	0,023
	G	0,00%	1,40%	0,50%	0,30%	0,33%	1,00%	0,95%
	C	584,63	875,92	882,04	1129,64	1387,87	1049,60	824,22
1-interchange	t(s)	0,002	0,003	0,005	0,015	0,036	0,008	0,011
	G	0,00%	2,80%	0,54%	0,34%	0,57%	2,05%	1,13%
	C	575,20	888,08	870,10	1059,64	1300,12	1060,10	830,90
JoinRoutes-2	t(s)	1,122	1,213	1,534	1,900	3,200	2,009	2,200
	G	1,6%	1,4%	1,92%	6,96%	7,35%	1,03%	0,31%
	C	580,23	850,87	880,90	1040,40	1330,54	1071,07	828,90
JoinRoutes-3	t(s)	2,101	2,303	2,011	4,122	5,123	1,111	2,411
	G	0,80%	5,80%	0,67%	8,94%	4,90%	0,00%	0,56%

Como apresentado na tabela 4.2, o algoritmo das economias possui um bom desempenho quanto ao custo computacional e, em média, os desvios em relação aos custos ótimos conhecidos são baixos.

Quando aplicado o algoritmo de busca local $k-opt$ às soluções iniciais criadas por esta heurística construtiva, obtém-se uma média de redução inferior a 1%. A redução dos custos da solução não é tão expressiva porque as soluções geradas pela heurística construtiva possuem rotas bastante circulares, diferentemente da etapa construtiva do algoritmo de varredura, o que impede movimentações mais expressivas no espaço de soluções e por consequência uma redução do valor da função objetivo.

Uma característica das rotas geradas pelo algoritmo das economias é a presença de rotas não geograficamente distribuídas, como pode ser visualizado na figura 4.4. Tal característica

favorece que a aplicação do algoritmo $\lambda - interchange$ refine a estrutura dos *clusters* e aumente a qualidade das soluções, já que este é um procedimento de melhoria inter-rota. Tais características também favorecem a aplicação do algoritmo *JoinRoutes*, pois este também prima por melhorar a distribuição geográfica das rotas. Tais resultados podem ser verificados na tabela 4.12.

