CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
Diretoria de Pesquisa e Pós-Graduação
Programa de Mestrado em Modelagem Matemática e Computacional

ESTUDO DO PROBLEMA DE ROTEAMENTO ABERTO DE VEÍCULOS COM JANELAS DE TEMPO UTILIZANDO ALGORITMOS COLÔNIA DE FORMIGA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional, como parte dos requisitos exigidos para a obtenção do título de Mestre em Modelagem Matemática e Computacional.

Aluno : José Maurício Costa

Orientador : Prof. Dr. Sérgio Ricardo de Souza

Co-Orientador : Prof. Dr. Marcone Jamilson Freitas Souza

Belo Horizonte - MG
Agosto de 2012

Costa, José Maurício
M931a ESTUDO DO PROBLEMA DE ROTEAMENTO ABERTO DE VEÍCULOS COM JANELAS DE TEMPO UTILIZANDO ALGORITMOS COLÔNIA DE FORMIGA / José Maurício Costa. – Belo Horizonte, 2012.
86p.

Dissertação (Mestrado) – Centro Federal de Educação Tecnológica de Minas Gerais
Programa de Pós-Graduação em Modelagem Matemática e Computacional
Orientador: Prof. Dr. Sérgio Ricardo de Souza
Co-orientador: Prof. Dr. Marcone Jamilson Freitas Souza

1. Pesquisa Operacional - Teses. 2. Colônia de Formigas - Teses. 3. Metaheurísticas. 4. Computação.
I. Souza, Sérgio Ricardo de. II. Souza, Marcone Jamilson Freitas.
III. Centro Federal de Educação Tecnológica de Minas Gerais.
IV. Título.

CDD: 006.3

Dedico ao meu pai, a minha mãe e ao meu irmão, que são as pessoas mais importantes na minha vida:
Adauto, Vilani e Armando.

Agradecimentos

Este trabalho representa a conclusão de um sonho que apenas foi possível graças à presença daqueles que fizeram parte de cada momento que eu vivi para poder realizá-lo. Por isso, agradeço de coração a todos vocês.

Primeiramente, agradeço a Deus por estar sempre ao meu lado, iluminado e guiando os meus passos para que eu possa seguir a minha com a sua graça.

Aos meus pais, Adauto e Vilani, por todo o seu amor, carinho e amizade incondicionais. Vocês são os principais responsáveis por eu ter me tornado a pessoa que sou hoje, por meio do seu exemplo e de seus ensinamentos, sempre me mostrando que a humildade, fé, honestidade, educação e trabalho são coisas que sempre devemos ter em nossas vidas. Agradeço também ao meu irmão por ter me incentivado a continuar arriscando pelos meus sonhos.

Ao meu orientador, professor Sérgio Ricardo de Souza, por ter me aceitado como seu aluno e acreditar no meu potencial. Além disso, quero também agradecê-lo por estar sempre presente e preocupado conosco, como um "pai zeloso", mostrando que somos mais capazes do imaginamos.

Ao meu orientador, professor Marcone Jamilson Freitas Souza, por ter me mostrado o mundo das "metaheurísticas" com a sua empolgação contagiante, nos estimulando a pesquisar. Quero agradecê-lo também pelos seus ensinamentos e suas dicas que de forma direta ou indireta contribuíram em muito para com o meu aprendizado e desenvolvimento desta pesquisa.

Agradeço os meus amigos do CEFET-MG que sempre estiveram ao meu lado, nos momentos bons e difíceis, e que tive a benção de poder conhecê-los. Posso dizer que para muitos deles como o Nilmar, Flaviana, Juliana, Alline, Abelardo, Lillia, Carol, Saulo, Jeanderson, Harley, Breno, Bruno, Eduardo, Renan, Herondino e Camila eu tenho grande consideração, amizade e admiração.

Nunca me esquecerei dos gestos de bondade demonstrados por Nilmar e Gisele em me deixarem estudar com eles e tirarem sempre as minhas dúvidas, Flaviana ao ser umas das primeiras pessoas em Belo Horizonte a estenderem a mão para me ajudar, Saulão pelas dicas de programação, Renan com as suas palavras de incentivo e pela ajuda ao me indicar um local para morar, Dayanne devidos aos conselhos e materiais de estudo que me forneceu, Marcelus por suas idéias que contribuíram para com a pesquisa e Abelardo pelo apoio moral e consultoria em português.

Agradeço também a Lenize pela amizade, apoio e paciência que teve comigo quando eu ia na coordenação.

Agradeço a Sabrina que me ajudou em muito durante o desenvolvimento desta pesquisa, sendo muito gentil e prestativa, me orientando sobre o funcionamento dos

algoritmos baseados na metaheurística Colônia de Formigas.

Agradeço ao pessoal do CCC pelo suporte, me ajudando na utilização dos computadores para a realização dos experimentos.

Agradeço aos meus amigos que fiz durante a época que morei em Arcos e Rio Paranaíba que, apesar da distância, sei que estão sempre torcendo por mim. Destaco a ajuda dada pelo meu amigo Roberto que me ajudou a me mudar para Arcos e depois para Belo Horizonte, contribuindo para a realização do meu sonho em poder estudar.

Agradeço também ao CEFET e a CAPES, pelo apoio financeiro.

"Não tentes ser bem sucedido, tenta antes ser um homem de valor."
Albert Einstein
"Calma, tudo tem o seu tempo."
Vilani Abadia de Babos Costa

Resumo

Neste trabalho, é proposto um estudo acerca do Problema de Roteamento Aberto de Veículos com Janelas de Tempo (PRAVJT). O PRAVJT é uma variação do Problema de Roteamento de Veículos em que o veículo não é obrigado a retornar para o depósito após servir o último cliente que pertence à rota percorrida por ele. No PRAVJT, cada rota é definida como uma sequência de clientes, que inicia no depósito e termina em um dos clientes. O objetivo do PRAVJT é minimizar o número de veículos necessários para se percorrer todas as rotas, além de minimizar a distância percorrida e o tempo total gastos pelos veículos para atender todos os clientes de suas respectivas rotas. Esta dissertação apresenta a solução de instâncias desse problema através das hibridização das metaheurísticas *Max-Min Ant System*, *Population-Based Ant Colony Optimization*, ILS (*Iterated Local Search*) e GRASP (*Greedy Randomized Adaptive Search Procedure*). São utilizados 10 movimentos de vizinhança, na forma de movimentos intra e inter-rotas e de eliminação de rotas. As metaheurísticas desenvolvidas são testadas utilizando as instâncias de Solomon e de Homberger-Gehring, adaptadas para o caso de roteamento aberto. Os resultados computacionais obtidos são analisados utilizando o Teste Kruskal-Wallis.

Palavras-chave: Problema de Roteamento Aberto de Veículos com Janelas de Tempo. Metaheurísticas. Métodos de Colônia de Formigas.

Abstract

This paper proposes a study of the Open Routing Problem with Time Windows Vehicle (OVRPTW). The OVRPTW is a variation of the Vehicle Routing Problem in which the vehicle is not required to return the deposit, after serving the last customer who belongs to the route taken by him. In the OVRPTW, each route is defined as a sequence of customers in the warehouse that starts and ends in one of the clients. The objective of OVRPTW is to minimize the number of vehicles needed to cover all routes, while minimizing the total distance and time spent by vehicles to suit all customers of their respective routes. It is hoped, through this work was to study the application of different metaheuristics in order to solve the OVRPTW, finding new solutions for the same. Thus, were developed and applied different algorithms for solving the PRAVJT in which instances are used in the literature for testing the performance. Furthermore, the results by algorithms developed were analyzed for the validation.

Keywords: Open Vehicle Routing Problem with Time Windows. Combinatory Optimization. Metaheuristics.

Sumário

1 Introdução **1**
- 1.1 Justificativas . . . 2
- 1.2 Objetivos . . . 3
  - 1.2.1 Objetivo Geral . . . 3
- 1.3 Metodologia Proposta . . . 3
- 1.4 Organização do Trabalho . . . 4

2 Caracterização do Problema **5**
- 2.1 Problema de Roteamento de Veículos . . . 5
- 2.2 Problema de Roteamento Aberto de Veículos . . . 6
- 2.3 Problema de Roteamento Aberto de Veículos com Janela de Tempo . . . 7
- 2.4 Trabalhos Relacionados ao PRAV . . . 8
- 2.5 Modelos Matemáticos para PRAV . . . 12
  - 2.5.1 Modelo Matemático de Toth e Vigo (PRV) . . . 12
  - 2.5.2 Modelo Matemático de Tan (PRVJT) . . . 14
  - 2.5.3 Modelo Matemático de MirHassani e Abolghasemi (PRAV) . . . 15
  - 2.5.4 Modelo Matemático de Yu (PRAV) . . . 17
  - 2.5.5 Modelo Matemático de Guiyun (PRAVJT) . . . 18
  - 2.5.6 Modelo Matemático de Repoussis (PRAVJT) . . . 20
  - 2.5.7 Comparação entre os Modelos Matemáticos . . . 24

3 Heurísticas Estudadas **25**
- 3.1 Heurísticas Construtivas . . . 25
  - 3.1.1 *Push Forward Insertion Heuristic* . . . 26
  - 3.1.2 Heurística Sorteio da Copa do Mundo . . . 28
- 3.2 Heurísticas de Busca Local . . . 30
  - 3.2.1 Método de Primeira Melhora . . . 30
  - 3.2.2 Método de Descida Randômica . . . 31
- 3.3 Metaheurísticas . . . 31
  - 3.3.1 *Iterated Local Search* . . . 32
  - 3.3.2 *Greedy Randomized Adaptative Search Procedure* . . . 33
  - 3.3.3 *Variable Neighborhood Descent* . . . 35
- 3.4 Colônia de Formigas . . . 36
  - 3.4.1 Metaheurística *Ant System* . . . 37
  - 3.4.2 *MAX-MIN Ant System* . . . 40
  - 3.4.3 *Population-Based Ant Colony Optimization* . . . 44

4 Algoritmos Desenvolvidos **47**
4.1 Representação de Uma Solução . . . 47
4.2 Estruturas de Vizinhança . . . 47
4.2.1 Movimentos das Estruturas de Vizinhança *Shift'*(k) e *Exchange* 47
4.2.2 Movimentos da Estrutura de Vizinhança *Shift*(k,0) . . . 49
4.2.3 Movimentos da Estrutura de Vizinhança *Swap*(k,l) . . . 51
4.2.4 Estratégias de Eliminação de Rotas . . . 52
4.3 Função de Avaliação . . . 55
4.4 Algoritmos Propostos . . . 55
4.4.1 ILS . . . 55
4.4.2 GRASP+ILS . . . 57
4.4.3 HSCM+ILS . . . 58
4.4.4 *MAX-MIN Ant System*+ILS . . . 59
4.4.5 P-ACO+ILS . . . 60

5 Experimentos Computacionais **66**
5.1 Instâncias Propostas para o PRAVJT . . . 66
5.2 Definições para os Resultados Computacionais . . . 67
5.3 Resultados para o ILS . . . 68
5.4 Resultados para o GRASP+ILS . . . 71
5.5 Resultados para o HSCM+ILS . . . 74
5.6 Resultados para o *MAX-MIN Ant System*+ILS . . . 76
5.6.1 Resultados para as Instâncias com 200 Clientes . . . 79
5.6.2 Resultados para as Instâncias com 400 Clientes . . . 79
5.6.3 Resultados para as Instâncias com 600 Clientes . . . 81
5.7 Resultados para o P-ACO . . . 82
5.7.1 Resultados para o P-ACO *Age-based Strategy*+ILS . . . 82
5.7.2 Resultados para o P-ACO *Quality-based Strategy*+ILS . . . 85
5.7.3 Resultados para o P-ACO *Elitist-based Strategy*+ILS . . . 88

6 Análise dos Experimentos Computacionais **91**
6.1 Desempenho Computacional . . . 91
6.2 Dados Gerais das Soluções . . . 99
6.2.1 Geração de Novas Soluções . . . 99
6.2.2 Sumário de Resultados . . . 101
6.3 Análise Estatística e Teste Kruskal-Wallis . . . 102
6.3.1 Apresentação do Teste Kruskal-Wallis . . . 102
6.3.2 Aplicação do Teste Kruskal-Wallis . . . 103
6.4 Análise dos Resultados através de Gráficos BoxPlot . . . 105

7 Considerações Finais **111**
7.1 Publicações Originárias desta Dissertação . . . 113
7.2 Trabalhos Futuros . . . 114

Referências **115**

Lista de Tabelas

2.1 Trabalhos relacionados ao PRAV. 12

4.1 Valores dos parâmetros definidos para as estratégias do P-ACO. . . . 61

5.1 Informações das instâncias de Solomon 67
5.2 Exemplo de uma instância para o PRAVJT 67
5.3 Resultados para o conjunto de instâncias R1 de Solomon 69
5.4 Resultados para o conjunto de instâncias C1 de Solomon 69
5.5 Resultados para o conjunto de instâncias RC1 de Solomon 70
5.6 Resultados para o conjunto de instâncias R2 de Solomon 70
5.7 Resultados para o conjunto de instâncias C2 de Solomon 70
5.8 Resultados para o conjunto de instâncias RC2 de Solomon 71
5.9 Resultados para o conjunto de instâncias R1 de Solomon 71
5.10 Resultados para o conjunto de instâncias C1 de Solomon 72
5.11 Resultados para o conjunto de instâncias RC1 de Solomon 72
5.12 Resultados para o conjunto de instâncias R2 de Solomon 72
5.13 Resultados para o conjunto de instâncias C2 de Solomon 73
5.14 Resultados para o conjunto de instâncias RC2 de Solomon 73
5.15 Resultados para o conjunto de instâncias R1 de Solomon 74
5.16 Resultados para o conjunto de instâncias C1 de Solomon 74
5.17 Resultados para o conjunto de instâncias RC1 de Solomon 75
5.18 Resultados para o conjunto de instâncias R2 de Solomon 75
5.19 Resultados para o conjunto de instâncias C2 de Solomon 76
5.20 Resultados para o conjunto de instâncias RC2 de Solomon 76
5.21 Resultados para o conjunto de instâncias R1 de Solomon 77
5.22 Resultados para o conjunto de instâncias C1 de Solomon 77
5.23 Resultados para o conjunto de instâncias RC1 de Solomon 77
5.24 Resultados para o conjunto de instâncias R2 de Solomon 78
5.25 Resultados para o conjunto de instâncias C2 de Solomon 78
5.26 Resultados para o conjunto de instâncias RC2 de Solomon 78
5.27 Resultados para as instâncias do conjunto R1 com 200 clientes 79
5.28 Resultados para as instâncias do conjunto C1 com 200 clientes 79
5.29 Resultados para as instâncias do conjunto RC1 com 200 clientes . . . 80
5.30 Resultados para as instâncias do conjunto R1 com 400 clientes 80
5.31 Resultados para as instâncias do conjunto C1 com 400 clientes 80
5.32 Resultados para as instâncias do conjunto RC1 com 400 clientes . . . 81
5.33 Resultados para as instâncias do conjunto R1 com 600 clientes 81
5.34 Resultados para as instâncias do conjunto C1 com 600 clientes 82

5.35 Resultados para as instâncias do conjunto RC1 com 600 clientes . . . 82
5.36 Resultados para o conjunto de instâncias R1 de Solomon 83
5.37 Resultados para o conjunto de instâncias C1 de Solomon 83
5.38 Resultados para o conjunto de instâncias RC1 de Solomon 84
5.39 Resultados para o conjunto de instâncias R2 de Solomon 84
5.40 Resultados para o conjunto de instâncias C2 de Solomon 85
5.41 Resultados para o conjunto de instâncias RC2 de Solomon 85
5.42 Resultados para o conjunto de instâncias R1 de Solomon 86
5.43 Resultados para o conjunto de instâncias C1 de Solomon 86
5.44 Resultados para o conjunto de instâncias RC1 de Solomon 86
5.45 Resultados para o conjunto de instâncias R2 de Solomon 87
5.46 Resultados para o conjunto de instâncias C2 de Solomon 87
5.47 Resultados para o conjunto de instâncias RC2 de Solomon 87
5.48 Resultados para o conjunto de instâncias R1 de Solomon 88
5.49 Resultados para o conjunto de instâncias C1 de Solomon 88
5.50 Resultados para o conjunto de instâncias RC1 de Solomon 89
5.51 Resultados para o conjunto de instâncias R2 de Solomon 89
5.52 Resultados para o conjunto de instâncias C2 de Solomon 90
5.53 Resultados para o conjunto de instâncias RC2 de Solomon 90

6.1 Resultados encontrados para as instâncias escolhidas para Testes de Probabilidade Empíric
6.2 Dados das soluções alvo. 92
6.3 Dados Gerais dos Resultados paras os Conjuntos R1, C1 e RC1 . . . 101
6.4 Dados Gerais dos Resultados paras os Conjuntos R2, C2 e RC2 . . . 101
6.5 Dados Gerais dos Resultados 101
6.6 Notação usada no Teste de Kruskal-Wallis. 102
6.7 Legendas. 103
6.8 Resultados para o Teste de Kruskal-Wallis 104
6.9 Resultados para o Teste de Kruskal-Wallis 104

Lista de Figuras

2.1 Caracterização de Roteamento Aberto de Veículos. 7

3.1 Inserção das cidades por meio do PFIH. 27
3.2 Pontes de tamanhos iguais. 36
3.3 Pontes de tamanhos diferentes. 37
3.4 Representação do processo de escolha da cidade pela formiga. 38

4.1 Movimento *Shift*'(1). 48
4.2 Movimento *Shift*'(2). 48
4.3 Movimento *Shift*'(3). 49
4.4 Movimento *Exchange*. 49
4.5 Movimento *Shift*(1,0). 50
4.6 Movimento *Shift*(2,0). 50
4.7 Movimento *Shift*(3,0). 51
4.8 Movimento *Swap*(1,1). 51
4.9 Movimento *Swap*(2,1). 52
4.10 Movimento *Swap*(2,2). 52
4.11 Retirada de uma rota da solução. 54
4.12 Estratégia Elimina Rota. 54
4.13 Estratégia Elimina Rota Função Objetivo. 55

6.1 Teste de Probabilidade Empírica para a instância R102. 93
6.2 Time-to-target para a instância C104. 94
6.3 Time-to-target para a instância RC104. 95
6.4 Time-to-target para a instância R205. 96
6.5 Time-to-target para a instância C204. 97
6.6 Time-to-target para a instância RC204. 98
6.7 Percentual de soluções novas encontradas para os conjuntos R1 e R2. 99
6.8 Percentual de novas soluções encontradas para os conjuntos C1 e C2. 100
6.9 Percentual de novas soluções encontradas para os conjuntos RC1 e RC2. 100
6.10 Gráficos BoxPlot acerca da distância para o conjunto R102. 105
6.11 Gráficos BoxPlot acerca da distância para o conjunto C104. 106
6.12 Gráficos BoxPlot acerca da distância para o conjunto RC104. 107
6.13 Gráficos BoxPlot acerca da distância para o conjunto R205. 108
6.14 Gráficos BoxPlot acerca da distância para o conjunto C204. 109
6.15 Gráficos BoxPlot acerca da distância para o conjunto RC204. 110

Lista de Algoritmos

1 ConstrucaoGulosa . . . 26
2 ConstrucaoAleatoria . . . 26
3 HSCM . . . 29
4 Construcao . . . 29
5 PrimeiraMelhora . . . 31
6 DescidaRandomica . . . 31
7 ILS . . . 33
8 GRASP . . . 33
9 Construcao . . . 34
10 BuscaLocal . . . 34
11 VND . . . 35
12 Colônia de Formigas . . . 40
13 MAX-MIN Ant System . . . 43
14 ILS . . . 57
15 GRASP . . . 58
16 GRASP+ILS . . . 58
17 HSCM+ILS . . . 58
18 MAX-MIN Ant System+ILS . . . 59
19 MAX-MIN Ant System . . . 60
20 $P - ACO_{Age}$+ILS . . . 61
21 $P - ACO_{Age}$ . . . 62
22 $P - ACO_{Quality}$+ILS . . . 62
23 $P - ACO_{Quality}$ . . . 63
24 $P - ACO_{Elitist}$ . . . 64
25 $P - ACO_{Elitist}$+ILS . . . 65

Capítulo 1

Introdução

No contexto econômico atual, os custos relativos ao transporte de bens compõem uma grande parcela dos gastos de uma empresa. Várias companhias cada vez mais contratam serviços de transporte terceirizados para realizar a entrega de seus produtos para os seus clientes. Isso acontece pois, por meio da contratação de frotas de veículos de entrega, é possível obter uma significativa redução em gastos referentes aos serviços de transporte. Para Repoussis et al. (2006), essa contratação se justifica mesmo em casos em que o custo do aluguel de frotas de veículos seja mais caro pela distância percorrida, devido à economia com a manutenção da frota, por exemplo, dentre outros gastos. Ao contratar outra empresa para realizar as entregas de seus produtos, as companhias pagam pelo serviço conforme a distância percorrida por cada um dos veículos alugados. Desse modo, não há preocupação por parte da empresa se o veículo utilizado para a entrega retornará ou não para o ponto de partida.

Muitas empresas, como as de laticínios, contratam frotas de veículos para a entrega de leite em diferentes pontos; outras, como as editoras de jornais, também fretam veículos para a entrega de jornais em residências. Em ambos os casos, tais empresas não pagam pela viagem feita pelos veículos entre o último cliente atendido e o ponto de partida da rota, ou seja, o percurso de retorno ao depósito. Outra situação que possui estrutura semelhante à dos casos citados é a de roteamento de ônibus escolares. Casos como esses caracterizam o Problema de Roteamento Aberto de Veículos (PRAV), que é uma variante do Problema de Roteamento de Veículos (PRV) clássico.

De acordo com Li et al. (2007), no PRAV, o veículo não é obrigado a retornar para o depósito após servir o último cliente que faz parte da rota percorrida por ele. Mas, se isso ocorrer, ele deve fazer a viagem no caminho inverso. Assim, diferentemente do PRV, em que as rotas percorridas pelos veículos representam um ciclo hamiltoniano, no PRAV, as rotas são um caminho hamiltoniano.

Para Brandão (2004), o Problema de Roteamento Aberto de Veículos consiste em definir as melhores rotas para uma frota de veículos, que, por sua vez, devem atender um conjunto de clientes que possuem determinada demanda e localização geográfica conhecidas. Com isso, cada rota é definida como uma sequência de clientes, que inicia no depósito e termina em um dos clientes. Apesar de não haver a necessidade de retorno ao depósito nesta variação do PRV, o PRAV é considerado um problema de otimização combinatória NP-difícil, pois, para solucioná-lo, é preciso encontrar o melhor caminho hamiltoniano para cada conjunto de clientes atribuídos à rota de um veículo. Esse subproblema pode ser considerado como NP-difícil, pois pode

ser convertido em um ciclo hamiltoniano equivalente; deste modo, o problema como um todo é também NP-difícil (Brandão, 2004). De acordo com Ziviani (2004), um caminho hamiltoniano é uma sequência de vértices e arestas alternados, de maneira que cada aresta é incidente ao nó anterior e posterior de um grafo em que o vértice inicial é diferente do final. Mas, quando o vértice inicial e o final são o mesmo, o caminho passa a ser um ciclo hamiltoniano. Desta forma, a utilização de heurísticas e metaheurísticas para sua solução se justificam. Brandão (2004) descreve o PRAV como um problema em que todos os veículos possuem a mesma capacidade; o tempo de viagem de cada veículo não pode ultrapassar um dado limite; a demanda total de todos os clientes pertencentes a uma rota não pode ultrapassar a capacidade do veículo; e cada cliente é visitado apenas uma vez por um dos veículos, de forma que a sua demanda seja completamente atendida. O objetivo do problema, caracterizado dessa maneira, é o de minimizar o número de veículos necessários para percorrer todas as rotas, além de minimizar a distância e o tempo total gastos pelos veículos para atender todos os clientes de suas respectivas rotas.

Conforme Li et al. (2009), apesar de ser fortemente adequado para modelar problemas de roteamento do mundo real, o PRAV não tem recebido a mesma atenção que o PRV. Nesta dissertação, é feito o estudo de uma variante do PRAV, que é o Problema de Roteamento Aberto de Veículos com Janelas de Tempo (PRAVJT), em que se considera a existência de períodos de tempo para o atendimento aos clientes.

1.1 Justificativas

Brandão (2004) diz que, mesmo não havendo a necessidade dos veículos retornarem para o depósito, como no PRV, o PRAV é um problema de otimização combinatória NP-difícil, ou seja, possui ordem de complexidade exponencial. Problemas como este possuem um grande espaço de busca para se fazer a pesquisa por soluções, fazendo com que não seja adequado o uso de métodos computacionais exatos para se obter soluções ótimas. Desse modo, são utilizadas heurísticas e metaheurísticas que, apesar de não proporcionarem a garantia de que a solução encontrada seja ótima, permitem a obtenção de soluções aproximadas com custos de tempo e processamento viáveis.

Segundo Li et al. (2009), o PRAV é um problema de grande importância na área logística, sendo próprio para modelar muitos problemas de roteamento do mundo real, possuindo importante aplicação prática. Mas ele não tem sido tão pesquisado quanto o PRV, fazendo com que seja relativamente pequeno o número de trabalhos referentes a ele de forma específica na literatura. Para Brandão (2004), apesar de terem sido desenvolvidos bons algoritmos para solucionar o PRV, os mesmos não podem ser simplesmente utilizados para resolver o PRAV, de modo que o trajeto entre o último cliente de cada rota e o depósito adjacente da solução final seja retirado. Esta medida não é adequada, pois o tempo computacional e a qualidade das soluções construídas geralmente são piores, quando não são utilizados algoritmos específicos para o PRAV. Isto ocorre porque, quando algoritmos que foram desenvolvidos para o PRV são aplicados diretamente ao PRAV deixam de serem levadas em conta as características específicas do problema, como a restrição que impede o retorno dos veículos ao depósito, por exemplo. Desta forma, uma solução ótima para o PRV pode ser uma

solução inadequada para o PRAV. Se restrições como a de tamanho máximo da rota são consideradas, o uso de algoritmos não adaptados para o PRAV não é apropriado, porque uma solução que é infactível para o PRV pode ser factível para o PRAV. Assim, devido à complexidade do PRAV, torna-se necessário o desenvolvimento de novos algoritmos para a resolução deste tipo de problema.

1.2 Objetivos

1.2.1 Objetivo Geral

Esta dissertação tem, como objetivo geral, estudar a aplicação de algoritmos baseados nas metaheurísticas Colônia de Formigas, GRASP e ILS ao Problema de Roteamento Aberto de Veículos com Janelas de Tempo.

Os objetivos específicos são:

- Realização de uma revisão bibliográfica acerca do Problema de Roteamento Aberto de Veículos com Janelas de Tempo e do uso de metaheurísticas para a resolução do mesmo;
- Desenvolvimento algoritmos baseados nas metaheurísticas Colônia de Formigas, GRASP e ILS para a resolução do PRAVJT;
- Aplicação de análise estatística para avaliar o desempenho dos algoritmos desenvolvidos.

1.3 Metodologia Proposta

Esta dissertação tem como objetivo realizar um estudo acerca da aplicação de metaheurísticas para a resolução do PRAVJT. Deste modo, primeiramente, é feita uma revisão bibliográfica sobre o tema.

Para a solução do problema em análise, são desenvolvidos diferentes algoritmos híbridos adaptados para o PRAVJT, baseados nas metaheurísticas Colônia de Formigas, GRASP e ILS. Os algoritmos são implementados em linguagem C++, utilizando o compilador gnu GCC.

Como não existem instâncias na literatura específicas para o PRAVJT, são utilizados, assim como em Repoussis et al. (2006), de maneira adaptada, os conjuntos de instâncias propostos por Solomon (1987) para o Problema de Roteamento de Veículos com Janela de Tempo, contendo 100 clientes. Também são utilizadas, de forma adaptada, os conjuntos de instâncias apresentadas em Homberger e Gehring (1999). As instâncias de Homberger e Gehring possuem características semelhantes às de Solomon, porém, são formadas por uma quantidade maior de clientes. Foram utilizadas, neste caso, somente as instâncias com 200, 400 e 600 clientes.

Com a finalidade de testar a eficiência dos métodos desenvolvidos, são feitos experimentos computacionais, que consistem na aplicação destes métodos sobre as instâncias propostas. Os resultados encontrados são comparados com os existentes na literatura, relativos às mesmas instâncias utilizadas, de modo a validar a qualidade

do que foi implementado. Além disso, é realizada uma análise estatística, para efetivamente demonstrar a qualidade dos resultados.

1.4 Organização do Trabalho

Esta dissertação está organizada da seguinte maneira: no Capítulo 1 é feita uma introdução do PRAVJT e realizada uma breve descrição sobre o mesmo, bem como apresentada a justificativa, os objetivos gerais e específicos e a metodologia adotada referentes ao presente trabalho. No Capítulo 2, são descritas as principais características do PRAVJT, trabalhos relacionados a este problema de roteamento e modelos matemáticos referentes ao mesmo e suas variantes. No Capítulo 3, são apresentadas as heurísticas utilizadas para a solução do problema em estudo. O Capítulo 4 apresenta os algoritmos desenvolvidos para a solução do PRAVJT e o Capítulo 5 mostra os resultados encontrados a partir da aplicação destes algoritmos às instâncias de teste. No Capítulo 6 é feita uma análise dos resultados encontrados, utilizando procedimentos estatísticos. O Capítulo 7 finaliza a dissertação, mostrando as conclusões gerais acerca do trabalho realizado e apontando para trabalhos futuros.

Capítulo 2

Caracterização do Problema

Na primeira seção deste Capítulo é feita uma descrição do Problema de Roteamento de Veículos (PRV). Na Seção 2.2 é descrito o Problema de Roteamento Aberto de Veículos (PRAV) e sua variante objeto de interesse da presente dissertação, o Problema de Roteamento Aberto de Veículos com Janelas de Tempo. Este problema possui as mesmas restrições que o PRAV, porém, com a adição de restrições referentes às janelas de tempo que o compõem. Na Seção 2.4 é feita uma apresentação de trabalhos da literatura relacionados ao problema estudado. Na Seção 2.5 são realizadas descrições da formulação matemática para o Problema de Roteamento de Veículos; para o Problema de Roteamento de Veículos com Janelas de Tempo; para o Problema de Roteamento Aberto de Veículos; e para o Problema de Roteamento Aberto de Veículos com Janelas de Tempo. A Seção 2.5 mostra uma comparação entre os modelos matemáticos apresentados.

2.1 Problema de Roteamento de Veículos

De acordo com Toth e Vigo (2002), o Problema de Roteamento de Veículos (PRV) é um problema de otimização combinatória NP-difícil, que consiste na utilização de uma frota de veículos para realizar o serviço de entrega de mercadorias para todo um conjunto de clientes, de forma que seja minimizado o número de veículos usados e a distância total percorrida por eles. No PRV, cada cliente possui uma demanda e uma localização geográfica conhecidas. A frota de veículos é homogênea, ou seja, todos possuem uma mesma capacidade de carga Q, que não pode ser ultrapassada. No PRV, após sair do depósito e atender os clientes pertencentes a sua rota, o veículo deve retornar ao ponto de partida. Assim, cada rota consiste em uma sequência de clientes que inicia e termina no depósito. No PRV, todo cliente deve ser atendido somente uma vez, por apenas um veículo.

O PRV pode ser modelado por meio de um grafo $G = (V, A)$, completo e não-direcionado, formado por um conjunto V de N vértices e um conjunto A de arestas, sendo $V = \{1, 2, \cdots, N\}$ e $A = \{(i, j)|\ i, j \in V, i \neq j\}$, respectivamente. Neste caso, o conjunto V possui N vértices, que representam os clientes e o depósito, em que o depósito possui índice igual a 1. O conjunto A de arestas representa as ligações entre os vértices. Cada cliente possui uma demanda q_i associada a ele. Além disso, para cada uma das arestas (i, j), há um custo de viagem c_{ij} associado, que é equivalente à distância entre os dois clientes i e j.

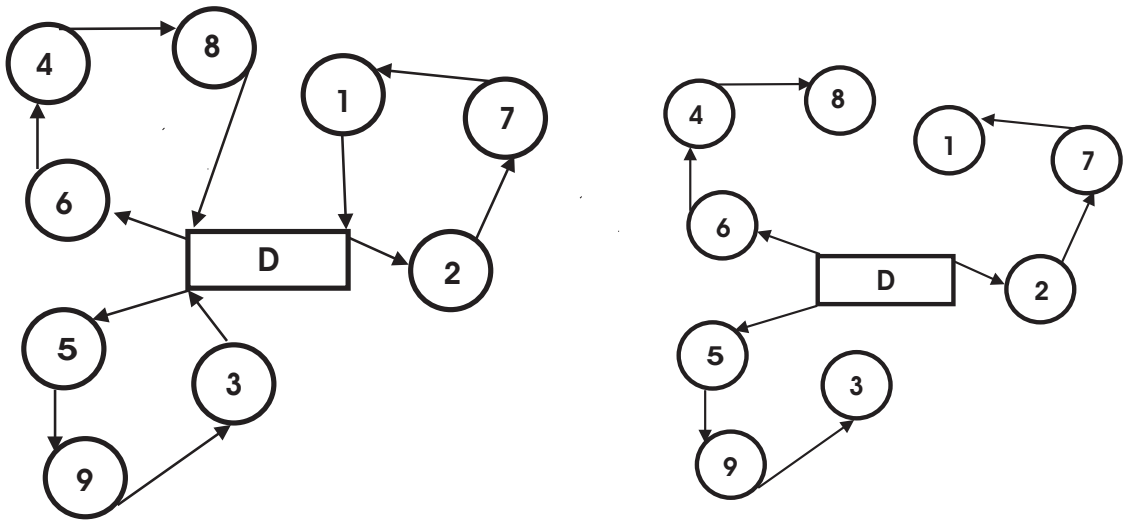
2.2 Problema de Roteamento Aberto de Veículos

Li et al. (2009) consideram o PRAV como sendo uma relaxação do PRV clássico. Esse problema pode ser representado na forma de um grafo $G = (V, A)$ completo e não-direcionado, que possui um conjunto V de vértices e um conjunto A de arestas, sendo $V = \{1, 2, \cdots, N\}$, o que inclui o depósito, e os demais vértices são os clientes a serem atendidos pelos veículos, a partir do depósito. O conjunto de clientes a serem atendidos é $C = \{2, 3, \cdots, N\}$. O conjunto A de arestas é definido como $A = \{(i, j)|\ i, j \in V, i \neq j\}$. A cada aresta (i, j) está associado um valor d_{ij}, que representa a distância entre os vértices i e j que são ligados por ela. Cada cliente i possui uma demanda q_i, que deve ser atendida por apenas um dos veículos. No PRAV, cada veículo atende a um subconjunto de clientes em sua rota, que, por sua vez, começa no depósito e termina no último cliente. A frota de veículos é homogênea e, deste modo, todos os veículos possuem os mesmos limites de capacidade Q.

Conforme MirHassani e Abolghasemi (2011), o PRAV é um problema de otimização combinatória que consiste em encontrar o número mínimo de veículos e as suas respectivas rotas, de modo que todas as demandas dos clientes sejam satisfeitas. Nesse tipo de problema, cada cliente é visitado exatamente uma única vez por um veículo, e a capacidade de cada veículo não pode ser ultrapassada.

Segundo Li et al. (2009), no PRAV o veículo não é obrigado a retornar ao depósito de origem após atender o último cliente da rota, já que eles são apenas contratados para realizar as entregas das mercadorias. Conforme Brandão (2004), o PRAV é diferente do PRV, pois os veículos só retornam para o depósito caso tenham de realizar a coleta de mercadorias. Nesse caso, o percurso é feito na ordem inversa em que foi feita a rota de entrega.

A principal diferença entre o PRAV e o PRV está no fato de que o veículo não precisa retornar ao depósito e daí deriva o nome adotado. Ou seja, no PRV as rotas se iniciam e terminam no depósito, enquanto no PRAV as rotas se iniciam no depósito e terminam em um cliente, que é nomeado como o último daquela rota. Desse modo, no PRV o percurso é definido como um ciclo hamiltoniano e, no PRAV, é definido com um caminho hamiltoniano. O PRAV é um problema que serve para modelar vários tipos de situações do mundo real, apesar de ter recebido pouca atenção dos pesquisadores. As Figuras 2.1(a) e 2.1(b) ilustram, a diferença entre o PRV e o PRAV acerca da forma que é representada uma solução.



(a) Problema de Roteamento de Veículos. (b) Problema de Roteamento Aberto de Veículos.

Figura 2.1: Caracterização de Roteamento Aberto de Veículos.

2.3 Problema de Roteamento Aberto de Veículos com Janela de Tempo

O principal objeto de estudo desta dissertação é o Problema de Roteamento Aberto de Veículos com Janela de Tempo (PRAVJT).

De acordo com Repoussis et al. (2006), esta variante do PRAV se difere da sua forma padrão devido a adição das restrições de tempo que o compõem. No PRAVJT, cada demanda q_i referente a um cliente i deve ser atendida de acordo com a sua respectiva janela de tempo, que modela o intervalo de tempo $[e_i, l_i]$ em que tal cliente pode ser atendido. Os valores e_i e l_i são os instantes de tempos inicial e final referentes ao período de atendimento do cliente, respectivamente. Além disso, cada cliente requer um tempo de serviço s_i, que é o período de tempo que o veículo deve aguardar para efetuar suas tarefas. Assim, a soma dos tempos de viagem e dos tempos de serviço dos clientes já visitados por um veículo deve ser maior ou igual ao horário inicial e menor que o horário final de atendimento da janela de tempo associada ao próximo cliente a ser visitado. Caso o veículo chegue mais cedo ao consumidor, ele pode esperar o início da janela de tempo no local. Quando o veículo chega no local após o horário inicial de atendimento, o atendimento do cliente já pode ser feito, iniciando o tempo de serviço.

Para Repoussis et al. (2006), o PRAVJT tem, como objetivo, criar o menor conjunto de rotas, de forma que sejam respeitadas as restrições que compõem tal problema. No PRAVJT, procura-se, assim, primeiramente, minimizar o número de veículos que são necessários para atender aos clientes, para que, em seguida, seja feita a minimização da distância total de viagem do número de rotas obtidas. Conforme (Brandão, 2004), o custo de contratação de um veículo adicional sempre irá ser maior do que qualquer custo de deslocamento para atender um ou mais clientes. Com isso, é preferível obter uma solução que necessite de menor número de veículos mas que per-

corra maiores distâncias de viagem, ao invés de uma solução que utilize mais veículos que caminhem por distâncias menores.

O PRAVJT é composto pelas seguintes restrições:

(i) Cada veículo inicia sua rota a partir do depósito e a termina no último cliente atendido. O veículo não pode retornar para o depósito; se o fizer, ele deve voltar pelo caminho inverso ao que foi percorrido.

(ii) Cada cliente só pode ser atendido por exatamente uma rota.

(iii) A demanda total de qualquer rota não pode ultrapassar a capacidade Q do veículo.

iv) Para cada cliente i, o início do atendimento deve estar dentro do intervalo de tempo definido por $[e_i, l_i]$.

(v) Todo veículo deve aguardar o cumprimento do tempo de serviço s_i.

De acordo com Repoussis et al. (2006), o PRAVJT pode ser analisado como sendo composto por três subproblemas, que são:

a) Problema de Entrega.

b) Problema de Coleta.

c) Problema de Coleta e Entrega.

Em relação ao problema de entrega, os veículos são designados para percorrer as rotas de entrega sem ter de retornar ao depósito. Quanto ao problema de coleta, os veículos são designados para rotas de coleta, que se iniciam a partir dos clientes, ou seja, no outro extremo da rota, tendo o depósito como destino. No caso do problema de coleta e entrega, quando o veículo termina todas as entregas, ele deve realizar o percurso na ordem inversa para coletar os bens que devem ser enviados para o depósito, ou, depois de terminar todas as coletas, ele retorna ao depósito pela rota inversa à de coleta e realiza as entregas para os clientes.

2.4 Trabalhos Relacionados ao PRAV

Segundo Brandão (2004), o PRAV foi mencionado primeiramente no trabalho desenvolvido por Schrage (1981), que descreve diversos problemas de roteamento reais e suas aplicações.

Brandão (2004) desenvolveu em sua pesquisa um algoritmo baseado na metaheurística Busca Tabu para explorar a estrutura do PRAV. Este algoritmo utiliza dois métodos para a geração da solução inicial, que são a Heurística do Vizinho mais Próximo (*Nearest Neighbour Heuristic*) e o procedimento US (*Unstringing and Stringing*) proposto por Gendreau et al. (1992), em que a melhor solução construída por um destes procedimentos é escolhida para ser refinada pela metaheurística Busca Tabu. Foram utilizadas as instâncias propostas por Cristofides et al. (1979) e Fisher (1994), propostas inicialmente para o PRV, mas que foram adaptadas por Brandão

(2004) para o PRAV. Os resultados obtidos pelo algoritmo desenvolvido por Brandão (2004) foram comparados com os do método apresentado em Sarikilis e Powell (2000), em que o primeiro apresentou um melhor desempenho.