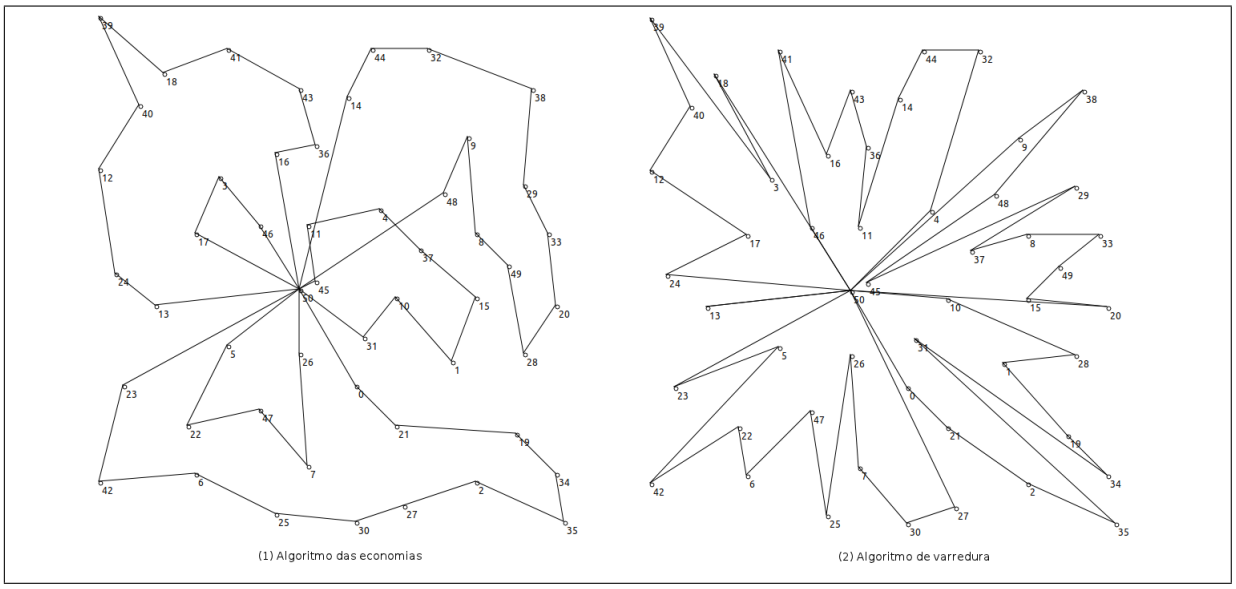

FIGURA 4.4: Exemplo de aplicação da heurísticas das economias e da heurística de varredura a instância vrpnc1, proposta em Christofides et al. (CHRISTOFIDES; MINGOZZI; TOTH, 1979)

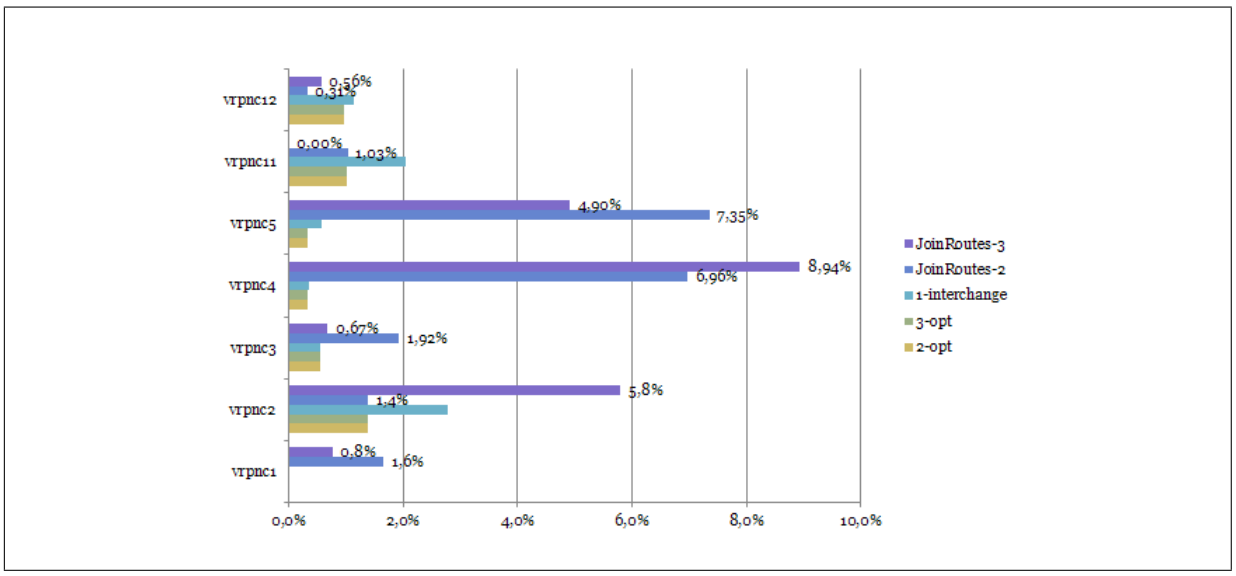

FIGURA 4.5: Gráfico comparativo entre os percentuais de melhoria atingidos com a aplicação das buscas locais ao algoritmo das economias.

Com a aplicação dos métodos de busca local ao algoritmo das economias, nota-se que os algoritmos de melhoramento inter-rota alcançam os melhores resultados. Verifica-se ainda que

o algoritmo *JoinRoutes*, com parâmetro $N = 3$, apresentou, em média, resultados superiores que as tradicionais abordagens de melhoramento, figura 4.12.

Apresenta-se, na figura 4.5, as melhorias percentuais obtidas pela aplicação dos métodos de busca local a heurística das economias.