Tarantilis et al. (2005) elaboraram, para resolver o PRAV, uma metaheurística chamada LBTA (*List Based Threshold Accepting*), que é uma variação do algoritmo *Threshold-Accepting* proposto por Dueck e Scheuer (1990). Esta metaheurística é uma abordagem de busca estocástica e consiste na exploração do espaço de busca por meio de um parâmetro de controle, que serve para indicar quais são as regiões mais adequadas do espaço para se pesquisar. Este parâmetro é chamado de limiar. A eficiência do algoritmo foi testada por meio do uso do conjunto de instâncias proposto por Cristofides et al. (1979), sendo que os resultados alcançados foram comparados com os até então presentes na literatura. Além disso, o algoritmo LBTA foi usado para solucionar um problema do mundo real. Os resultados obtidos mostraram que LBTA apresentou soluções melhores do que as presentes na literatura para a maioria dos casos. Em relação à aplicação do LBTA em um problema do mundo real, o algoritmo também demonstrou um bom desempenho.

Letchford et al. (2006) implementaram um algoritmo exato, baseado no método *Branch and Cut*, para resolver o PRAV. Foram utilizados diferentes tipos de instâncias para testar a eficiência do algoritmo exato desenvolvido. A qualidade das soluções geradas pelo algoritmo exato é comparada às soluções apresentadas em outros trabalhos, presentes na literatura, que utilizam métodos heurísticos para solucionar o PRAV. É feita uma comparação entre o grau de dificuldade para solucionar o PRAV e o PRV, usando-se, para solucionar este último, uma versão adaptada do algoritmo exato apresentado. Conclui-se, neste caso, que os métodos heurísticos são capazes de encontrar soluções ótimas apenas para instâncias de menor. Os dados referentes aos testes de desempenho demonstraram que o custo computacional para solucionar o PRAV é menor do que o do PRV. De acordo com os testes realizados, percebeu-se que, para instâncias de pequeno e de médio porte, assim como ocorre para o PRV, o uso do *Branch-and-Cut* para a obtenção de soluções exatas para o PRAV também é viável.

Repoussis et al. (2006) usaram uma heurística de construção de rotas gulosa com "olhar à frente" para solucionar o PRAVJT. Esta heurística utiliza as informações das janelas de tempo integrantes do problema através da combinação da seleção dos clientes e do critério de inserção de rotas. Com isso, o critério de aproveitamento das relações entre os clientes, definido pelas janelas de tempo, determina a sequência em que os veículos devem visitar os clientes. O desempenho do algoritmo foi testado por meio das instâncias propostas por Solomon (1987) e por Homberger e Gehring (1999). A qualidade das soluções encontradas foi comparada com a obtida pelas heurísticas desenvolvidas por Solomon (1987) e por Ioannou et al. (2001), originalmente propostas para o PRVJT, mas que, para este trabalho, foram adaptadas para o PRAVJT. Os resultados apresentados no artigo mostram que que a qualidade das soluções pelo algoritmo proposto é boa, tanto em relação ao número de veículos quanto pela distância total percorrida por eles.

Aksen et al. (2007) considerou como objeto de estudo outra variação do PRAV que é o Problema de Roteamento de Veículos Aberto com Nós dos Motoristas (PRAV-M). Neste problema, os veículos saem do depósito, atendem um conjunto de clientes e terminam suas rotas em nós especiais, que são chamados de nós de motoristas. Um nó

de motorista pode ser a casa do motorista ou um estacionamento onde o veículo passa a noite. Em seu trabalho, eles consideraram três classes de problemas do PRAV-M:

i) com duração de tempo máxima para se percorrer uma rota;

ii) sem restrições de tempo; e

iii) com duração de tempo máxima para se percorrer uma rota e horários de atendimento individuais para visitar os clientes.

Neste tipo de problema, busca-se solucionar o problema do caminho hamiltoniano que possui dois nós fixos, que são o depósito e o nó controlador. Um exemplo de aplicação para este tipo de formulação em específico é o problema de roteamento de ônibus escolares, pois, na parte da manhã, um ônibus parte do nó de motorista e busca os alunos para levá-los para a escola e, na parte da tarde, o percurso é invertido. Nesta variação do PRAV, os objetivos são os de minimização do número de ônibus, do tempo total de viagem dos alunos e o equilíbrio das cargas e dos tempos de viagem dos ônibus. Aksen et al. (2007) propuseram uma metaheurística de Busca Tabu para resolver o PRAV-M. Devido ao fato de que não existem na literatura trabalhos anteriores que tenham abordado o PRAV-M, foram utilizadas instâncias geradas aleatoriamente para os testes de desempenho. Foi utilizado o CPLEX para gerar soluções para o PRAV-M, de forma que seus resultados pudessem ser utilizadas como referência para a análise da qualidade das soluções obtidas por meio da metaheurística Busca Tabu. A partir dos resultados obtidos, percebeu-se que a heurística apresentou um bom desempenho em relação ao tempo de execução e qualidade das soluções.

Li et al. (2009) elaboraram um algoritmo híbrido, baseado nas metaheurística *Max-Min Ant System* e Busca Tabu, para a resolução do PRAV, em que a primeira é responsável por construir a solução inicial e a segunda pela busca local. O *Hyper Cube Framework*, apresentado em Blum (2004) e proposto por Christian Blum para o desenvolvimento de metaheurísticas baseadas no *Ant Colony Optimization*, foi utilizado para a implementação da metaheurística *Max-Min Ant System* usado neste trabalho. Foram utilizadas 14 das instâncias propostas por Cristofides et al. (1979) e 2 das propostas por Fisher (1994) para testar o algoritmo proposto. Os resultados obtidos foram comparados com os apresentados por Sarikilis e Powell (2000), Brandão (2004) Tarantilis et al. (2004a), Tarantilis et al. (2004b), Fu et al. (2005) e Li e Tian (2006). De acordo com os resultados alcançados pelo algoritmo híbrido, percebeu-se que o mesmo foi capaz de encontrar soluções melhores do que as apresentadas pelos trabalhos analisados na maioria dos testes realizados. A metaheurística desenvolvida por Li et al. (2009) também foi testada para solucionar um problema do mundo real. Desta forma, foi feito um estudo de caso sobre uma editora de jornais na cidade de Xangai, China. Este editora possui cerca de 30 centros de venda, que estão distribuídos geograficamente na cidade de Xangai. Esta editora não possui frota de veículos própria, sendo necessária a contratação de veículos de terceiros para a realização dos serviços de entrega de jornais. Como os veículos não são obrigados a retornar para a sede da editora e o pagamento de cada motorista é feito de acordo com a distância total de viagem entre a sede da editora e o cliente, este problema pode

ser, então, caracterizado como um PRAV. Deste modo, a metaheurística desenvolvida foi utilizada para resolver o problema de roteamento da editora. O resultado obtido foi comparado com a abordagem utilizada pela empresa para organizar a frota de veículos para atender seus clientes. A comparação entre o resultado alcançado pelo algoritmo desenvolvido e a solução obtida por meio da abordagem adotada pela editora mostra que o primeiro apresentou uma solução melhor tanto em número de veículos utilizados quanto em distância total percorrida.

Guiyun (2009) apresentou também um algoritmo baseado na metaheurística Colônia de Formigas para solucionar, no entanto, o PRAVT. No algoritmo Colônia de Formigas, várias formigas trabalham de forma que, em toda iteração, cada uma delas é responsável por construir uma solução para o problema abordado. Durante o processo de construção, essas formigas realizam a escolha do caminho a ser percorrido de maneira estocástica, sendo esta escolha feita de forma tendenciosa, em uma relação entre a distância entre os clientes e a quantidade de feromônio que foi depositada por ela e pelas demais formigas durante o processo de busca. Deste modo, com a finalidade de melhorar o desempenho do algoritmo, neste artigo é proposto que o processo de construção das soluções seja feito pelas formigas de forma paralela, ou seja, várias formigas trabalhando na construção de soluções ao mesmo tempo. Como as formigas trabalham de forma paralela, elas interagem entre si, por meio das informações acerca do feromônio depositado por elas durante a construção das soluções pelas quais elas são responsáveis, já que o feromônio depositado por uma formiga influência o processo de decisão da outra. O algoritmo foi testado utilizando um problema teste de pequeno porte, em que, após a realização dos testes, percebeu-se que este método apresentou uma boa convergência.

Repoussis et al. (2009) propuseram o uso de um algoritmo evolucionário para resolver o PRAVJT. A cada iteração, uma nova população é gerada por meio do processo de mutação, que, por sua vez, é baseado nas arestas extraídas de pais individuais. A seleção e a combinação das arestas é feita conforme os dados contidos em um vetor de parâmetros. Os valores destes parâmetros dependem da frequência com que surgem as arestas na população e a sua diversidade corrente. As soluções referentes aos filhos gerados são melhoradas através do algoritmo Busca Tabu. É utilizado um esquema determinístico para a seleção dos indivíduos sobreviventes, auxiliando na geração de melhores indivíduos. As instâncias de Solomon (1987), compostas por 100 clientes, foram usadas para avaliar o desempenho do algoritmo. Os resultados foram comparadas aos de outros trabalhos da literatura que também abordaram o PRAVJT. Os resultados obtidos pela metaheurística desenvolvida foram melhores que os encontrados na literatura para todas as instâncias testadas.

Salari et al. (2010) apresentaram uma algoritmo para solucionar o PRAV baseado em técnicas de programação linear inteira. O algoritmo funciona de modo que, a partir de uma solução inicial, são feitas remoções dos clientes de maneira aleatória, para que os mesmos possam ser reinseridos por meio dos procedimentos do algoritmo que são baseados em métodos de programação linear inteira. Para testar o algoritmo proposto, foram utilizadas algumas das instâncias propostas por Cristofides et al. (1979), Fisher (1994), Li et al. (2007) e Derigs e Reuter (2009). Com base nos resultados encontrados na literatura, o algoritmo desenvolvido demonstrou ser capaz de gerar soluções de boa qualidade, sendo que, para 10 instâncias, a técnica proposta encontrou soluções melhores do que as encontradas na literatura.

Fenghua e Xiaonian (2010) estudaram o PRAVIJTS (Problema de Roteamento Aberto de Veículos Incompleto com Janelas de Tempo Suaves), que é uma variante do PRAVJT. Para solucionar o PRAVIJTS, foi projetado um algoritmo genético adaptado para este problema. O desempenho do algoritmo genético desenvolvido foi comparado ao da heurística de economias de *Clarke-Wright*. Para a realização dos testes foi utilizado um problema teste composto por 100 clientes, sendo que a capacidade dos veículos igual a 4 toneladas e o número máximo de veículos igual 15. Com base nos resultados obtidos, o algoritmo genético conseguiu um desempenho melhor do que a heurística de *Clarke-Wright*, gerando um solução de maior qualidade.

Em seu trabalho, MirHassani e Abolghasemi (2011) utilizaram uma versão do algoritmo PSO (*Particle Swarm Optimization*) para resolver o PRAV. Foi utilizado no PSO um método de decodificação, que consiste no uso de um vetor para armazenar as posições dos clientes em ordem decrescente, sendo cada cliente designado para uma rota, levando-se em conta as condições de factibilidade. Com isso, é também aplicado um movimento sobre as rotas construídas, com o objetivo de se obter soluções melhores. O procedimento responsável por realizar o movimento de modificação sobre as rotas faz a realocação de um cliente para outra posição da solução corrente, em que este cliente pode ser tanto reinserido em outra posição da mesma rota ou de uma rota diferente. O algoritmo foi testado através da utilização de 15 instâncias encontradas na literatura que possuem entre 19 e 72 clientes. O algoritmo PSO foi capaz de alcançar em 80% dos testes a solução ótima.

Na Tabela 2.1 é apresentada uma lista dos trabalhos relacionados ao PRAV e suas variantes que foram descritas na seção 2.4.

Tabela 2.1: Trabalhos relacionados ao PRAV.

Trabalhos	Sub-tipo	Método	Instâncias
Brandão (2004)	PRAV	Busca Tabu	Cristofides et al. (1979) e Fisher (1994)
Tarantilis et al. (2005)	PRAV	LBTA	Cristofides et al. (1979)
Letchford et al. (2006)	PRAV	*Branch and Cut*	Site: http://www.branchandcut.org
Repoussis et al. (2006)	PRAVJT	Heurística construtiva	Solomon (1987) e Homberger e Gehring (1999)
Aksen et al. (2007)	PRAV-M	Busca Tabu	Aleatórias
Li et al. (2009)	PRAV	*Max−Min Ant System*	Cristofides et al. (1979) e Fisher (1994)
Guiyun (2009)	PRAVJT	Colônia de Formigas	Aleatórias
Repoussis et al. (2009)	PRAVJT	Algoritmo evolucionário	Solomon (1987)
Salari et al. (2010)	PRAV	ILP procedure	Cristofides et al. (1979), Fisher (1994), Li et al. (2007) e Derigs e Reuter (2009)
Fenghua e Xiaonian (2010)	PRAVIJTS	Algoritmo Genético	Aleatória
MirHassani e Abolghasemi (2011)	PRAV	PSO	Site: http://www.branchandcut.org

2.5 Modelos Matemáticos para PRAV

Nas subseções a seguir são descritos modelos matemáticos propostos para o Problema de Roteamento de Veículos, Problema de Roteamento Aberto de Veículos, Problema de Roteamento de Veículos com Janelas de Tempo, Problema de Roteamento Aberto de Veículos com Janelas de Tempo e Problema de Roteamento Aberto de Veículos com Nós dos Motoristas.

2.5.1 Modelo Matemático de Toth e Vigo (PRV)

Conforme Toth e Vigo (2002), na formulação matemática do PRV considera-se V como o conjunto de vértices, sendo $V = \{1, 2, \cdots, n\}$, em que n é o número de

vértices. A quantidade de clientes é igual a $n-1$, em que cada cliente i possui uma demanda q_i. O índice i do depósito é representado por 0. O número de veículos é igual $|K|$, em que cada veículo possui um índice k que varia de 1 a $|K|$. Como a frota de veículos é homogênea, a capacidade de todos é igual a C. O custo de viagem c_{ij} de um cliente i para um cliente j é associado a um valor equivalente à distância entre estes clientes. Também são usadas as variáveis de decisão x_{ijk} e z_{ik} no modelo matemático do PRV. A variável x_{ijk} recebe o valor 1 se o veículo k saiu do cliente i para atender o cliente j e recebe o valor 0 caso contrário. A variável y_{ik} recebe o valor 1 se o cliente i é atendido pelo veículo k. O modelo do PRV é dado por:

$$\min \sum_{i\ \in V} \sum_{j\ \in V} c_{ij} \sum_{k=1}^{K} x_{ijk} \tag{2.1}$$

Sujeito a:

$$\sum_{k=1}^{K} y_{ik} = 1 \ \forall\ i\ \in\ V \setminus \{0\} \tag{2.2}$$

$$\sum_{k=1}^{K} y_{ok} = K \tag{2.3}$$

$$\sum_{j\ \in V} x_{ijk} = \sum_{j\ \in V} x_{jik} = y_{ik},\ \forall i \in V, k = 1, \cdots, K \tag{2.4}$$

$$\sum_{i\ \in\ V} q_i\ y_{ik} \leqslant C, \forall k = 1, \cdots, K \tag{2.5}$$

$$\sum_{i\ \in\ S} \sum_{j\ \in\ S} x_{ijk} \leqslant |S| - 1, \forall\ S \subseteq V \setminus \{0\}, |S| \geqslant 2, k = 1, \cdots, K \tag{2.6}$$

$$y_{ik} \in \{0, 1\}, \forall\ i \in V, \forall\ k = 1, \cdots, K \tag{2.7}$$

$$x_{ijk} \in \{0, 1\}, \forall\ i, j \in V, \forall\ k = 1, \cdots, K \tag{2.8}$$

No modelo matemático descrito anteriormente, a função objetivo tem, como finalidade, modelar os custos de viagem das rotas feitas por todos os veículos. As restrições (2.2), (2.3) e (2.4) impõem que cada cliente é visitado apenas uma vez; que o número de veículos que deixam o depósito é igual a $|K|$; e que o mesmo veículo entra e sai de um dado cliente, respectivamente. A restrição (2.5) garante que o total de carga transportada por um veículo não será maior que a sua capacidade. A restrição (2.6) garante a eliminação de sub rotas. As restrições (2.7) e (2.8) definem os valores que as variáveis y_{ik} e x_{ijk} podem assumir para cada veículo k.

2.5.2 Modelo Matemático de Tan (PRVJT)

Para Tan et al. (2001) o Problema de Roteamento de Veículos com Janelas de Tempo (PRVJT) é uma extensão do PRV, em que cada cliente possui uma janela de tempo $[e_i, l_i]$ associada, que define o horário que ele pode ser atendido por um veículo k. Nesse problema, a quantidade de veículos disponíveis é igual a $k = |K|$, sendo $K = \{1, \cdots, k\}$ o conjunto dos veículos. No PRVJT, cada janela de tempo possui dois limitantes, um inferior e_i e outro superior l_i, que são responsáveis por definir o horário inicial e final em que um cliente pode ser atendido por um veículo. Neste problema, também é considerado o tempo de s_i que o veículo requer para realizar o atendimento do cliente. Quando o tempo de chegada de um veículo k em um cliente i é menor do que e_i, este veículo deve aguardar um tempo de espera w_i. Assim como no PRV, o PRVJT tem como objetivo criar uma solução em que sejam minimizados o número de veículos utilizados e a distância total percorrida por eles, mas respeitando as restrições referentes às janelas de tempo associadas aos clientes a serem atendidos. O modelo matemático do PRVJT pode ser formulado de modo que se tenha um conjunto V de vértices, sendo $V = \{0, 1, 2, \cdots, N\}$. O número de locais existentes é igual a N, ou seja, os clientes e o depósito, sendo o índice do depósito igual a 0. Cada aresta (i, j) possui um valor d_{ij} associado, em que d_{ij} é a distâncias entre dois clientes. O valor do custo c_{ij} e do tempo de viagem t_{ij} entre dois clientes i e j é equivalente a d_{ij}. A variável de decisão x_{ijk} tem como finalidade definir se um cliente i precede um cliente j atendido por um veículo k, em que ela recebe o valor 1 se o cliente i precede j e 0 caso contrário. A capacidade cada veículo é igual a Q e r_k é o tempo de atendimento máximo de uma rota atendida por um veículo k. O modelo é dado por:

$$\min \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{j \neq i}^{K} \sum_{k=1} c_{ij} x_{ijk} \tag{2.9}$$

Sujeito a:

$$\sum_{j=1}^{N} \sum_{k=1}^{K} x_{ijk} \leqslant K \ , \ i = 0 \tag{2.10}$$

$$\sum_{j=1}^{N} x_{ijk} = \sum_{j=1}^{N} x_{ijk} \leqslant 1 \ , \ i = 0, \ \forall \ k \in \{1, \cdots, K\} \tag{2.11}$$

$$\sum_{k=1}^{K} \sum_{j=0}^{N} \sum_{j \neq i} x_{ijk} = 1, i \in \ \{1, \cdots, N\} \tag{2.12}$$

$$\sum_{k=1}^{K} \sum_{i=0}^{N} \sum_{i \neq j} x_{ijk} = 1, j \in \ \{1, \cdots, N\} \tag{2.13}$$

$$\sum_{i=1}^{N} q_i \sum_{j=0}^{N} \sum_{j\neq 1} x_{jik} \leqslant Q,\ k \in \{1, \cdots, K\} \tag{2.14}$$

$$\sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{j\neq i} x_{ijk} \left(t_{ij} + s_i + w_i\right) \leq r_k,\ k \in \{1, \cdots, K\} \tag{2.15}$$

$$t_0 = s_0 = w_0 = 0 \tag{2.16}$$

$$\sum_{k=1}^{K} \sum_{i=0}^{N} \sum_{i\neq j} x_{ijk} \left(t_{ij} + s_i + w_i\right) \leqslant t_j,\ j \in \{1, \cdots, N\} \tag{2.17}$$

$$e_i \leq (t_i + w_i) \leq l_i, i \in \{1, \cdots, N\} \tag{2.18}$$

$$x_{ijk} \in \{0, 1\} \tag{2.19}$$

Neste modelo, a função objetivo tem como finalidade minimizar os custo de viagem das rotas. A restrição (2.10) define que todos os veículos saem do depósito e que o número máximo de veículos utilizados é igual a K. A restrição (2.11) garante que os veículos retornam para o depósito. A restrição (2.12) indica que um cliente é atendido apenas uma vez e somente por um veículo. A restrição (2.12) juntamente com a restrição (2.13) assegura a continuidade da rota, fazendo com que cada veículo saia do cliente atendido por ela após o termino do serviço. A restrição (2.14) evita que a carga total de um veículo seja maior do que a sua capacidade. A restrição (2.15) faz com o tempo total gasto por um veículo para atender uma rota não seja maior do que o limite r_k. A restrição (2.16) define que os tempos iniciais de viagem, serviço e espera são iguais a 0. A restrição (2.17) garante que o tempo de viagem de um cliente i para um cliente j gasto por um veículo k não pode ser maior do que o limitante superior da janela de tempo do cliente j. A restrição (2.18) é responsável pelo respeito das janelas de tempo dos clientes. A restrição (2.19) define os valores que a variável x_{ijk} pode assumir.

2.5.3 Modelo Matemático de MirHassani e Abolghasemi (PRAV)

De acordo com o modelo matemático apresentado por MirHassani e Abolghasemi (2011), no PRAV o conjunto de vértices é representado por V, sendo $V = \{0, 1, 2, \cdots, n\}$ e n é o número de vértices. Cada um dos vértices representa um dos clientes a serem atendidos pela frota de veículos. O depósito é representado pelo vértice de índice 0. Cada cliente i, em que $i \in V$, com exceção do depósito, possui uma demanda q_i associada. O conjunto de veículos é definido como k, de modo que a quantidade de veículos é igual a $\mid K \mid$ e a capacidade de carga de cada um deles é igual a Q. Um valor c_{ij} é associado ao custo de viagem de um cliente i para um cliente j, em que c_{ij} é usada como uma medida com ajustes apropriados. O custo de ativação de um veículo k é definido por um valor w_k.

São utilizadas as variáveis de decisão x_{ij}^k e z_k para a definição da formulação matemática do PRAV. A variável de decisão x_{ij}^k serve para definir a sequência em que os veículos visitam os consumidores, em que x_{ij}^k recebe o valor 1 se o veículo k partiu do cliente i para visitar o cliente j e 0, caso contrário. A variável z_k serve para definir se o veículo k está ativo, em que a mesma recebe o valor 1 e 0, caso contrário. Um veículo é considerado ativo se ele estiver atendendo pelo menos um cliente. O modelo é dado por:

$$\min \sum_{k=1}^{K} \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij}^k + \sum_{k=1}^{K} w_k z_k \tag{2.20}$$

Sujeito a:

$$\sum_{k=1}^{K} \sum_{i=0}^{n} x_{ij}^k = 1, \forall\ j = 1, 2, ..., n \tag{2.21}$$

$$\sum_{k=1}^{K} \sum_{j=1}^{n} x_{ij}^k = 1, \forall\ i = 1, 2, ..., n \tag{2.22}$$

$$x_{ij}^k \leq z_k, \quad \forall\ k = 1, ..., K \quad \forall\ i = 0, 1, 2, ..., n \quad \forall\ j = 1, 2, ..., n \tag{2.23}$$

$$\sum_{i=0}^{n} x_{iu}^k - \sum_{j=1}^{n} x_{uj}^k = 0, \forall\ k = 1, 2, ..., k \quad \forall\ u = 1, 2, ..., n \tag{2.24}$$

$$\sum_{(i,j)\ \in\ S \times S} x_{ij}^k \leq |S| - 1, \forall\ S \subseteq V : 1 \leq |S| \leq n, \forall\ k \tag{2.25}$$

$$\sum_{i=1}^{n} q_i \left(\sum_{i=0}^{n} x_{ij}^k \right) \leq Q, \quad \forall\ k = 1, 2, ..., K \tag{2.26}$$

$$\sum_{j=1}^{n} x_{0j}^k \leq 1, \quad \forall\ k = 1, 2, ..., K \tag{2.27}$$

$$\sum_{i=1}^{n} x_{i0}^k = 0, \quad \forall\ k = 1, 2, ..., K \tag{2.28}$$

$$x_{ij}^k \in \{0, 1\} \tag{2.29}$$

$$\forall\ k = 1, 2, ..., K$$

$$\forall\ i = 0, 1, ..., n$$

$$\forall\ j = 1, ..., n$$

$$z_k \in \{0, 1\} \tag{2.30}$$

$$\forall\ k = 1, 2, ..., K$$

No modelo matemático do PRAV, a função objetivo apresenta a negociação entre as rotas e os custos dos veículos. O primeiro termo da função objetivo representa o custo das rotas feitas por todos os veículos após deixarem o depósito. O segundo termo da função objetivo representa o custo total dos veículos utilizados. As restrições (2.21) e (2.22) servem para garantir que apenas um veículo entra e sai de cada cliente e do depósito, respectivamente. A restrição (2.23) se refere às variáveis de decisão x_{ij}^k e z_k e garante que todos os clientes são atendidos por veículos ativados. O conjunto de restrições em (2.24) define o fluxo de conservação, que assegura a continuidade de cada rota de veículos. A restrição (2.25) serve para eliminar as sub rotas, e a restrição (2.26) define que a carga total transportada por um veículo não pode ser maior que a sua capacidade de carga. As restrições (2.27) e (2.28) têm como finalidade garantir que apenas um veículo irá deixar o depósito para atender a uma sequência de clientes e que nenhum veículo retornará ao depósito, respectivamente. Nas restrições (2.29) e (2.30) são definidos os valores que as variáveis de decisão x_{ij}^k e z_k podem assumir.

2.5.4 Modelo Matemático de Yu (PRAV)

No modelo matemático do PRAV utilizado por Yu et al. (2011), o conjunto de clientes é $N = \{1, 2, \cdots, n\}$, sendo n o número de clientes. O conjunto de clientes mais o depósito é N_0, sendo $N_0 = \{0, 1, \cdots, n\}$. O depósito é identificado pelo índice 0. O conjunto de veículos é $K = \{1, 2, \cdots, k\}$ e cada cliente i, sendo $i \in N$, possui uma demanda d_i associada. Cada rota inicia no depósito e termina no último cliente atendido pelo veículo utilizado. Além disso, cada veículo possui uma capacidade de carga L_h, sendo $h \in K$. Desta forma, o somatório das demandas dos clientes atendidos por um veículo não pode ser maior do que a sua capacidade L_h. Percebe-se que, neste modelo, cada veículo possui uma capacidade de carga L_h, o que caracteriza o PRAV com frota heterogênea. Para que esse modelo possa ser adaptado para representar o PRAV com frota homogênea, basta considerar L_h igual a L para todo os veículos $h \in K$. O custo de viagem entre dois clientes i e j é associado a um valor c_{ij} que, por sua vez, representa a distância entre tais clientes. São também utilizadas as variáveis de decisão y_{ih} e x_{ijh}. A variável de decisão y_{ih} serve para indicar se um cliente i foi atendido por um veículo h, em que ela recebe o valor 1 se tiver sido atendido e 0, caso contrário. A variável de decisão x_{ijh} assume o valor 1 se o veículo h partiu do cliente i para atender o cliente j e 0, caso contrário.

$$\min \sum_{i=0}^{n} \sum_{j=1}^{n} \sum_{h=1}^{K} c_{ij} x_{ijh} \tag{2.31}$$

Sujeito a:

$$\sum_{i=0}^{n} d_i y_{ih} \leqslant L_h, \ \forall \ h \in K \tag{2.32}$$

$$\sum_{h=1}^{k} y_{ih} = 1, \ i = \{1, 2, ..., n\} \tag{2.33}$$

$$\sum_{i=0}^{n} x_{ijh} = y_{jh}, \ j = \{1, 2, \cdots, n\}, \forall \ h \in K \tag{2.34}$$

$$\sum_{i=0}^{n} x_{ijh} = y_{ih}, \ i = \{0, 1, \cdots, n\}, \forall \ h \in K \tag{2.35}$$

$$\sum_{j=1}^{n} x_{jih} = 0, \ j = \{1, 2, \cdots, n\}, \forall \ h \in K \tag{2.36}$$

$$x_{ijh} \in \{0, 1\} \ \ i, j = \{1, 2, \cdots, n\}, \forall \ h \in K \tag{2.37}$$

$$y_{ih} \in \{0, 1\} \ \ i = \{1, 2, \cdots, n\}, \forall \ h \in K \tag{2.38}$$

Neste modelo, a função objetivo tem a finalidade de minimizar a distância total percorrida pelos veículos. A restrição (2.32) evita que a carga transportada pelos veículos ultrapasse a sua capacidade. A restrição (2.33) expressa que todo cliente deve ser atendido. As restrições (2.34) e (2.35) garantem que todo cliente é servido por exatamente um veículo. A restrição (2.36) impede que os veículos retornem para o depósito. As restrições (2.37) e (2.38) especificam os valores que as variáveis x_{ijh} e y_{ih} podem assumir.

2.5.5 Modelo Matemático de Guiyun (PRAVJT)

Para Guiyun (2009), na formulação matemática do PRAVJT assume-se que cada veículo possui uma capacidade de carga Q e que a distância total percorrida por um veículo não pode ser maior do que L. Cada veículo só pode atender um mesmo cliente apenas uma vez, sendo que cada cliente não pode ser atendido por mais de um veículo. Toda rota começa no depósito e termina no último cliente a ser atendido.

Cada cliente possui uma demanda q_i e a distância entre dois clientes i e j é igual a um valor d_{ij}. A quantidade de veículos disponíveis é igual a K. Cada cliente i deve ser atendido de acordo com limites inferior a_i e superior b_i de sua janela de tempo $[a_i, b_i]$. Caso um veículo chegue em um cliente i antes do tempo definido pelo limite inferior de sua janela de tempo, o veículo deve aguardar um tempo de espera antes de iniciar o seu atendimento. Ao iniciar o atendimento de um cliente, o veículo também consome um tempo de serviço t_i para realizar a descarga das mercadorias. Desta forma, o tempo de viagem de um veículo é o somatório dos seus tempos de viagem, espera e serviço. V é o conjunto de veículos, sendo $V = \{1, 2, \cdots, K\}$; C é o conjunto de clientes, sendo $C = \{1, 2, \cdots, n\}$ e $N = \{0, 1, 2, \cdots, n\}$ é o conjunto de clientes mais o depósito; o depósito possui índice igual a 0. A variável de decisão x_{ijk} serve para indicar se a aresta (i, j) faz parte da rota k, em que ela assume o valor 1 caso ela faça parte e 0, caso contrário. A formulação é dada por:

$$\min \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} \tag{2.39}$$

$$\min \sum_{k \in V} \sum_{j \in N} x_{ojk}$$

Sujeito a:

$$\sum_{k \in V} \sum_{i \in N} x_{ijk} = 1, \forall\ i \in C \tag{2.40}$$

$$\sum_{i \in C} q_i \sum_{j \in N} x_{ijk} \leqslant Q, \forall\ k \in V \tag{2.41}$$

$$\sum_{j \in C} x_{0jk} = 1, \forall\ k \in V \tag{2.42}$$

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0, \forall\ k \in V, \forall\ h \in C \tag{2.43}$$

$$s_{ik} + T_i + t_{ij} - M\,(1 - x_{ijk}) \leqslant s_{jk},\ \forall k \in V,\ \forall i, j\ \in N \tag{2.44}$$

$$a_i \leq s_{ik} \leq b_i,\ \forall k \in V,\ \forall i\ \in N \tag{2.45}$$

$$\sum_{i \in N} \sum_{j \in N} d_{ij} x_{ijk} \leqslant L, \forall\ k \in V \tag{2.46}$$

$$\sum_{j \in C} x_{j0k} = 0, \forall\ k \in V \tag{2.47}$$

$$x_{ijk} \in \{0,1\}\,,\ \forall k \in V,\ \forall i,j\ \in N \tag{2.48}$$

Nesta formulação para o PRAVJT, o primeiro e o segundo termo da função objetivo são responsáveis pela minimização do custo total de viagem das rotas e do número de veículos, respectivamente. A restrição (2.40) garante que cada cliente é visitado por apenas um veículo. A restrição (2.41) assegura que não seja ultrapassada a capacidade de carga dos veículos. De acordo com a restrição (2.42), apenas K veículos podem ser utilizados. A restrição (2.43) define que o mesmo veículo entra e sai de cada cliente atendido por ele. A restrição (2.44) serve para evitar que o horário de chegada de um veículo que parte de um cliente i e chega em um cliente j não seja maior do que o tempo final de atendimento deste último. M é um número de valor alto. A restrição (2.45) evita que o horário de atendimento de um cliente não seja dentro do horário definido por sua respectiva janela de tempo. A restrição (2.46) especifica que o custo de viagem de um rota não seja maior do que L. A restrição (2.47) faz com que nenhum veículo retorne ao depósito. A restrição (2.48) define os valores que a variável x_{ijk} pode assumir.

2.5.6 Modelo Matemático de Repoussis (PRAVJT)

Segundo Repoussis et al. (2006), o PRAVJT pode ser formulado de modo que se tenha um conjunto de rotas abertas para um conjunto de $|K|$ veículos de igual capacidade Q. Desta forma, os veículos devem atender a um conjunto de clientes $C = \{2, 3, \cdots, n\}$ a partir do depósito, com o objetivo de minimizar o custo da viagem. A quantidade de locais é n, o que também inclui o depósito, e a quantidade de clientes é $n-1$. Os índices i, j e u referem-se aos clientes e assumem valores entre 2 e n. O depósito é identificado pelo índice $i = 1$. O índice k conta os veículos, assumindo valores que vão de 1 a $|K|$. Cada cliente i possui uma demanda q_i, que deve ser atendida de acordo com o horário definido por sua respectiva janela de tempo $[e_i, l_i]$, que modela o período de tempo que o cliente pode ser atendido por um veículo. Além disso, há um tempo de atendimento s_i, que é necessário para o atendimento de cada cliente i por um veículo. Associados com a sequência i de clientes para os clientes j, há um custo c_{ij}, um tempo de viagem t_{ij} e uma distância d_{ij}. Assume-se também que c_{ij}, t_{ij} e d_{ij} são medidas equivalentes, com ajustamentos adequados. Um custo w_k (custo para a contratação de um veículo) é usado para a ativação do veículo $k \in K$. Todas as rotas devem satisfazer às restrições de capacidade e de tempo, e as restrições da janela de tempo definem que o veículo não pode atender um cliente i antes ou depois dos limites de tempo inferior e superior, respectivamente.

O Problema de Roteamento Aberto de Veículos com Janelas de Tempo possui uma formulação similar ao PRAV. A variável x_{ij}^k serve para definir a sequência em que os veículos visitam os clientes, de modo que assume o valor 1 se o cliente i precede um cliente j já visitado por um veículo k e 0, caso contrário. As variáveis a_i e p_i especificam, para cada cliente i, os instantes de chegada e partida para o cliente i, respectivamente. A variável z_k define se o veículo k está ou não ativo, de modo que assume o valor 1 se estiver e 0, caso contrário. Um veículo é considerado ativo quando o mesmo está atendendo pelo menos um cliente. A formulação matemática do PRAVJT é, então, segundo Repoussis et al. (2006), dada por:

$$\min \sum_{k=1}^{|K|} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^{k} x_{ij}^{k} + \sum_{k=1}^{|K|} w_k z_k \tag{2.49}$$

Sujeito a:

$$\sum_{k=1}^{|K|} \sum_{i=1}^{n} x_{ij}^{k} = 1, \forall\ j = 2, 3, ..., n \tag{2.50}$$

$$\sum_{k=1}^{|K|} \sum_{j=1}^{n} x_{ij}^{k} = 1, \forall\ i = 2, 3, ..., n \tag{2.51}$$

$$x_{ij}^{k} \leq z_k, \forall\ i, j = 2, 3, ..., n \tag{2.52}$$

$$\sum_{i=1}^{n} x_{iu}^{k} - \sum_{j=1}^{n} x_{uj}^{k} = 0, \forall\ k = 2, 3, ..., |K|, \tag{2.53}$$

$$\forall\ u = 1, 2, ..., n$$

$$\sum_{(i,j) \in\ S \times S} x_{ij}^{k} \leq |S|, \forall\ S \subseteq C : 2 \leq |S| \leq n, \forall\ k \in K \tag{2.54}$$

$$\sum_{i=1}^{n} q_i \left(\sum_{j=1}^{n} x_{ij}^{k} \right) \leq Q, \forall\ k = 1, 2, ..., |K| \tag{2.55}$$

$$a_j \geq (p_i + t_{ij}) - \left(1 - x_{ij}^{k}\right) M, \forall\ k = 1, 2, ..., |K|, \tag{2.56}$$

$$\forall\ i, j = 1, 2, ..., n$$

$$a_j \leq (p_i + t_{ij}) + (1 - x_{ij}^{k}) M, \forall\ k = 1, 2, ..., |K|, \tag{2.57}$$

$$\forall\ i, j = 1, 2, ..., n$$

$$a_j \leq p_i - s_i, \forall\ i = 2, 3, ..., n \tag{2.58}$$

$$e_i \leq p_i \leq l_i, \forall\ i = 2, 3, ..., n \tag{2.59}$$

$$p_1 = 0 \tag{2.60}$$

$$x_{ij}^k \in \{0,1\}, \forall\ k = 1,2,...,|K|, \tag{2.61}$$

$$z_k \in \{0,1\}, \forall\ k = 1,2,...,|K|, \tag{2.62}$$

A função objetivo (2.49) modela a negociação entre as rotas e os custos com os veículos. O primeiro termo da função objetivo refere-se ao custo das rotas seguidas por todos os veículos depois de se afastarem do depósito. O segundo termo da expressão (2.49) representa o custo de contratação dos veículos. As restrições (2.50) e (2.51) garantem que exatamente um veículo entra e sai de cada cliente. A restrição (2.52) relaciona as variáveis de decisão x_{ij}^k e z_k, garantindo que todos os clientes são atendidos por veículos ativos. A restrição (2.53) é a expressão de conservação de fluxo, que garante a continuidade de cada rota de veículo. A restrição (2.54) serve para eliminar as sub rotas. A restrição (2.55) define que a quantidade de carga transportada por um veículo não pode ser superior a Q, que é a sua capacidade. As restrições (2.56) e (2.57) estão relacionadas com as janelas de tempo e servem para garantir a viabilidade do cronograma para cada veículo. Caso os clientes i e j sejam consecutivos na rota do veículo k, o horário de chegada ao cliente j é igual ao horário de partida do cliente i, mais o tempo de viagem entre estes dois clientes. Nas restrições (2.56) e (2.57), M é um número com valor grande. No caso de os clientes i e j não serem atendidos pelo mesmo veículo ou não serem consecutivos, as restrições (2.56) e (2.57) ficam inativas. As restrições (2.58) e (2.59) garantem que as relações entre os instantes de chegada, partida e de serviço, no que diz respeito ao cliente i, são compatíveis com a janela de tempo do cliente. A restrição (2.60) define a hora de partida a partir do depósito como igual a zero, pois todas as rotas provêm do depósito. As restrições (2.61) e (2.62) definem as variáveis x_{ij}^k e z_k para cada veículo k.

As expressões citadas fazem parte da definição do Problema de Roteamento de Veículos Clássico com Janelas de Tempo, não levando em conta as restrições relacionadas com a abertura de caminhos (caminhos hamiltonianos). Desse modo, deve-se levar em consideração a disponibilidade dos veículos, limitando o número de arcos relacionados a cada veículo, que diretamente deixam o depósito.

No PRAVJT, o problema de entrega é definido pelas seguintes restrições:

$$\sum_{j=2}^{n} x_{1j}^k \leq 1, \forall\ k = 1,2,...,|K| \tag{2.63}$$

$$\sum_{i=2}^{n} x_{i1}^k = 0, \quad \forall\ k = 1,2,...,|K| \tag{2.64}$$

As restrições (2.63) e (2.64) definem que apenas um veículo irá deixar o depósito para atender uma sequência de clientes e que nenhum veículo irá retornar para o depósito, respectivamente.

O processo de coleta funciona de forma contrária ao de entrega, em que os veículos são designados para as rotas de coleta, que iniciam seus percursos no final das

rotas de entrega e terminam no depósito. As restrições (2.65) e (2.66) garantem que exatamente um veículo irá visitar uma sequência de clientes que termina no depósito e que tal veículo não irá deixar o depósito novamente, respectivamente.

$$\sum_{i=2}^{n} x_{i1}^{k} \leq 1, \quad \forall\ k = 1, 2, ..., |K| \tag{2.65}$$

$$\sum_{j=2}^{n} x_{1j}^{k} = 0, \quad \forall\ k = 1, 2, ..., |K| \tag{2.66}$$

Por meio da adição das restrições (2.67), (2.68) e (2.69) nessa formulação do PRAVJT, obtém-se o modelo do problema de coleta e entrega, as restrições específicas desse problema são descritas a seguir:

$$\sum_{j=2}^{n} x_{1j}^{k} \leq 1, \quad \forall\ k = 1, 2, ..., |K| \tag{2.67}$$

$$\sum_{i=2}^{n} x_{i1}^{k} \leq 1, \quad \forall\ k = 1, 2, ..., |K| \tag{2.68}$$

$$x_{ij}^{k} = x_{ji}^{k}, \qquad \forall\ k = 1, 2, ..., |K| \tag{2.69}$$

A restrição (2.67) especifica que apenas um veículo irá sair do depósito para atender uma sequência de clientes, realizando o serviço de entrega. A restrição (2.68) se refere ao serviço de coleta em que tal restrição serve para garantir que somente um veículo irá atender uma sucessão de clientes que inicia no último cliente atendido de uma rota de entrega e termina no depósito. No problema de coleta e entrega, ao terminar de atender todos os clientes realizando o serviço de entrega o veículo realiza o serviço de coleta pelo caminho contrário ao da rota de entrega. Deste modo, a restrição de igualdade (2.69) define que um veículo k deve visitar um cliente i seguido de um cliente j durante o processo de entrega, e visitar um cliente j seguindo de um cliente i quando é feito o processo de coleta. Restrições separadas de janela de tempo podem ser necessárias para cada cliente, tanto no processo de entrega quanto o de coleta.