Buscas locais combinadas

A partir da avaliação dos resultados obtidos da aplicação dos algoritmos de busca locais e já tendo verificado que os algoritmos de melhoramento inter-rota apresentam os melhores resultados, busca-se analisar se a combinação destas heurísticas com os métodos de melhoramento intra-rotas são atrativas em termos de melhoria dos custos atingidos. Para tanto, são analisadas as seguintes combinações de buscas locais aplicadas ao algoritmo das economias:

1. C1 = construtivo + JoinRoutes-2 + 2opt
2. C2 = construtivo + JoinRoutes-3 + 2opt
3. C3 = construtivo + $\lambda - interchange$ + 2-opt

Escolheu-se o método de melhoramento intra-rota $2 - opt$ pois este possui uma complexidade computacional inferior que o $3 - opt$ e porque apresenta, em média, valores equiparados a esta heurística.

Segue na tabela 4.13 os custos da função objetivo para cada instância, o tempo requerido para a execução e o percentual de melhoria em relação ao método construtivo. Representa-se o custo da função objetivo como C, o tempo de execução como t(s) e a melhoria em relação ao método construtivo como G.

TABELA 4.13: Combinações de heurísticas de busca local aplicadas a heurísticas das economias.

		Instâncias						
Combinações		vrpnc1	vrpnc2	vrpnc3	vrpnc4	vrpnc5	vrpnc11	vrpnc12
	C	576,88	873,54	878,92	1120,86	1370,15	1048,06	821,99
C1	t(s)	0,012	0,004	0,011	0,018	0,046	0,062	0,07
	G	1,34%	0,27%	0,36%	0,78%	1,29%	0,15%	0,27%
	C	575,20	880,98	861,39	1053,28	1300,12	1048,54	822,59
C2	t(s)	1,17	1,21	1,60	1,90	3,21	2,10	2,20
	G	0,00%	0,80%	1,00%	0,60%	0,00%	1,10%	1,00%
	C	569,20	850,87	874,73	1028,95	1317,23	1067,85	828,90
C3	t(s)	2,10	2,30	2,10	4,30	5,20	1,14	2,45
	G	1,90%	0,00%	0,70%	1,10%	1,00%	0,30%	0,00%

A partir dos resultados gerados pelas combinações de buscas locais aplicadas ao algoritmo das economias, nota-se que as combinações C2 e C3 são atrativas a um custo computacional baixo, figura 4.13. As combinações apresentam GAP de 3,35% e 3,13%, respectivamente. Os GAP para as instâncias podem ser verificados na tabela 4.14.

TABELA 4.14: Comparativo entre os GAPs atingidos pela aplicação dos métodos de busca local à heurística das economias.

		Instâncias							
		vrpnc1	vrpnc2	vrpnc3	vrpnc4	vrpnc5	vrpnc11	vrpnc12	
									Média
C1	GAP	9,96%	4.57%	6,39%	8,99%	6,11%	0,57%	0,30%	5,27%
C2	GAP	9,64%	5,46%	4,27%	2,42%	0,68%	0,62%	0,37%	3,35%
C3	GAP	8,50%	1,86%	5,88%	0,05%	2,01%	2,47%	1,14%	3,13%

4.3 Análise da meta-heurística

Esta seção tem o intuito de analisar os resultados gerados pela meta-heurísticas *D-Ants* aplicado às instâncias de Christofides et al. (CHRISTOFIDES; MINGOZZI; TOTH, 1979) e também apresentar os parâmetros utilizados no procedimento para alcançar tais resultados.

4.3.1 D-Ants

O procedimento *D-Ants*, por ser um procedimento meta-heurístico baseado em *Ant Colony* requer a definição de alguns parâmetros. Tais parâmetros definem algumas características e comportamentos do algoritmo construtivo utilizado bem como os aspectos inerentes à atualização do feromônio. Tais características influenciam diretamente no comportamento do algoritmo, como descreve Reimman et al. (REIMANN; DOERNER; HARTL, 2004).

Alguns parâmetros definidos são: o número de subproblemas a serem resolvidos pelo $D-Ants$, o fator de evaporação do feromônio (ρ), a influência do valor das economias (β) e a influência da quantidade de feromônio (α) na etapa construtiva, o número de soluções de elite e o número de formigas.

Os parâmetros utilizados foram baseados nas conclusões obtidas no trabalho de Reimann et al (REIMANN; DOERNER; HARTL, 2004). Segundo Reimann, a escolha do número de subproblemas a serem resolvidos pelo algoritmo deve ser influenciado pelo tamanho das instâncias do problema. Pelo tamanho das instâncias envolvidas neste trabalho e pela análise do trabalho de Reimman et al. foi definido os valores a serem utilizados nas execuções do método.