Este modelo matemático descreve o PRAVJT bem como todas as restrições relacionadas às janelas de tempo. Além das restrições descritas, o tempo de viagem total gasto por um veículo para percorrer a sua rota não deve ultrapassar um determinado limite. Repoussis et al. (2006) consideram que não há necessidade de se definir uma restrição para limitar o tempo de viagem total de cada rota de veículo, pois, assumindo que há uma janela de tempo associada com o depósito, as restrições responsáveis pelo controle do cronograma de atendimento dos clientes também realizam o controle do tempo ou distância do veículo.

2.5.7 Comparação entre os Modelos Matemáticos

Ao analisar os modelos matemáticos mostrados em Repoussis et al. (2006), Guiyun (2009), Yu et al. (2011) e MirHassani e Abolghasemi (2011), percebe-se que eles possuem formulações semelhantes as do PRV e PRVJT. Porém, as restrições (2.63) e (2.64) do modelo encontrado em Repoussis et al. (2006), por exemplo, definem tais modelos como sendo variações do PRAV, diferenciando-os do modelo para o PRV, já que essas restrições servem para evitar que qualquer veículo retorne ao depósito, como é proposto para o PRAV.

Diferente do PRV e do PRVJT, a função objetivo do PRAV e do PRVJT que são apresentados por MirHassani e Abolghasemi (2011) e Repoussis et al. (2006) possuem um segundo termo, que tem, como finalidade, minimizar os gastos com a contratação dos veículos. Comparando o modelo para o PRAV mostrado em MirHassani e Abolghasemi (2011) com as demais formulações apresentadas, percebe-se que a diferença entre eles está nas restrições (2.56) a (2.60) utilizadas em Repoussis et al. (2006) para o PRAVJT. Tais restrições são referentes às janelas de tempo, que controlam os prazos de atendimento e entrega de bens dos veículos, que definem estes modelos como sendo para o PRAVJT, ao contrário da formulação exibida em MirHassani e Abolghasemi (2011), que descreve apenas o PRAV, já que ela não possui as restrições de tempo. Como o problema estudado neste trabalho é o PRAVJT e dentre os modelos matemáticos analisados, o apresentado por Repoussis et al. (2006) possui uma formulação que representa melhor esse problema, tal modelo foi o adotado para a realização do presente estudo.

Capítulo 3

Heurísticas Estudadas

Segundo Dorigo e Stützle (2004), são utilizadas duas diferentes classes de algoritmos para solucionar problemas de otimização combinatória: os métodos exatos e os aproximados. Apesar de serem capazes de resolver problemas de pequeno porte e garantir que as soluções obtidas são ótimas, o uso de métodos exatos para resolver instâncias de grande porte é inadequado no caso geral. Para Hertz e Widmer (2003), isto ocorre porque, no caso de problemas de otimização NP-Difíceis, o tempo computacional gasto para a obtenção de solução por um método exato é de tal ordem que torna computacionalmente inviável a utilização destes métodos. Assim, o uso de algoritmos aproximados, também conhecidos como heurísticas, que são capazes de encontrar soluções de boa qualidade e em tempo computacional viável acaba se tornando a forma de abordagem mais adequada. Conforme Dorigo e Stützle (2004), heurísticas são algoritmos aproximados que buscam obter soluções de boa qualidade em um tempo computacional baixo, mas sem garantir que as soluções obtidas são ótimas.

Neste Capítulo é feita, na Seção 3.1, uma descrição de Heurísticas Construtivas, tendo-se ênfase em heurísticas construtivas utilizadas diretamente em problemas de roteamento de veículos. Em seguida, na Seção 3.2, é apresentada uma descrição de Heurísticas de Busca Local. A Seção 3.3, por fim, descreve as metaheurísticas de interesse dessa dissertação, em especial a família de metaheurísticas Colônia de Formigas, além das metaheurísticas baseadas em *Iterated Local Search* (ILS) e GRASP.

3.1 Heurísticas Construtivas

De acordo com Dorigo e Stützle (2004), as heurísticas construtivas são algoritmos que têm como função construir uma solução para um problema de otimização de maneira incremental, ou seja, passo a passo. Com isso, o processo de construção de uma solução é feito de forma que ele se inicia a partir de uma solução vazia e, a cada etapa, é adicionado um novo componente da solução, até que a mesma fique completa. O critério de escolha dos elementos a serem inseridos na solução pode ser aleatório ou guloso. Quando a heurística de construção trabalha de maneira aleatória, a cada iteração um elemento, dentre os ainda não inseridos na solução, é escolhido aleatoriamente e adicionado à solução. No caso de uma heurística de construção gulosa, quando é feita a adição de um novo elemento na solução que está sendo construída,

é selecionado o elemento dentre os ainda não inseridos que proporciona maior benefício ao valor da função objetivo. Os Algoritmos 1 e 2 apresentam os pseudocódigos genéricos das heurísticas de construção gulosa e aleatória, respectivamente.

Algoritmo 1: ConstrucaoGulosa

```
   Entrada: g(.),s
   Saída: Solucao
 1 início
 2     s ← 0;
 3     Inicialize o conjunto C de elementos candidatos;
 4     enquanto (C ≠ 0) faça
 5         g(t_melhor) = melhor{g(t)|t ∈ C};
 6         s ← s ∪ {t_melhor};
 7         Atualize o conjunto C de elementos candidatos;
 8     fim
 9     Retorne Solucao
10 fim
```

Algoritmo 2: ConstrucaoAleatoria

```
   Entrada: g(.),s
   Saída: Solucao
 1 início
 2     s ← 0;
 3     Inicialize o conjunto C de elementos candidatos;
 4     enquanto (C ≠ 0) faça
 5         Escolha aleatoriamente t_escolhido ∈ C;
 6         s ← s ∪ {t_escolhido};
 7         Atualize o conjunto C de elementos candidatos;
 8     fim
 9     Retorne Solucao
10 fim
```

No caso em específico do Problema de Roteamento de Veículos com Janela de Tempo (PRVJT), uma das principais e mais utilizadas heurísticas para a construção de soluções é a Heurística de Inserção e Avanço (*Push Forward Insertion Heuristic* - PFIH), apresentada em Solomon (1987). Esta heurística é descrita na subseção a seguir. Em seguida, é introduzida a Heurística Sorteio da Copa do Mundo, publicada em Costa et al. (2012), representando uma contribuição desta dissertação.

3.1.1 *Push Forward Insertion Heuristic*

A heurística PFIH (*Push Forward Insertion Heuristic*) foi proposta por Solomon (1987) para a construção de soluções para o PRVJT. O PFIH consiste em um algoritmo de construção que realiza a inserção das cidades nas rotas de acordo com o seu custo de inserção. O custo de inserção das cidades é calculado de acordo com a sua distância geográfica, limitante superior da janela de tempo e o ângulo polar entre a cidade e o depósito. Assim, quando uma cidade é designada para ser inserida na solução, o seu custo de inserção é verificado em todas as possíveis posições das rotas pertencentes à solução corrente. Quando é analisado o custo de inserção de uma cidade, é também verificado se o atendimento da cidade em determinada posição não viola as restrições das janelas de tempo e de carga dos consumidores. Com isso, após a análise da inserção da cidade em todas as posições da solução, é escolhida a posição em que o atendimento da cidade apresentar o menor custo, ou seja, a menor distância total percorrida.

A sequência em que os consumidores são escolhidos para serem inseridos na solução é determinada por meio da equação (3.1):

$$c_i = -\alpha.d_{0i} + \beta.b_i + \gamma.\left[\left(\frac{p_i}{360}\right).d_{0i}\right] \quad \forall i \in C \tag{3.1}$$

Nesta equação, os valores dos parâmetros α, β e γ foram definidos por Solomon (1987) de forma empírica, sendo fixados em $\alpha = 0,7$; $\beta = 0,1$ e $\gamma = 0,2$. O parâmetro d_{0i} é a distância entre o depósito e o consumidor i; b_i é o limitante superior da janela de tempo do consumidor i; e p_i é o ângulo polar do consumidor i em relação ao depósito.

A sequência definida para o atendimento dos consumidores na solução do PRVJT é um fator importante para a geração de soluções de qualidade. Essa é a justificativa pela qual Solomon (1987) desenvolveu esta função, uma vez que ela permite determinar a ordem de inserção dos consumidores na solução, com base nas características do problema, como a distância entre os clientes e suas janelas de tempo.

Na Figura 3.1 é ilustrada uma aplicação do PFIH. Nesta figura, é feito o processo de atendimento dos clientes 2 e 6, ainda não atendidos na solução parcialmente construída pelo PFIH. Ao ser realizada a inserção destes clientes pelo PFIH, é analisada a possibilidade da sua inserção em todas as possíveis posições existentes da solução.

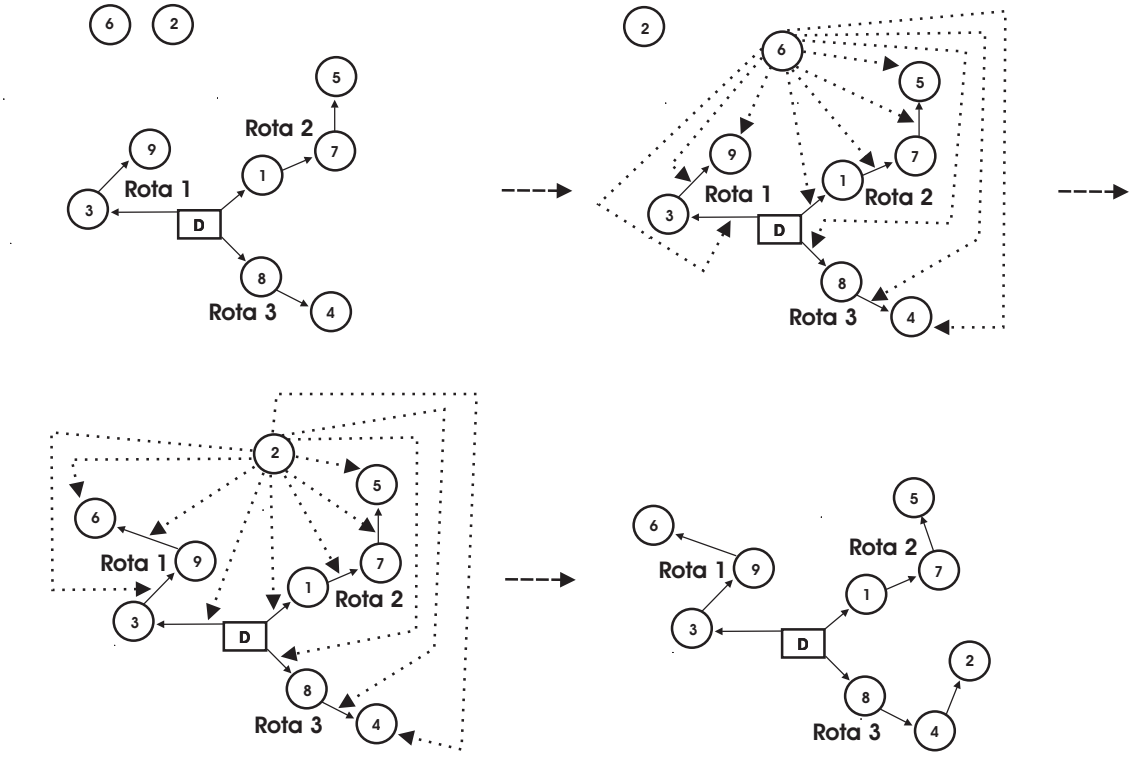


Figura 3.1: Inserção das cidades por meio do PFIH.

3.1.2 Heurística Sorteio da Copa do Mundo

Neste trabalho, é proposto o uso de um novo algoritmo para a construção de soluções para o PRAVJT. Por utilizar uma estratégia de escolha dos clientes a serem atendidos baseada na forma de sorteio que designa quais times de futebol irão jogar em determinados grupos de um torneio, como é o caso da copa do mundo de seleções de futebol, o algoritmo desenvolvido é chamado de Heurística Sorteio da Copa do Mundo (HSCM).

O HSCM funciona de modo que sejam consideradas as características do PRAVJT, como as restrições de carga e de tempo do problema, por exemplo.

Considerando o fato de que, neste problema, a frota de veículos é homogênea, a quantidade R mínima de rotas que a solução irá possuir é definida de acordo o resultado da equação (3.2):

$$R = \lceil \frac{\sum_{i=1}^{|N|} q_i}{Q} \rceil \tag{3.2}$$

Ou seja, o valor R é determinado pela divisão da demanda total dos clientes pela capacidade de carga dos veículos. Nesta expressão, q_i é a demanda de cada cliente i e Q a capacidade dos veículos. Além disso, os clientes são divididos em grupos de sorteio para serem atendidos, sendo esta divisão realizada de acordo com as suas respectivas localizações geográficas e janelas de tempo, por meio da equação (3.1) utilizada pela heurística PFIH. A quantidade de grupos de sorteio é calculada pela razão entre o número de cidades e de rotas iniciais R. Com isso, são criados $\lceil \frac{|N|}{R} \rceil$ grupos que armazenam até R clientes.

A ordem de inserção dos clientes em cada um dos grupos de sorteio criados é feita utilizando-se novamente a equação (3.1), sendo que cada cliente i é designado para fazer parte de um grupo j. Com isso, os grupos de sorteio com menor índice j tendem a receber os clientes que, segundo a equação (3.1), são os mais adequados para serem atendidos primeiro por um dos veículos da frota.

Após a divisão dos clientes em grupos, é feito o sorteio de cada um dos clientes a serem inseridos na solução. O sorteio se inicia no grupo de sorteio 1 e termina no grupo n. Desta forma, enquanto todos os clientes de um grupo j ainda não foram atendidos, é feita a escolha aleatória de um dos clientes i pertencente a este grupo para ser inserido na solução. Ao ser sorteado para ser inserido na solução, o custo de inserção do cliente escolhido é analisado em todas as posições das rotas existentes.

Durante a análise do custo de inserção, também é verificado se nenhuma das restrições de capacidade dos veículos e das janelas de tempo dos clientes são violadas. Assim, após a análise do custo de inserção, é escolhida a posição da solução que apresentar o menor custo, ou seja, aquela que levar ao menor impacto no valor da distância total percorrida pelos veículos.

Quando todos os clientes de um grupo j são atendidos, o processo de sorteio para o atendimento dos clientes é feito sobre o próximo grupo de sorteio e o processo é repetido, até que todos os clientes dos grupos de sorteio sejam atendidos.

A definição do número mínimo de rotas necessário, a divisão dos clientes em grupos de sorteio para o seu atendimento e o procedimento de inserção do cliente na posição de menor custo auxiliam a designar os clientes a serem atendidos de forma

mais uniforme. Os Algoritmos 3 e 4 mostram os pseudocódigos da heurística HSCM e do seu procedimento de construção.

Algoritmo 3: HSCM

```
   Entrada: |N|, iter_max
   Saída: Solucao
 1 início
 2     f* ← ∞;
 3     Defina o valor de R de acordo com a equação (3.2) ;
 4     Crie ⌈|N|/R⌉ grupos de sorteio;
 5     Calcule para todo o cliente i, o custo PFIH conforme a equação (3.1);
 6     Ordene em ordem crescente todos os clientes segundo o seu custo PFIH;
 7     iter ← 0;
 8     i ← 0;
 9     for j = 1 → ⌈|N|/R⌉ do
10         k ← 0;
11         enquanto (k < R && i < |N|) faça
12             Grupo(j) ← Grupo(j) ∪ {Cliente(i)};
13             k ← k + 1;
14             i ← i + 1;
15         fim
16     end
17     enquanto (iter < iter_max) faça
18         Inicialize uma solução s com R rotas;
19         s' ← Construcao(s, Grupo, |N|, R);
20         se f(s') < f(s*) então
21             s* ← s';
22             f* ← f(s');
23         fim
24         iter ← iter + 1;
25     fim
26     Retorne Solucao
27 fim
```

Algoritmo 4: Construcao

```
   Entrada: s, Grupo, |N|, R
   Saída: Solucao
 1 início
 2     ImpactoInsercao_best ← ∞;
 3     Posicao_best ← 0;
 4     for t = 1 → ⌈|N|/R⌉ do
 5         enquanto (houver clientes no Grupo(t)) faça
 6             Selecione, aleatoriamente, um cliente c do Grupo(t)
 7             enquanto todas as rotas ∈ s não forem percorridas faça
 8                 se inserção de c entre os clientes i e j respeita todas as restrições então
 9                     se f(s ∪ {c}) < ImpactoInsercao_best) então
10                         ImpactoInsercao_best ← f(s ∪ {c});
11                         Posicao_best ← (i, j);
12                     fim
13                 fim
14             fim
15             se não for possível a inserção de c em nenhuma rota então
16                 Crie uma nova rota e insira o cliente c nela;
17                 f(s) ← f(s ∪ {c});
18             senão
19                 Insira o cliente c na posição de menor custo (Posicao_best);
20                 f(s) ← ImpactoInsercao_best;
21             fim
22         fim
23     end
24     Retorne Solucao
25 fim
```

3.2 Heurísticas de Busca Local

Conforme Dorigo e Stützle (2004), uma heurística de busca local é um procedimento que tem como finalidade encontrar soluções de boa qualidade para problemas de otimização combinatória com um baixo custo de tempo computacional. Um método de busca local começa a partir de uma solução inicial, sendo seu desempenho dependente da qualidade da mesma. Este tipo de procedimento consiste na exploração de forma iterativa sobre a vizinhança da solução corrente por meio de alterações locais, que são definidas por uma estrutura de vizinhança apropriada.

A estrutura de vizinhança define quais tipos de alterações locais podem ser aplicadas sobre a solução corrente durante uma iteração do algoritmo. Segundo Dorigo e Stützle (2003), por meio das alterações locais determinadas pela estrutura de vizinhança é que se define o conjunto de soluções que pode ser obtido a partir da solução corrente. Deste modo, a heurística de busca local explora a vizinhança da solução corrente em busca de uma solução melhor. Quando uma solução melhor é encontrada, a solução anterior é substituída e o método de busca local continua até que não seja mais encontrada nenhuma solução melhor na vizinhança.

Outro fator importante em um método de busca local é a forma como as soluções são aceitas. Neste caso, a regra de escolha e aceitação da solução passa a definir a estrutura do método adotado. Regras típicas quanto a isso são Melhor-Melhora (*Best-Improvement*) e a de Primeira Melhora (*First-Improvement*), por exemplo. No critério de Melhor-Melhora, é aceita a solução vizinha que proporcionar a maior melhora no valor da função de avaliação. No caso do critério de Primeira-Melhora, é aceita a primeira solução encontrada da vizinhança que oferecer melhora no valor da função de avaliação.

Estes procedimentos são apresentados a seguir.

3.2.1 Método de Primeira Melhora

De acordo com Stützle (1998), o método de Primeira Melhora (*First Improvement Method*) é uma heurística de busca local que realiza o processo de busca de forma que quando um vizinho melhor é encontrado, o método encerra a busca. Com isso, somente no pior caso toda a vizinhança é explorada. Uma desvantagem ao se utilizar esse método é que ao utilizá-lo ele fica preso no primeiro ótimo local encontrado. Para evitar que este método, a cada iteração, gere as mesmas soluções, a exploração do espaço de busca deve iniciar a partir de diferentes regiões de tal espaço. O Algoritmo 5 apresenta o pseudocódigo do método de Primeira melhora para um problema de minimização.

Algoritmo 5: PrimeiraMelhora

```
   Entrada: f(.),N(.),IterMax,s
   Saída: Solucao
1  início
2      Iter ← 0;
3      enquanto (Iter < IterMax) faça
4          Iter ← Iter + 1;
5          Selecione s' ∈ N(s);
6          se f(s') < f(s) então
7              s ← s';
8              Retorne Solucao
9          fim
10     fim
11 fim
```

3.2.2 Método de Descida Randômica

Conforme Souza (2009), o método de Descida Randômica (*Random Descent Method*) realiza a busca local por meio da análise de um vizinho qualquer, aceitando-o apenas se o mesmo for melhor do que a solução corrente. Depois de um determinado número de iterações sem obter um vizinho melhor do que a solução corrente, este método é encerrado. Nesse método não há a garantia de que toda a vizinhança da solução corrente é explorada. Assim, não se pode assegurar que a solução obtida é um ótimo local. O Algoritmo 6 apresenta o pseudocódigo do método de Descida Randômica para um problema de minimização.

Algoritmo 6: DescidaRandomica

```
   Entrada: f(.),N(.),IterMax,s
   Saída: Solucao
1  início
2      Iter ← 0;                                  // Contador de iterações sem melhora
3      enquanto (Iter < IterMax) faça
4          Iter ← Iter + 1;
5          Selecione aleatoriamente s' ∈ N(s);
6          se f(s') < f(s) então
7              Iter ← 0;
8              s ← s';
9          fim
10     fim
11     Retorne Solucao
12 fim
```

3.3 Metaheurísticas

De acordo com Dorigo e Stützle (2004), o uso de apenas métodos heurísticos como os construtivos ou de busca local em muitos casos não permite a obtenção de soluções de alta qualidade. Isto ocorre porque heurísticas construtivas, principalmente as gulosas, permitem a geração de um número limitado de soluções e as heurísticas de busca local não fornecem meios para se escapar de ótimos locais, fornecendo soluções de baixa qualidade.

Tendo em vista as limitações dos métodos heurísticos, foi proposta a abordagem de problemas de otimização por meio de metaheurísticas.

Para Blum et al. (2011), metaheurísticas são técnicas utilizadas para a solucionar problemas de otimização combinatória, capazes de gerar soluções de alta qualidade. Dorigo e Stützle (2004) afirmam que as metaheurísticas se diferenciam de muitos tipos de procedimentos heurísticos por poderem ser usadas para resolver diferentes tipos de problemas, ao invés de problemas específicos, e pela sua capacidade de tentar escapar de ótimos locais.

As metaheurísticas fazem a exploração do espaço de busca utilizando uma heurística auxiliar, que, por sua vez, é conduzida pela metaheurística para regiões deste espaço que contém soluções de alta qualidade. Hertz e Widmer (2003) dizem que as metaheurísticas se dividem em dois tipos básicos: metaheurísticas de busca local ou de trajetórias e metaheurísticas de busca populacional.

As metaheurísticas de busca local consistem na exploração de maneira intensiva do espaço de soluções por meio da realização de movimentos, que são feitos a cada passo sobre a solução corrente, com o objetivo de gerar, em sua vizinhança, uma nova solução considerada promissora. As metaheurísticas *Simulated Annealing* (Kirkpatrick et al., 1983), Busca Tabu (Glover e Laguna, 1997), ILS (Lourenço et al., 2003) e GRASP (Feo e Resende, 1995) são exemplos de métodos de busca local.

As metaheurísticas de busca populacional trabalham mantendo um conjunto de boas soluções, que são combinadas com o objetivo de se produzir soluções ainda melhores. Dentre as metaheurísticas de busca populacional, destacam-se os Algoritmos Genéticos (Goldberg, 1989) e os Algoritmos Colônia de Formigas (Dorigo et al., 2006b).

Esta dissertação enfatiza o estudo dos Algoritmos de Colônia de Formigas. A Seção 3.4 apresenta uma extensiva revisão destes métodos, enfocando, em especial, o método *Max-Min Ant System* (MMAS), proposto em Stützle e Hoos (1996) e o método *Population-Based Ant Colony Optimization* (P-ACO), apresentado em Guntsch (2004). As subseções 3.3.1 3.3.2 e 3.3.3, a seguir, apresentam, de forma sucinta, as metaheurísticas ILS (*Iterated Local Search*), GRASP e VND, respectivamente.

3.3.1 *Iterated Local Search*

De acordo com Lourenço et al. (2003), o algoritmo Busca Local Iterativa (*Iterated Local Search* – ILS) trabalha de maneira que o processo de busca local pode ser melhorado, gerando novas soluções de partida por meio de perturbações na solução ótima local.

Os principais componentes do ILS são os procedimentos de geração da solução inicial, busca local, perturbação e critério de aceitação. O procedimento de geração da solução inicial, como o próprio nome diz, tem a função de gerar uma solução inicial s_o para o problema. O procedimento de busca local serve para encontrar uma solução s'' possivelmente melhorada. O procedimento de perturbação consiste na modificação da solução corrente s, conduzindo o processo para uma solução s'. O procedimento de critério de aceitação serve para definir qual a solução em que a próxima perturbação vai ser aplicada.

Para que o ILS possa gerar soluções de alta qualidade, é importante que a solução inicial gerada tenha boa qualidade. A escolha do método de busca a ser utilizado pelo ILS é outro fator que exerce grande influência sobre a qualidade da solução final e na velocidade de convergência. Por isso, devem ser escolhidos métodos de

busca adequados para o problema a ser abordado. Ao ser aplicado o procedimento de perturbação, a intensidade com que são feitas as perturbações sobre as soluções até então obtidas devem ser fortes o suficiente para possibilitar escapar de ótimos locais e explorar outras regiões. Mas os níveis de intensidade das perturbações não podem ser tão fortes ao ponto de não fazer com que sejam mantidas as características do ótimo local corrente. O procedimento de aceitação auxilia o algoritmo ILS a decidir sobre qual solução o processo de busca continuará e qual será o nível de perturbação a ser aplicado sobre tal solução. O Algoritmo 7 mostra o pseudocódigo da metaheurística ILS para um problema de minimização.

Algoritmo 7: ILS

```
   Entrada: iter_max
   Saída: Solucao
 1 início
 2     s_0 ← SolucaoInicial();
 3     s ← BuscaLocal(s_0);
 4     iter ← 0;
 5     enquanto (iter < iter_max) faça
 6         iter ← iter + 1;
 7         s' ← Perturbacao(s, historico);
 8         s'' ← BuscaLocal(s');
 9         s ← CriterioAceitacao(s, s', s'');
10     fim
11     Retorne Solucao
12 fim
```

3.3.2 *Greedy Randomized Adaptative Search Procedure*

Proposta por Feo e Resende (1995), a metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure* - Procedimento de busca adaptativa gulosa e randômica) é composta por duas fases, denominadas fase de construção e fase de busca local. Na fase de construção, uma solução é gerada elemento a elemento; na fase de busca local, é feita uma pesquisa por um ótimo local na vizinhança da solução construída. A melhor solução obtida ao longo do processo de todas as iterações realizadas do GRASP é retornada como resultado. O Algoritmo 8 apresenta o pseudocódigo do GRASP para um problema de minimização.

Algoritmo 8: GRASP

```
   Entrada: f(.), g(.), N(.), GRASPmax, s
   Saída: Solucao
 1 início
 2     f* ← ∞;
 3     for Iter = 1 → GRASPmax do
 4         Construcao(g(.), α, s);
 5         BuscaLocal(f(.), N(.), s);
 6         se f(s) < f* então
 7             s* ← s;
 8             f* ← f(s);
 9         fim
10     end
11     s ← s*;
12     Retorne Solucao
13 fim
```

A cada iteração da fase de construção do GRASP a solução é construída e os próximos elementos candidatos a fazerem parte da solução são inseridos em uma

lista C de candidatos de acordo com um determinado critério de ordenação. Com o auxílio de uma função adaptativa gulosa g, que estima o benefício da inserção do elemento na solução, é feito o processo de seleção dos elementos candidatos. Em cada nova iteração da fase de construção, as informações acerca dos benefícios de cada elemento são atualizados em razão das mudanças proporcionadas pela inserção do elemento inserido na solução anteriormente. Os melhores elementos da lista de candidatos atual são escolhidos para compor um subconjunto que é chamado de Lista de Candidatos Restrita (LCR), em que um dentre os elementos de tal lista é escolhido aleatoriamente para ser inserido na solução. Com isso, a cada iteração do processo de construção, são atualizadas a lista de candidatos e a lista de candidatos restrita. Assim, diferentes soluções são construídas em cada iteração do GRASP. O Algoritmo 9 a seguir representa o pseudocódigo da fase de construção do GRASP.

Algoritmo 9: Construcao

```
   Entrada: g(.), α, s
   Saída: Solucao
1  início
2      s ← ∅;
3      Inicialize o conjunto C de candidatos;
4      enquanto (C ≠ ∅) faça
5          g(t_min) = min{g(t) | t ∈ C};
6          g(t_max) = max{g(t) | t ∈ C};
7          LRC = {t ∈ C|g(t) ≤ g(t_min) + α(g(t_max) − g(t_min))};
8          Selecione, aleatoriamente, um elemento t ∈ LCR;
9          s ← s ∪ {t};
10         Atualize o conjunto C de candidatos;
11     fim
12     Retorne Solucao
13 fim
```

Para controlar o nível de gulosidade e aleatoriedade da função de escolha dos elementos da LCR, é utilizado o parâmetro α, que assume um valor no intervalo entre 0 e 1. Quanto mais próximo o valor de α for de 0, mais gulosas serão as soluções, enquanto que as soluções serão mais aleatórias se o valor de α estiver próximo de 1. Após a fase de construção é aplicado o processo de busca local sobre a solução gerada. Como não há a garantia de que a solução gerada na fase de construção representa um ótimo local, torna-se importante a adoção do procedimento de busca local para se melhorar tal solução criada. O Algoritmo 10 mostra o pseudocódigo do método de Busca Local.

Algoritmo 10: BuscaLocal

```
   Entrada: f(.), N(.), s
   Saída: Solucao
1  início
2      V = {s' ∈ N(s) | f(s') < f(s)};
3      enquanto (|V| > 0) faça
4          Selecione s' ∈ V;
5          s ← s';
6          V = {s' ∈ N(s) | f(s') < f(s)};
7      fim
8      Retorne Solucao
9  fim
```

3.3.3 *Variable Neighborhood Descent*

Segundo Mladenovic e Hansen (1997), o VND (*Variable Neighborhood Descent*) é um procedimento que opera com r diferentes estruturas de vizinhança e faz a exploração do espaço de soluções de um problema por meio de trocas sistemáticas dessas vizinhanças. São aceitas apenas soluções de melhora da solução corrente, sendo feito o retorno à primeira estrutura de vizinhança quando uma nova solução é encontrada. No método VND, a cada iteração é feita uma busca pelo melhor vizinho, utilizando-se a estrutura de vizinhança k corrente. Porém, conforme o tipo de problema abordado, a busca pelo melhor vizinho pode ter um custo computacional caro, sendo necessário o uso de outros métodos de busca local, como o de Primeira Melhora ou o método de Descida Randômica. O processo de busca local também pode ser feito de maneira que seja considerado somente um determinado percentual da vizinhança, ou seja, a busca pelo melhor vizinho é feita apenas sobre o percentual de uma vizinhança com a finalidade de evitar que tal processo de busca se torne caro computacionalmente. O Algoritmo 11 apresenta o pseudocódigo do VND.

Algoritmo 11: VND

```
   Entrada: f(.),N(.),r,s
   Saída: Solucao
 1 início
 2     Seja r o número de estruturas diferentes de vizinhança;
 3     k ← 1;                                   // Tipo de estrutura de vizinhança corrente
 4     enquanto (k < r) faça
 5         Encontre o melhor vizinho s' ∈ N^k(s);
 6         se f(s') < f(s) então
 7             s ← s';
 8             k ← 1;
 9         senão
10             k ← k + 1;
11         fim
12     fim
13     Retorne s
14 fim
```

3.4 Colônia de Formigas

Segundo Dorigo et al. (1996), a metaheurística Otimização por Colônia de Formigas é um método baseado no comportamento de colônias de formigas reais. Segundo Dorigo et al. (2006a), quando as formigas saem do seu ninho em busca de alimento, depositam no solo uma substância chamada feromônio. Ao perceberem uma concentração de feromônio mais alta em determinado caminho, as formigas tendem a seguir este caminho devido à maior concentração dessa substância. Com isso, as formigas são capazes de transportar alimentos para o seu ninho de forma eficaz, já que o caminho (entre o ninho e o alimento) que possui maior concentração de feromônio é o mais curto. Deneubourg et al. (1990), com o objetivo de compreenderem a relação entre as trilhas de feromônio deixadas pelas formigas e a sua sequência de comportamento, realizaram diversos estudos como o "experimento da ponte binária", por exemplo. Neste experimento, o ninho de uma colônia de formigas foi ligado a uma fonte de alimento por duas pontes de comprimento iguais. A Figura 3.2 ilustra a ponte utilizada no experimento citado.

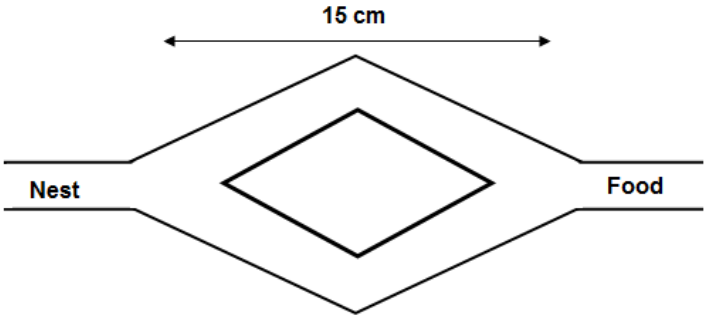


Figura 3.2: Pontes de tamanhos iguais.
Fonte:(Dorigo et al., 2006a).

Perante esta situação, as formigas exploraram os arredores do ninho até chegarem à fonte de alimento. Percebeu-se que, primeiramente, as formigas escolhiam uma das pontes de forma aleatória, e, ao longo do caminho, depositavam trilhas de feromônio. Devido às flutuações aleatórias, após certo período de tempo uma das pontes estava com uma maior concentração de feromônio que a outra, o que fazia com que as formigas fossem atraídas para a mesma, havendo uma convergência de toda a colônia para o uso da mesma ponte. Goss et al. (1989) realizaram o "experimento da ponte binária", utilizando duas pontes de tamanhos diferentes, em que uma ponte possui um tamanho relativamente maior. As formigas que escolheram de forma aleatória a ponte mais curta conseguiram chegar até o alimento e voltar para o ninho mais cedo. Como consequência, o feromônio depositado por elas na ponte mais curta adquiriu uma concentração maior, fazendo com que a probabilidade de escolha da ponte mais curta pelas demais formigas fosse maior. A Figura 3.3 ilustra as pontes de tamanhos diferentes usadas no experimento citado.

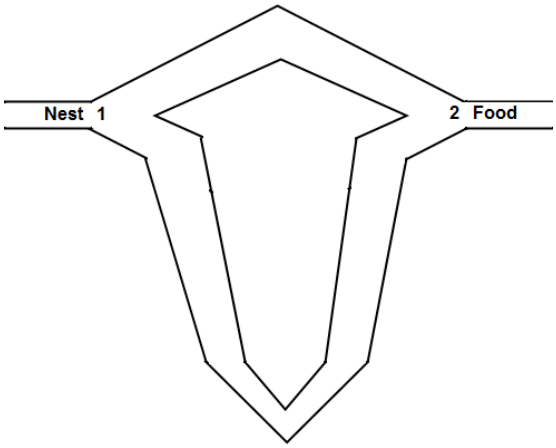


Figura 3.3: Pontes de tamanhos diferentes.
Fonte:(Dorigo et al., 2006a).

3.4.1 Metaheurística *Ant System*

Conforme Dorigo et al. (2006a), o modelo do comportamento forrageiro das formigas foi a principal fonte de inspiração para o desenvolvimento da metaheurística Colônia de Formigas. O algoritmo *Ant System*, apresentado em Dorigo et al. (1991), foi a primeira metaheurística ACO (*Ant Colony Optimization*). Neste método, um conjunto de formigas artificiais constroem soluções para um problema de otimização, e cooperam entre si, trocando informações sobre a qualidade das soluções construídas pelas mesmas. A troca de informações é feita por meio do feromônio depositado por cada uma das formigas ao se moverem dentro do espaço de busca, sendo a trilha deixada por elas usada como informação pelas demais formigas.

De acordo com Dorigo e Stützle (2003), as formigas utilizadas no método são procedimentos estocásticos de construção de soluções, que criam soluções por meio de critérios de escolha probabilísticos durante as iterações do processo. A cada iteração, são adicionadas novas informações para as soluções parciais, com base nas características da instância do problema a ser resolvido e nas trilhas de feromônio que se alteram dinamicamente, conforme a qualidade das soluções obtidas pelas formigas. As formigas artificiais, durante o processo de construção de soluções, percorrem, de forma concorrente e assíncrona, o espaço de busca do problema por meio da construção de caminhos sobre o mesmo. Os movimentos produzidos pelas formigas são realizados com base em uma política de decisão local estocástica, que se utiliza das trilhas de feromônio e informações heurísticas. A cada movimento as formigas constroem soluções para o problema de otimização. Durante a etapa de construção a formiga avalia a solução parcial ou completa e deposita o feromônio sobre os componentes ou conexões que ela usou. Esta informação de feromônio será usada para direcionar as próximas buscas a serem feitas pelas formigas.

A metaheurística Colônia de Formigas possui outros dois procedimentos, que são o de evaporação das trilhas de feromônio e as ações *daemon*, sendo o segundo componente opcional. O processo de evaporação de feromônio consiste na sua diminuição ao longo do tempo. Este processo serve para evitar uma convergência muito rápida

do algoritmo para regiões sub ótimas. Deste modo, é implementada uma forma útil de esquecimento, que favorece a exploração de novas regiões do espaço de busca. As ações *daemon* são usadas para implementar ações centralizadas, que não podem ser realizadas pelas formigas apenas. Exemplos de ações *daemon* são os procedimentos de busca local ou um conjunto de informações globais, que podem ser usadas para decidir sobre a utilidade em se depositar feromônio extra para influenciar o processo de busca sob uma perspectiva não local. A ação de observar o caminho construído por cada uma das formigas e escolher para receber um depósito de feromônio extra apenas sobre os componentes utilizados pela formiga que construiu a melhor solução é um exemplo de ação *daemon*. As atualizações realizadas pelo *daemon* são denominadas atualizações de feromônio *offline*.

Um exemplo de aplicação da metaheurística Colônia de Formigas é para a resolução do Problema do Caixeiro Viajante. Neste caso, cada formiga é inicialmente colocada em uma cidade de forma aleatória e tem uma memória que serve para armazenar as cidades já visitadas pela formiga k. Com isso, cada formiga sai de seu respectivo ponto de partida e realiza iterativamente movimentos de uma cidade para outra, ou seja, elas visitam uma série de nós em um grafo. A cada movimento, uma formiga k, estando em uma cidade i, escolhe qual será a próxima cidade j, dentre as ainda não visitadas, para fazer parte da solução. Esta escolha é feita de acordo com a probabilidade calculada pela equação (3.3):

$$p_{ij}^{k} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{l \in N_i^k}(\tau_{il})^{\alpha}(\eta_{il})^{\beta}}, \quad \text{se } j \in N_i^k, \tag{3.3}$$

Na Figura 3.4 é representado o processo de escolha da próxima cidade a ser visitada pela formiga:

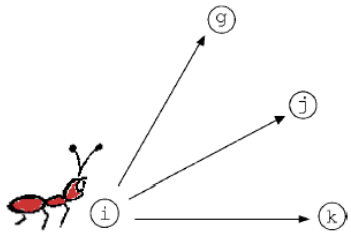


Figura 3.4: Representação do processo de escolha da cidade pela formiga.
Fonte:(Dorigo et al., 2006a).

Na equação (3.3), N_i^k representa o conjunto de cidades ainda não visitadas pela formiga k. A concentração de feromônios no arco (i,j) é representada por τ_{ij}. Os parâmetros α e β servem para determinar a influência relativa da trilha de feromônio e da informação heurística, respectivamente. O parâmetro $\eta_{ij} = \frac{1}{d_{ij}}$ é uma informação heurística disponível a *priori*, em que d_{ij} é a distância entre os vértices i e j. No caso em que $\alpha = 0$, a probabilidade de seleção é proporcional a $\lceil \eta_{ij} \rceil^{\beta}$, e as cidades mais próximas possuem mais chances de serem escolhidas. Assim, o algoritmo comporta-se de forma estocástica gulosa. Quando $\beta = 0$, apenas o uso de feromônio tem influência,

o que induz a uma estagnação rápida da busca, fazendo com que todas as formigas percorram o mesmo caminho, construindo as mesmas soluções.