- Número de subproblemas = 3
- Variável ρ = 0.95
- Variável α = 5
- Variável β = 5
- Número de soluções de elite = 5
- Número de formigas = $N/2$, onde N é o número de pontos de demanda

TABELA 4.15: Resultados para a aplicação da meta-heurísticas D-Ants (REIMANN; DOERNER; HARTL, 2004) ao conjunto de instâncias proposto em (CHRISTOFIDES; MINGOZZI; TOTH, 1979)

Instância	Custo	Tempo(s)	GAP
vrpnc1	524,61	9,280	0,00%
vrpnc2	838,59	56,638	0,39%
vrpnc3	837,52	220,532	1,38%
vrpnc4	1047,59	1400,770	1,86%
vrpnc5	1321,69	4222,530	2,35%
vrpnc11	1043,89	914,750	0,17%
vrpnc12	819,56	362,509	0,00%
		Média	0,88%
		Mediana	0,39%
		Desvio-padrão	0,97%

Mostra-se na tabela 4.15 os valores obtidos pela aplicação da meta-heurística $D-Ants$. Os valores obtidos destacam-se pela qualidade e proximidade com os valores ótimos conhecidos em todas as instâncias. Em quatro das sete instâncias os desvios em relação aos custos ótimos permaneceram abaixo de 1%.

É necessário um tempo maior de computação do que nas simples heurísticas pelo fato destes procedimentos iterarem inúmeras vezes até atingirem bons resultados.

5 *Considerações Finais*

O objetivo deste trabalho foi o estudo, a análise e o desenvolvimento de métodos heurísticos de resolução para o problema de roteamento de veículos capacitado que, potencialmente, são aplicáveis ao problema de despacho de ordens de serviço.

Para tanto, a primeira etapa deste trabalho primou pelo estudo do problema de roteamento de veículos e suas variações com as possíveis formulações matemáticas. Assim, tal estudo contemplou uma visão geral e forneceu subsídios teóricos para o estudo dos métodos de resolução aplicáveis.

A segunda etapa consistiu no estudo e desenvolvimento de heurísticas clássicas do PRV. As heurísticas clássicas estudadas, tanto as construtivas quanto as de melhoramento, foram selecionadas por apresentarem comportamentos necessários ao entendimento dos meios cabíveis à resolução eficiente do PRV. Analisou-se a heurística das economias por esta apresentar uma enorme facilidade e eficiência no tratamento do roteamento e a heurística de varredura por ser uma método de solucionar problemas de roteamento onde as instâncias possuem particularidades de agrupamentos. O método de melhoramento $k-opt$ foi estudado para o entendimento das melhorias intra-rotas e o $\lambda-interchange$ para o melhoramento inter-rota.

Com o estudo do PRV e dos métodos de resolução, nota-se que a utilização de simples heurísticas à resolução do PRV apresenta uma deficiência na resolução de todas as instâncias. Como o conjunto de instâncias contém arranjos de consumidores organizados em grupos, a heurística de varredura e a heurística das economias não conseguem resolver todas as instâncias com um grau de qualidade pois ambas possuem uma metodologia restritiva de resolução do problema. Assim, nota-se que a maneira ideal de resolução que mais se aproxime dos valores ótimos é utilizando uma abordagem que contemple os ótimos aspectos de roteamento encontrados no algoritmo das economias e os aspectos de agrupamento encontrados no algoritmo de varredura.

Assim, a terceira etapa consistiu do desenvolvimento de abordagens que contemplassem as características de resolução do PRV para prover maior eficiência à resolução. Para tanto,

criou-se neste trabalho o algoritmo *JoinRoutes* com o intuito de aprimorar os aspectos de melhoramento das rotas construídas. Tal algoritmo age principalmente na modificação das cardinalidades das rotas de tal forma que estas ocupem regiões distintas do espaço. Ainda no sentido de investigar métodos que alcançassem melhores resultados, foi implementado uma versão do algoritmo $D-Ants$, proposto por Reimman et al. (REIMANN; DOERNER; HARTL, 2004), pois este une os aspectos da heurística das economias e do algoritmo de varredura em um mesmo procedimento. Além disto, tal método é inspirado na meta-heurística *Ant Colony* que incorpora os aspectos de aleatoriedade e aprendizado computacional nas etapas construtivas.