A construção de uma solução termina após todas as formigas criarem os seus caminhos. Em seguida, as trilhas de feromônio são atualizadas, sendo feito o processo de evaporação das trilhas de feromônio e o depósito de feromônio. Enquanto que o processo de evaporação das trilhas é aplicado sobre todas elas, o depósito de feromônio é feito apenas nas conexões pertencentes às soluções construídas pelas formigas. A atualização do feromônio associado à aresta (i, j) é representado pela equação (3.4):

$$\tau_{ij} = (1 - \rho)\,\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{(k)} \tag{3.4}$$

Nesta equação, $0 < \rho < 1$ é a taxa de evaporação da trilha de feromônio e m é o número de formigas. O parâmetro ρ é utilizado para evitar o acúmulo ilimitado de trilhas de feromônio e permite que o algoritmo "esqueça" decisões ruins. Nos arcos que não são escolhidos pelas formigas, a concentração de feromônio diminui de forma exponencial, conforme o número de iterações. O parâmetro $\Delta\tau_{ij}^{(k)}(t)$ se refere à quantidade de feromônio depositada pela formiga k nos arcos, que, por sua vez, é definida pela equaçào (3.5):

$$\Delta\tau_{ij}^{(k)}(t) = \begin{cases} \dfrac{1}{L^k}(t) & \text{, se o arco } (i,j) \text{ é visitado pela formiga } k; \\ 0 & \text{, caso contrário;} \end{cases} \tag{3.5}$$

O parâmetro $L^k(t)$ é a distância percorrida pela formiga k. De acordo com a equação (3.5), os arcos pertencentes ao caminho mais curto feito por uma das formigas receberão mais feromônio. Assim, os arcos que forem mais utilizados pelas formigas terão maiores taxas de feromônios, pois terão maior chance de serem escolhidos. Outra forma de se calcular a quantidade de feromônio depositada em cada arco (i, j) visitado por uma formiga é através da definição do depósito de uma quantidade de feromônio proporcional ao comprimento do arco, dada por $d_{ij} \times \frac{Q}{L^k}$, sendo Q uma constante e d_{ij} a distância entre as cidades i e j.

Stützle e Hoos (1997) afirmam que metaheurísticas como Colônia de Formigas proporcionam bons resultados quando são utilizadas para resolver problemas de otimização combinatória NP-difíceis, como é o caso dos Problemas do Caixeiro Viajante e de Roteamento de Veículos.

O Algoritmo 12 apresenta o pseudocódigo da metaheurística Colônia de Formigas para um problema de minimização.

As subseções a seguir descrevem as variações desse método denominadas *Max-Min Ant System* (MMAS) e *Population-based Ant Colony Optimization* (P-ACO).

Algoritmo 12: Colônia de Formigas

```
   Entrada: Q, τ_0, iter_max, m, NumCidades
   Saída: Solucao
1  início
2      f* ← ∞;
3      iteracaoSemMelhora ← 0;
4      Faça Δτ_ij ← 0 τ_ij ← τ_0 ∀ arco(i, j);
5      enquanto (iteracaoSemMelhora < iter_max) faça
6          for k = 1 → m do
7              Selecione a cidade inicial para a k − ésima formiga;
8              Obtenha uma solução s^k para cada formiga k;
9              se f(s^k) < f* então
10                 s* ← s^k;
11                 f* ← f(s*);
12                 iteracaoSemMelhora ← 0;
13             fim
14             Calcule a quantidade de rastro deixado pela formiga k :
15             for i = 1 → NumCidades do
16                 for j = 1 → NumCidades do
17                     se arco(i, j) pertence à rota s^k então
18                         Δτ^k_ij ← d_ij × Q/L^k;
19                     senão
20                         Δτ^k_ij ← 0;
21                     fim
22                 end
23             end
24             Faça Δτ_ij ← Δτ_ij + Δτ^k_ij;
25         end
26         Faça τ_ij ← (1 − ρ) × τ_ij + Δτ_ij ∀ arco(i, j);
27         iteracaoSemMelhora ← iteracaoSemMelhora + 1;
28     fim
29     Retorne Solucao
30 fim
```

3.4.2 *MAX-MIN Ant System*

O algoritmo *MAX-MIN Ant System*, desenvolvido por Stützle e Hoos (1996), é uma das principais variações do *Ant System*. Stützle e Hoos (2000) dizem que, por meio de uma forte exploração das melhores soluções encontradas durante o processo de busca e da análise do espaço de soluções, é possível se obter melhores resultados. Porém, a realização de uma busca de forma gulosa faz com que ocorra uma convergência rápida do algoritmo. Com isso, o *MAX-MIN Ant System* nasceu da ideia de realizar um maior aproveitamento das melhores soluções encontradas no processo de busca e, ao mesmo tempo, utilizar um mecanismo para se evitar a estagnação precoce do processo de busca.

As principais características do *MAX-MIN Ant System* são o uso de apenas uma formiga no processo de atualização dos feromônios; utilização de um limitante inferior (τ_{min}) e outro superior (τ_{max}) para controlar a concentração de feromônios nas arestas utilizadas pelas formigas, de modo que se evite a rápida estagnação do processo de busca; e o uso de um valor equivalente ao valor do limitante superior (τ_{max}) para inicializar as trilhas de feromônio.

Segundo Stützle e Hoos (1999), a atualização das trilhas de feromônio é feita por apenas uma formiga ao final de cada iteração, de modo que é utilizada a formiga que encontrou a melhor solução da iteração (s^{ib}) ou a melhor solução desde o início da execução do algoritmo (s^{gb}). A equação (3.6) caracteriza a forma como é feita a atualização das trilhas de feromônio:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}^{best} \tag{3.6}$$

De acordo com Stützle e Hoos (1997), na equação (3.6), $\Delta\tau_{ij}^{best}$ é definido por $\Delta\tau_{ij}^{best} = 1/f\left(s^{best}\right)$ e $f\left(s^{best}\right)$ é o valor referente ao custo de s^{ib} ou s^{gb}. Para Stützle e Hoos (2000), o uso de apenas uma solução para atualizar as trilhas de feromônio faz com que os elementos que frequentemente fazem parte das melhores soluções encontradas tenham uma concentração de feromônio maior. Assim, há um melhor aproveitamento dos elementos que compõem as melhores soluções obtidas ao longo do processo para a construção de soluções de maior qualidade, devido à forma como é feita a atualização das trilhas de feromônio.

Porém, conforme Stützle e Hoos (2000), ao se utilizar somente s^{gb} para atualizar as trilhas de feromônio, a concentração de feromônios nas trilhas que compõem tal solução se torna tão forte que, durante a busca, as formigas são induzidas a construírem soluções muito semelhantes ou até mesmo iguais a s^{gb}, ou seja, há uma convergência rápida do processo de busca, não havendo diversificação durante a geração de novas soluções. Quando apenas s^{ib} é usada para atualizar as trilhas de feromônio, o algoritmo pode demandar uma grande quantidade de tempo para convergir e encontrar soluções de boa qualidade. Desta forma, o uso tanto de s^{ib} quanto o de s^{gb}, ao invés de apenas uma delas, torna a exploração do espaço de busca mais diversificada, evitando uma convergência precoce de tal exploração, já que os elementos que compõem s^{ib} também terão as suas respectivas trilhas de feromônio atualizadas.

Stützle e Hoos (2000) afirmam que, em instâncias de grande porte, é importante a adoção de um critério de escolha para se definir a maneira como s^{ib} e s^{gb} se alternam para a realização da atualização das trilhas de feromônio. Uma forma de se aplicar tal critério de escolha é a definição do uso de s^{ib} como principal para se atualizar a quantidade de feromônio nas arestas e utilizar s^{gb} somente a cada intervalo fixo de iterações. Este intervalo de iterações diminui e o número de vezes que s^{gb} é usada aumenta ao longo da execução do algoritmo.

Ao aplicar o *MAX-MIN Ant System* para o Problema do Caixeiro Viajante, Stützle e Hoos (2000) definiram a alternância entre s^{ib} e s^{gb} para atualizar as trilhas de feromônio de modo que s^{ib} é usada nas 25 primeiras iterações. Além disso, sendo o número de iterações realizadas igual a t, os intervalos de uso de s^{gb} foram determinados de modo que ela é usada a cada 5 iterações para $25 < t \leq 75$, 3 iterações posto que $75 < t \leq 125$, 2 iterações se $125 < t \leq 250$ e a cada 1 iteração quando $t > 250$. Este tipo de procedimento de alternância entre o uso da melhor solução da iteração (s^{ib}) ou da solução global (s^{gb}) para atualizar as trilhas de feromônio faz com que, ao longo do tempo, a utilização s^{ib} diminua e a de s^{gb} gradualmente aumente. Isto permite que, inicialmente, ocorra uma exploração mais diversificada do espaço de busca, de maneira que, ao final do processo de busca haja, uma forte exploração da região do espaço próxima da solução global.

O emprego de limitantes para controlar a concentração de feromônio e impedir a rápida estagnação do procedimento de pesquisa é outra importante característica do *MAX-MIN Ant System*. Ao ser feito o procedimento de atualização das trilhas de feromônio, são usados dois limitantes, um inferior (τ_{min}) e outro superior (τ_{max}) para regular a quantidade de feromônio que as trilhas possuem. Ao final de cada iteração,

deve ser verificado se a concentração de feromônio nas trilhas respeita os limites τ_{min} e τ_{max} definidos. Esta verificação é feita de acordo com a equação (3.7):

$$\tau_{ij}(t) = \begin{cases} \tau_{min} & , \text{se } \tau_{ij}(t) < \tau_{min} \\ \tau_{max} & , \text{se } \tau_{ij}(t) > \tau_{max} \end{cases} \tag{3.7}$$

Conforme Dorigo et al. (2006b), os valores de τ_{min} e τ_{max} podem ser definidos de maneira empírica. Porém, Stützle e Hoos (1999) propõem um método para a determinação de τ_{min} e τ_{max}, de modo que os seus respectivos valores sejam adequados para auxiliar o algoritmo a obter um desempenho eficiente. De acordo com este método, o valor de τ_{max} é calculado conforme a equação (3.8):

$$\tau_{max} = \frac{1}{(1-\rho)} * \frac{1}{f\left(s^{gb}\right)} \tag{3.8}$$

Na expressão (3.8), ρ e $f\left(s^{gb}\right)$ se referem à taxa de evaporação da quantidade de feromônio e ao custo da solução obtida, respectivamente. Quando uma nova solução é encontrada, os valores de τ_{min} e τ_{max} são atualizados.

O valor para o limitante inferior (τ_{min}) é calculado de acordo com a equação (3.9):

$$\tau_{min} = \frac{\tau_{max}(1-p_{dec})}{(avg-1)\,p_{dec}} = \frac{\tau_{max}\left(1-\sqrt[n]{p_{best}}\right)}{(avg-1)\sqrt[n]{p_{best}}} \tag{3.9}$$

Na equação (3.9), p_{dec} é a probabilidade de uma formiga escolher um determinado componente da solução. Ao construir uma solução, a formiga realiza n vezes o processo de escolha dos componentes da mesma, e a probabilidade da formiga escolher em cada passo o componente mais adequado é igual a p_{dec}^n. A probabilidade de uma formiga construir a melhor solução é igual a p_{best}; deste modo, $p_{dec}^n = p_{best}$, ou seja, $p_{dec} = \sqrt[n]{p_{best}}$. A variável $avg = n/2$ se refere ao número médio de componentes dentre os quais a formiga tem de escolher um em cada passo do processo de construção da solução. O valor de p_{best} é definido de modo que $0 < p_{best} < 1$, pois este valor deve ser significantemente maior do que 0, mas se ele for igual a 1, τ_{min} é igual a 0, e, caso o valor de p_{best} seja muito pequeno, τ_{min} pode assumir um valor maior do que τ_{max}, o que impede que haja uma diferença de valores adequada entre tais limitantes.

No *MAX-MIN Ant System*, a quantidade de feromônio inicial (τ_0) das trilhas é igual a τ_{max}. Esta medida é adotada, pois desta maneira é possível se ter uma melhor exploração do espaço de busca no início do algoritmo, já que todas as trilhas terão uma alta concentração de feromônios inicialmente. Como o valor de τ_0 recebe o valor de τ_{max}, no início do algoritmo, para se obter o valor do limitante superior por meio da expressão (3.8), o valor do $f\left(s^{gb}\right)$ inicial é igual ao custo de uma solução que é construída de maneira gulosa.

Para melhorar o desempenho do *MAX-MIN Ant System* é também utilizada a técnica de suavização das trilhas de feromônio. Esta técnica tem como finalidade facilitar a exploração do espaço de busca, aumentando as chances de escolha das trilhas de feromônio que possuem baixa concentração, o que permite que haja um maior aproveitamento de tais trilhas durante a pesquisa. Quando o algoritmo converge ou pelo menos está próximo de convergir, a concentração de feromônio das

trilhas tem o seu valor incrementado, sendo tal incremento proporcional à diferença entre a quantidade de feromônio da trilha e o limitante superior (τ_{max}). A equação (3.10) descreve como é feito o cálculo para se realizar o processo de suavização das trilhas de feromônio:

$$\tau_{ij}^{*}(t) = \tau_{ij}(t) + \delta\left(\tau_{max}(t) - \tau_{ij}(t)\right) \tag{3.10}$$

As variáveis τ_{ij} e τ_{ij}^{*} se referem à concentração de feromônio nas trilhas antes e depois do processo de suavização. O valor δ deve ser definido de modo que $0 < \delta < 1$, pois, se $\delta = 0$, o processo de suavização fica inativo, mas se $\delta = 1$, as trilhas de feromônio são reinicializadas para τ_{max}. Quando $\delta < 1$, as informações armazenadas durante o processo de busca do algoritmo não são completamente perdidas, pois elas são apenas enfraquecidas. O uso da técnica de suavização é indicado para auxiliar o algoritmo na exploração do espaço de busca quando o mesmo realiza um grande número de iterações para solucionar um determinado tipo de problema.

O Algoritmo 13 descreve o pseudocódigo do *MAX-MIN Ant System* para um problema de minimização.

Algoritmo 13: MAX-MIN Ant System

Entrada: $iter_{\max}$, m, $NumCidades$
Saída: $Solucao$

```
1  início
2      Inicialize os parâmetros α, β, ρ, τ_min e τ_max;
3      τ_0 ← τ_max, f(s^gb) ← ∞, iter ← 0;
4      Faça Δτ_ij ← 0 τ_ij ← τ_0 ∀ arco(i, j);
5      enquanto (iter < iter_max) faça
6          for k = 1 → m do
7              Selecione a cidade inicial para a k − ésima formiga;
8              Obtenha uma solução s^k para a formiga k;
9          end
10         s^ib = argmin(f(s^1), f(s^2), ..., f(s^m));
11         se f(s^ib) < f(s^gb) então
12             s^gb ← s^ib;
13             f(s^gb) ← f(s^ib);
14             Atualize os parâmetros τ_min e τ_max de acordo com as equações 3.8 e 3.9, respectivamente;
15         fim
16         defina s^best igual a s^ib ou s^gb;
17         for i = 1 → NumCidades do
18             for j = 1 → NumCidades do
19                 se arco(i, j) pertence à solução s^best então
20                     τ_ij = ρτ_ij + Δτ_ij^best;
21                     se τ_ij < τ_min então
22                         τ_ij = τ_min;
23                     fim
24                     se τ_ij > τ_max então
25                         τ_ij = τ_max;
26                     fim
27                 fim
28             end
29         end
30         iter ← iter + 1;
31     fim
32     Retorne Solucao
33 fim
```

3.4.3 *Population-Based Ant Colony Optimization*

Segundo Guntsch (2004), o algoritmo *Population-based Ant Colony Optimization* (P-ACO) é uma variante do *Ant System* que como as outras variações de tal algoritmo, se difere principalmente pela estratégia de atualização das trilhas de feromônios. Os demais procedimentos, como o de construção de soluções pelas formigas, por exemplo, são iguais aos do algoritmo Colônia de Formigas padrão. Geralmente, no processo de atualização das trilhas de feromônios, são utilizados os procedimentos de depósito e evaporação. O primeiro consiste no depósito de feromônios sobre os componentes da solução sendo feito por uma ou mais formigas conforme a estratégia adotada e o segundo, na diminuição da concentração de feromônios.

Guntsch (2004) afirma que no P-ACO a atualização das trilhas de feromônios é feita de acordo com uma população de soluções, ou seja, um determinado conjunto de soluções que são construídas pelas formigas e armazenadas em um arquivo P. Desta forma, quando uma nova solução é inserida no arquivo de soluções, é feito o depósito de feromônios sobre os componentes desta solução. Quando uma solução é retirada do arquivo de soluções, é realizado o processo de evaporação sobre os componentes pertencentes a ela, sendo retirada a quantidade de feromônio depositada durante o processo de inserção no arquivo.

Conforme Guntsch (2004), no P-ACO, para a realização do processo de atualização de feromônios são utilizadas duas estruturas de dados que são o arquivo de soluções P e a matriz de feromônios. Quando o P-ACO inicia sua execução, o arquivo de soluções está vazio e a concentração de feromônios inicial dos componentes da matriz de feromônios é igual a τ_0. O número máximo de soluções que o arquivo pode armazenar é K. A cada nova iteração, uma solução σ é inserida no arquivo de soluções e é feita a atualização da matriz de feromônios por meio do depósito de feromônios sobre a mesma, utilizado-se os componentes da solução σ. A forma geral da concentração de feromônios na matriz de feromônios usada pelo P-ACO é representada pela equação (3.11):

$$\tau_{ij} = \tau_0 + \Delta \sum_{k=1}^{|P|} w_k . I_{ij}^k \tag{3.11}$$

Na equação (3.11), I_{ij}^k é uma variável de decisão que assume o valor 1 se o componente (i, j) pertence à solução gerada pela formiga k e 0, caso contrário. A concentração de feromônios no componente (i, j) é representado por $\sum_{k=1}^{|P|} w_k$. O valor de Δ é definido por meio de equação (3.12):

$$\Delta = \frac{\tau_{max} - \tau_0}{K} \tag{3.12}$$

Nessa equação, τ_{max} é um parâmetro utilizado no algoritmo para determinar a quantidade máxima de feromônios nos componentes das soluções. Sempre que é realizado o depósito de feromônios, a matriz de feromônios é atualizada por meio da adição de um peso $\Delta.w_k$ sobre os componentes de σ. Quando uma nova solução é adicionada ao arquivo de soluções e o mesmo está cheio, ou seja, $|P| = K$, uma dentre as soluções armazenadas no arquivo é substituída pela solução a ser inserida no

mesmo. Com isso, é aplicado sobre todos os componentes da solução que é retirada de P o processo de evaporação, que consiste na remoção de uma quantidade de feromônios igual a $-\Delta.w_k$ que é equivalente a depositada quando tal solução foi inserida no arquivo, porém com Δ assumindo valor negativo.

Outro importante fator que influência o desempenho do P-ACO é a estratégia utilizada para gerenciar o arquivo de soluções, ou seja, a forma e o momento em que as soluções entram e saem de tal arquivo. Diferentes estratégias foram proposta para gerenciar o arquivo de soluções do P-ACO. Segundo Oliveira et al. (2011), dentre as estratégias existentes, pode-se destacar as estratégias *Age-based Strategy*, *Quality-based Strategy* e *Elitist-based Strategy*, apresentadas a seguir.

Age-based Strategy

Na *Age-based Strategy*, o controle do arquivo de soluções funciona de maneira semelhante a da estrutura de dados Fila, em que o primeiro elemento que entra é o primeiro que sai (*First in First out*). Assim, a cada iteração, melhor solução da iteração é armazenada no arquivo de soluções e o depósito de feromônios é feito sobre os componentes dessa solução. Caso o arquivo de soluções esteja cheio, a nova solução substitui a solução que está há mais tempo em P e a remoção de feromônios é aplicada nos componentes da solução que é retirada.

Quality-based Strategy

O gerenciamento do arquivo de soluções no *Quality-based Strategy* trabalha, de maneira que, a cada iteração a melhor solução da iteração só pode entrar no arquivo de soluções se ela for melhor do que a pior solução armazenada em P. Quando isso ocorre é feito o depósito de feromônios nos componentes da nova solução. Se a melhor solução da iteração é pior do que todas as soluções contidas em P, o arquivo de soluções não é alterado. Quando o arquivo de soluções está cheio, a nova solução substitui a pior solução armazenada em P e o processo de evaporação é aplicado nos componentes da solução substituída.

Elitist-based Strategy

Na *Elitist-based Strategy*, quando uma nova solução global é encontrada, a atualização elitista das trilhas de feromônio é feita. Nessa estratégia, enquanto o arquivo de soluções não está cheio, toda nova solução global (solução elitista) encontrada é armazenada no arquivo de soluções, substituindo a solução elitista anterior. A solução elitista que é substituída pela nova solução global é inserida no arquivo de soluções como se fosse a melhor solução da iteração de acordo com as regras de atualização da *Age-based Strategy*. Segundo Guntsch (2004), isto é feito para que a nova solução elitista não seja sujeita às mesmas regras de atualização que são empregadas sobre as demais soluções do arquivo e para que a mesma não seja perdida durante o processo de busca do algoritmo.