A última etapa consistiu da análise da aplicação dos métodos desenvolvidos ao conjunto de instâncias presentes na literatura. Tal análise objetivou verificar quais abordagens apresentavam melhores resultados computacionais quanto ao custo atingido e quanto ao tempo de execução requerido. Notou-se que a aplicação das heurísticas apresenta capacidade de exploração do espaço de soluções restrita, pois cada heurística foi desenvolvida com um propósito específico de busca. A utilização de métodos de busca local aplicados às heurísticas resulta em melhorias significativas, porém suas combinações nem sempre são atrativas computacionalmente, pois há um grande consumo dos recursos para um pequeno melhoramento. A utilização dos métodos combinados deve ser considerada para a utilização do PDOS, já que este é um problema de natureza real e que contempla um cenário onde há um número extremamente grande de ordens de serviço a serem roteadas. Por último, nota-se que a utilização do procedimento $D-Ants$ apresentou as melhores médias de resultados para todas as instâncias. Tais resultados foram atingidos pois há uma sintonia na utilização das heurísticas das economias e da de varredura em conjunto com os princípios de aleatoriedade e aquisição de conhecimento provido pela meta-heurística $AntColony$.

Desta forma, as abordagens mais promissoras de serem aplicáveis ao PDOS são a meta-heurística $D-Ants$ e os procedimentos construtivos com buscas locais combinadas. Tais abordagens apresentaram custos relativos a função objetivo próximos aos ótimos conhecidos. Quanto ao tempo de execução, a meta-heurística $D-Ants$ apresentou uma grande disparidade em relação as demais abordagens.

Como trabalhos futuros podem-se citar: alterar as restrições dos algoritmos desenvolvidos para suportar as restrições ainda não contempladas do PDOS, analisar a aplicação das melhores abordagens estudas ao problema de despacho de ordens de serviço, conciliar as técnicas estudas do PRV com as técnicas do problema de agrupamento capacitado e analisar a complexidade das etapas do algoritmo proposto *JoinRoutes* com a finalidade de otimizar o seu tempo de execução.

Referências Bibliográficas

AARDAL, K.; NEMHAUSER, G. L.; WEISMANTEL, R. *Handbook of discrete optimization*. Amsterdam: Elsevier, 2005.

AUGERAT, P. et al. *Computational results with a branch and cut code for the capactated vehicle routing problem*. [S.l.], 1995.

BEASLEY, J. E. Route-first cluster-second methods for vehicle routing. *Omega*, v. 11, p. 403–408, 1983.

BLASUM, U.; HOCHSTÄTTLER, W. *Application of the branch and cut method to the vehicle routing problem*. [S.l.], 2000.

BULLNHEIMER, B.; HARTL, R. F.; STRAUSS, C. An improved ant system for the vehicle. *Annals of Operations Research*, v. 28, n. 459, p. 89–319, 1999.

CHRISTOFIDES, N.; MINGOZZI, A.; TOTH, P. *Combinatorial Optimization*. [S.l.]: John Wiley, 1979.

CLARKE, G.; WRIGHT, J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, v. 12, n. 568-581, 1964.

CORDEAU, J. F.; GENDREAU, M.; LAPORTE, G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *John Wiley & Sons*, 1997.

CORMEN, T. H. et al. *Algoritmos: Teoria e Prática*. [S.l.]: Elsevier, 2002.

DANTZIG, G.; FULKERSON, R.; JOHNSON, S. Solution of a large-scale travelling salesman problem. *Operations Research Society*, 1954.

DANTZIG, G. B.; RAMSER, J. The truck dispatching problem. *Management Science*, v. 6, p. 80–81, 1959.

DORIGO, M.; GAMBARDELLA, L. M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation IEEE Transactions*, v. 1, p. 53–66, 1997.

FISCHETTI, M.; TOTH, P.; VIGO, D. A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Operations research*, v. 42, p. 846–859, 1994.

FISHER, M. L. Optimal solution of vehicle routing problems using minimum k-trees. *Operations research*, v. 42, p. 626–642, 1994.

FISHER, M. L.; JAIKUMAR, R. A generalized assignment heuristic for vehicle routing. *Networks*, v. 11, n. 109-124, 1981.

GILLET, B. E.; MILLER, L. R. A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, v. 22, n. 341-349, 1974.

GLOVER, F.; KOCHENBERGER, G. A. *Handbook of metaheuristics*. [S.l.]: Kluwer Academic Publishers, 2003. ISBN 1-4020-7263-5.

GLOVER, F.; LAGUNA, M. *Tabu Search*. [S.l.]: Kluwer Academic Publisher, 1997.

GOLDBARG, M. C.; LUNA, H. P. L. *Otimização Combinatória e Programação Linear*. [S.l.]: Elsevier, 2005.

GOLDBERG, E. D. Genetic algorithm in search. *Optimization and Machine Learning*, v. 24, p. 41–49, 1989.

LAPORTE, G. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, n. 59, p. 345–358, May 1992.

LAPORTE, G.; NOBERT, Y. Exact algorithms for the vehicle routing problem. *Surveys in Combinatorial Optimization*, p. 147–184, 1987.

LAWLER, E. L. et al. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. New York: Wiley and Sons, 1985.

LENSTRA, J. K.; KAN, A. H. G. R. Complexity of vehicle routing and scheduling problems. *Networks*, v. 11, p. 221–227, 1981.

LIN, S. Computer solutions of the traveling salesman problem. *Bell Syst. Tec*, v. 44, p. 2245–2269, 1965.

LIN, S.; KERNIGHAN, B. W. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, n. 21, p. 498–516, 1973.

LUENBERGER, D. G.; YE, Y. *Linear and Nonlinear Programming*. [S.l.]: Springer, 2008.

MILLER, D. L. A matching based exact algorithm for capacitated vehicle routing problems. *ORSA Journal on Computing*, p. 1–9, 1995.

MOLE, R. H.; JAMESON, S. R. A sequential route-building algorithm employing a generalized saving criterion. *Operational Research Quarterly*, v. 27, p. 503–511, 1976.

OSMAN, I.; CHRISTOFIDES, N. Capacitated clustering problems by hybrid simulated annealing and tabu search. *International Transactions in Operational Research*, Blackwell Publishing, v. 1, n. 3, p. 317–336, 1994.

REIMANN, M.; DOERNER, K.; HARTL, R. F. D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers and operations research*, v. 31, p. 563–591, 2004.

RENAULD, J.; BOCTOR, F. F. A sweep-based algorithm for the fleet size and mix vehicle routing problem. *European Journal of Operational Research*, v. 140, p. 618–628, 2002.

RESENDE, T. A. F. e M. G. C. Greedy randomized adaptive search procedures. *Journal of Global Optimiation*, v. 6, p. 109–133, 1995.

ROSENKRANTZ, D. J.; STEARNS, R. E.; LEWIS, P. M. An analysis of several heuristics for the traveling salesman problem. *Society for Industrial and Applied Mathematics*, v. 6, p. 563–581, 1977.

RYAN, D. M.; HJORRING, C.; GLOVER, F. Extensions of the petal method for vehicle routing. *Journal of the Operational Research Society*, v. 44, p. 289–296, 1993.

S. Kirkpatrick, C. D. G. J. M. P. V. Optimization by simulated annealing. *Science*, v. 220, n. 4598, p. 671–680, 1983.

TAHA, H. A. *Operation Research: An introduction*. 8. ed. [S.l.]: Pearson Prentice Hall, 2007. ISBN 0-13-188923-0.

TAILLARD, É. Heuristic methods for large centroid clustering problems. *Journal of Heuristics*, Springer, v. 9, n. 1, p. 51–73, 2003.

TOTH, P.; VIGO, D. *The Vehicle Routing Problem*. [S.l.]: Society for Industrial and Applied Mathematics, 2002.