Se o arquivo de soluções estiver cheio, toda vez que é encontrada uma nova solução global e ela substitui a solução elitista anterior, a atualização elitista é empregada. Na atualização elitista, as trilhas de feromônio referentes a nova solução global são atualizadas com um peso igual a $w_e.\tau_{max}$, em que o valor de w_e está no intervalo

entre 0 e 1. Os componentes das demais soluções do arquivo são atualizados com um peso igual a $\Delta.(1 - w_e)/(K - 1)$. Quando a melhor solução da iteração não é melhor do que a solução global, ela é inserida no arquivo de soluções conforme as regras de atualização da *Age-based Strategy* e a solução global permanece inalterada.

Capítulo 4

Algoritmos Desenvolvidos

4.1 Representação de Uma Solução

Uma solução para o PRAVJT é representada por um vetor solução que armazena uma permutação de cidades, numeradas de 1 a n, e separadas de acordo com o número de rotas criadas. O valor 0 (zero) é utilizado como elemento separador, e indica o depósito. Cada rota percorrida por um veículo é chamada de pétala. Como exemplo, considere uma região onde existem 9 cidades que podem ser atendidas por 3 caminhões. Um exemplo para o vetor solução para este caso seria S= [0, 2, 5, 7, 0, 3, 6, 4, 0, 8, 1, 9], em que [0, 2, 5, 7], [0, 3, 6, 4] e [0, 8, 1, 9] são as rotas (pétalas) desta solução.

4.2 Estruturas de Vizinhança

O espaço de busca consiste no conjunto de soluções $s \in S$ que pertencem ao PRAVJT, em que se procura obter novas soluções por meio de movimentos de troca e realocação para a geração de vizinhos de s. Neste trabalho foram utilizadas dez diferentes estruturas de vizinhança para se explorar o espaço de busca. Com isso, são utilizados movimentos que consistem na modificação de apenas uma rota (intra-rotas) e movimentos que alteram duas rotas (inter-rotas). Os movimentos intra-rota são referentes às estruturas de vizinhança do tipo *Shift*'(k) e *Exchange*. Os movimentos inter-rota são relativos às estruturas de vizinhança do tipo *Shift*(k,0) e *Swap*(k,l). Todos os movimentos utilizados foram desenvolvidos de maneira a aceitar apenas soluções viáveis.

Além dos movimentos descritos anteriormente, foram também utilizadas duas estratégias para alterar as soluções, a saber: a ER(Elimina Rota) e a ERFO (Elimina Rota Função Objetivo).

4.2.1 Movimentos das Estruturas de Vizinhança *Shift*'(k) e *Exchange*

Os movimentos do tipo *Shift*'(k) consistem na transferência de k clientes adjacentes para outra posição da rota à qual pertencem. Os movimentos relativos a este tipo de estrutura de vizinhança que foram implementados são o *Shift*'(1), *Shift*'(2) e

Shift'(3). Outro movimento intra-rotas que também foi implementado é o *Exchange*, que consiste na permutação entre dois clientes de uma mesma rota.

No movimento *Shift*'(1), apenas um cliente é retirado de uma rota e depois inserido em outra posição da mesma. Na Figura 4.1 é exibida a aplicação do *Shift*'(1), em que o cliente 4 da rota 1 é reinserido na posição do arco (1,14).

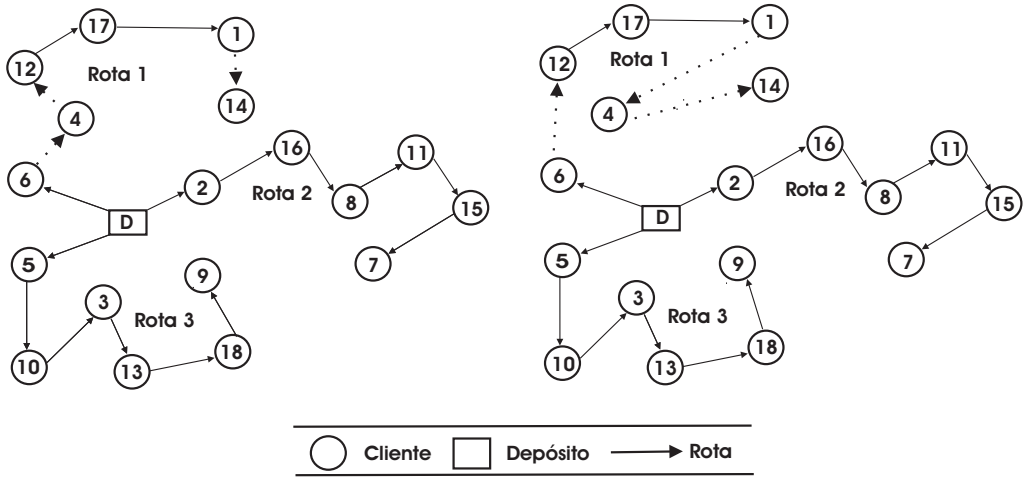


Figura 4.1: Movimento *Shift*'(1).

No movimento *Shift*'(2) são realocados dois clientes consecutivos de uma rota para outra posição da mesma. A Figura 4.2 apresenta a realocação dos clientes 4 e 12 da rota 1 na posição do arco (1,14).

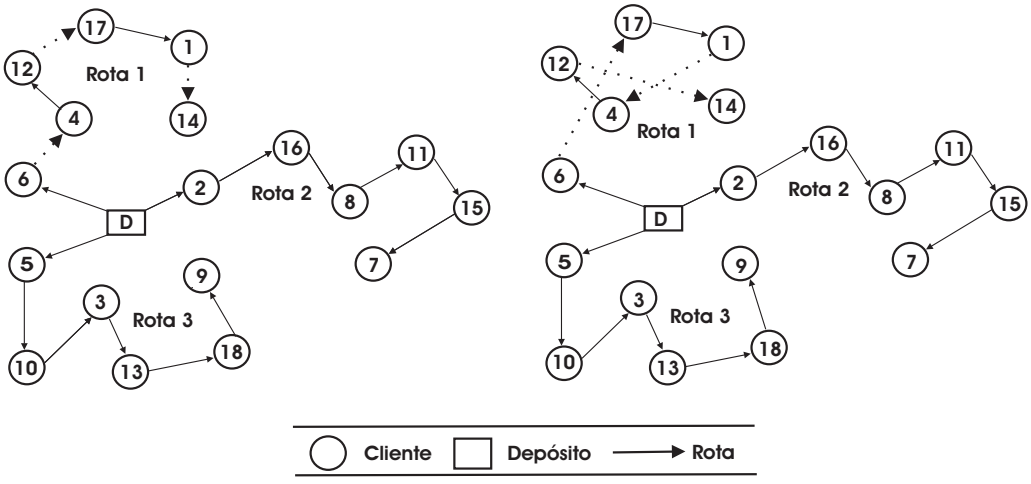


Figura 4.2: Movimento *Shift*'(2).

Já no movimento *Shift*'(3) são realocados dois clientes consecutivos de uma rota para outra posição de tal rota. A Figura 4.3 ilustra a realocação dos clientes 4, 12 e 17 da rota 1 na posição do arco (1,14).

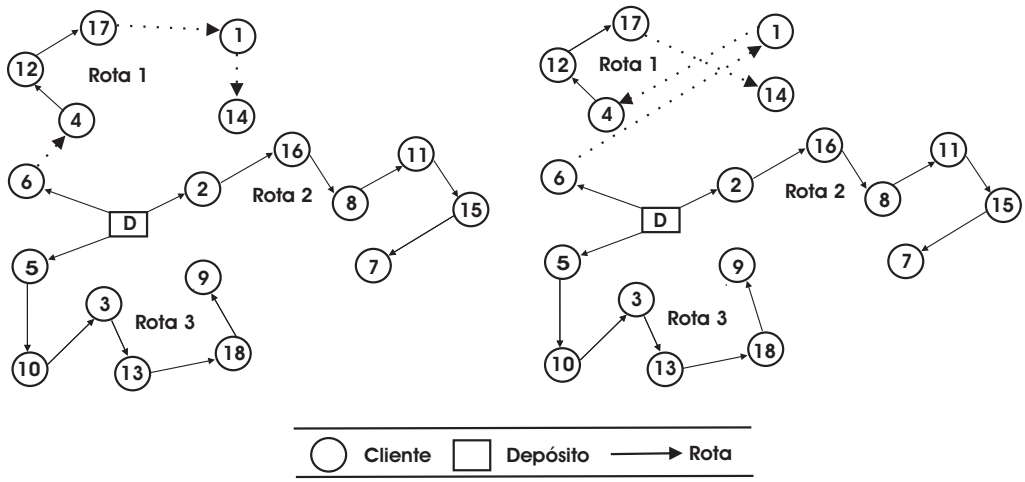


Figura 4.3: Movimento *Shift*'(3).

O movimento *Exchange* tem a função de realizar a troca de dois clientes de uma mesma rota. A Figura 4.4 mostra a troca entre os 3 e 9 da rota 3.

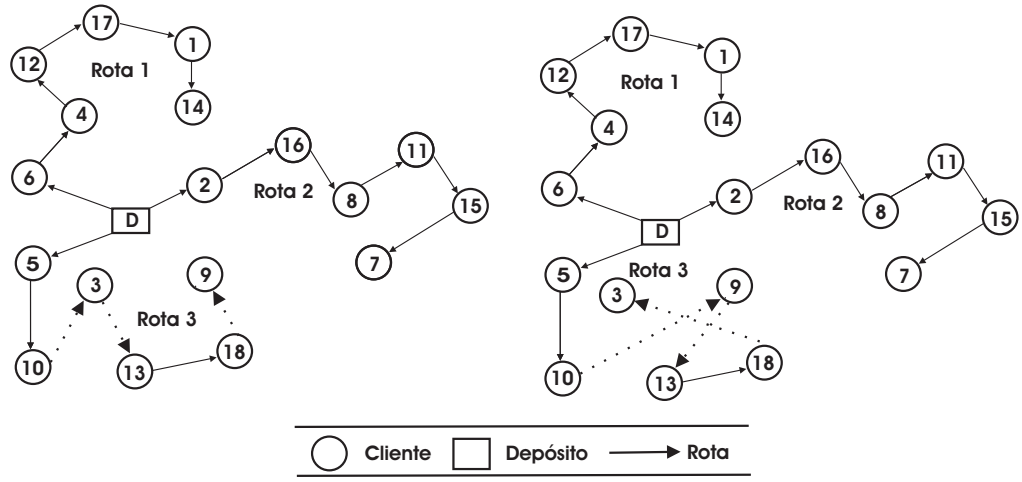


Figura 4.4: Movimento *Exchange*.

4.2.2 Movimentos da Estrutura de Vizinhança *Shift*(k,0)

Os movimentos do tipo *Shift*(k,0) realizam a alteração de duas rotas por meio da realocação de k clientes adjacentes de uma rota para outra. Os movimentos deste tipo de estrutura que foram desenvolvidos são o *Shift*(1,0), *Shift*(2,0) e o *Shift*(3,0).

O movimento *Shift*(1,0) consiste na realocação de um cliente de uma rota para outra. Na Figura 4.5 é mostrado um exemplo em que o cliente 9 da rota 3 é realocado para rota 2.

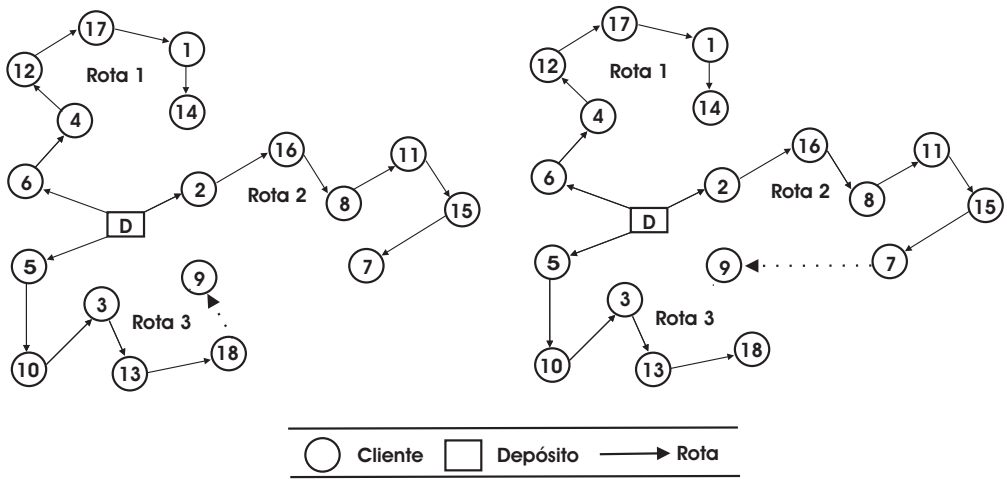


Figura 4.5: Movimento *Shift*(1,0).

O movimento *Shift*(2,0) se baseia na realocação de dois clientes adjacentes de uma rota para outra. Na Figura 4.6 é ilustrado um exemplo em que os clientes 18 e 9 da rota 3 são realocados para rota 2.

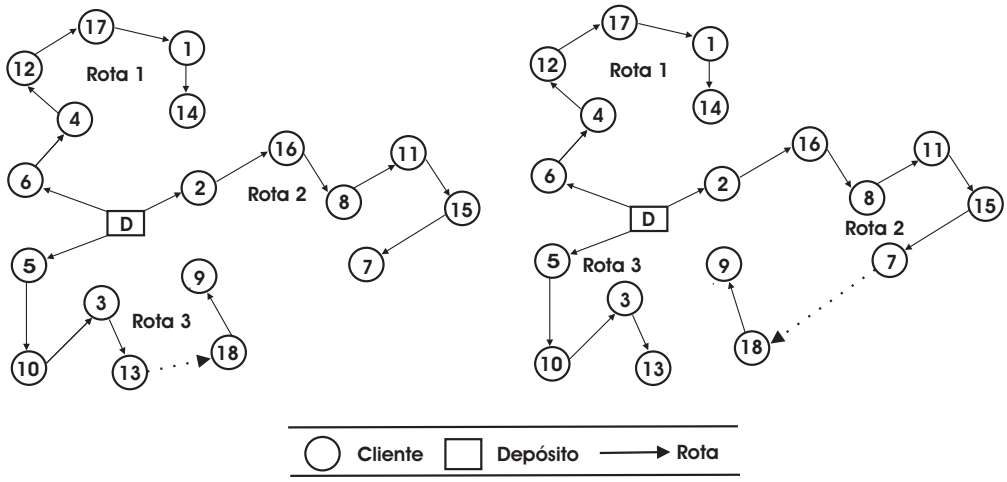


Figura 4.6: Movimento *Shift*(2,0).

Finalmente, no movimento *Shift*(3,0) é feita a realocação de 3 clientes adjacentes de uma rota para outra. Na Figura 4.7 é apresentado um exemplo em que os clientes 17, 1 e 14 da rota 1 são realocados para rota 2.

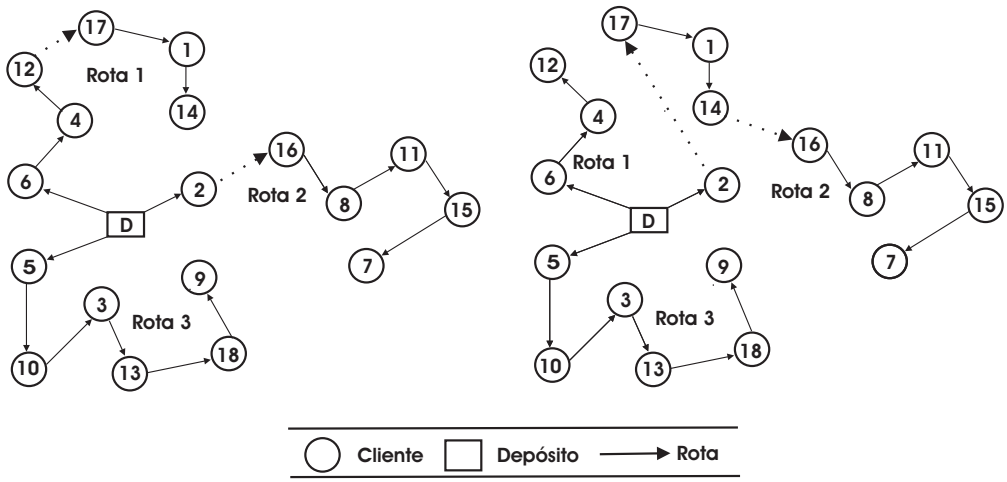


Figura 4.7: Movimento *Shift*(3,0).

4.2.3 Movimentos da Estrutura de Vizinhança *Swap*(k,l)

Os movimentos do tipo *Swap*(k,l) também alteram duas rotas quando realizados, permutando k clientes adjacentes de uma rota com l clientes adjacentes de outra. Neste trabalho foram implementados os movimentos *Swap*(1,1), *Swap*(2,1) e *Swap*(2,2).

O movimento *Swap*(1,1) faz a troca de um cliente pertencente a uma rota r_a com outro cliente de uma rota r_b. Na Figura 4.8 pode ser vista uma ilustração em que o cliente 16 da rota 2 é trocado com o cliente 14 da rota 1.

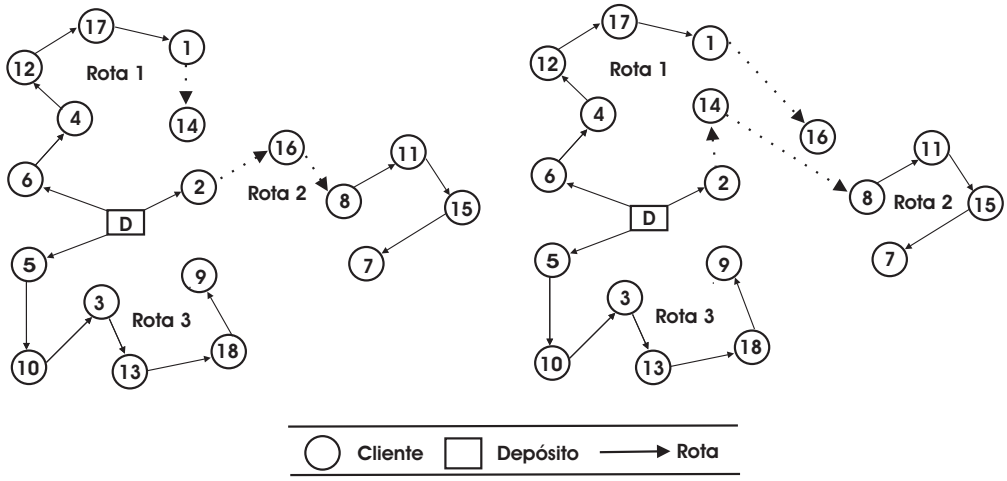


Figura 4.8: Movimento *Swap*(1,1).

Em um movimento do tipo *Swap*(2,1) são trocados dois clientes consecutivos de uma rota com um cliente de outra. É exemplificada na Figura 4.9 a troca dos clientes 1 e 14 da rota 1 com o cliente 2 da rota 2.

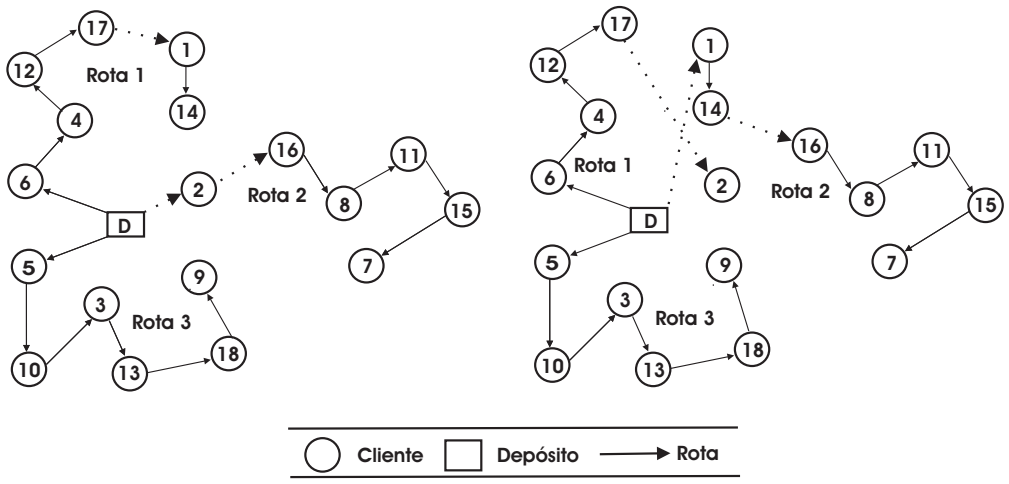


Figura 4.9: Movimento *Swap*(2,1).

Quando o movimento *Swap*(2,2) é aplicado, ele realiza a troca de dois clientes consecutivos de uma rota por dois clientes de outra. A aplicação do movimento *Swap*(2,2) é ilustrada pela Figura 4.10 que mostra a troca dos clientes 1 e 14 da rota 1 com os clientes 2 e 16 da rota 2.

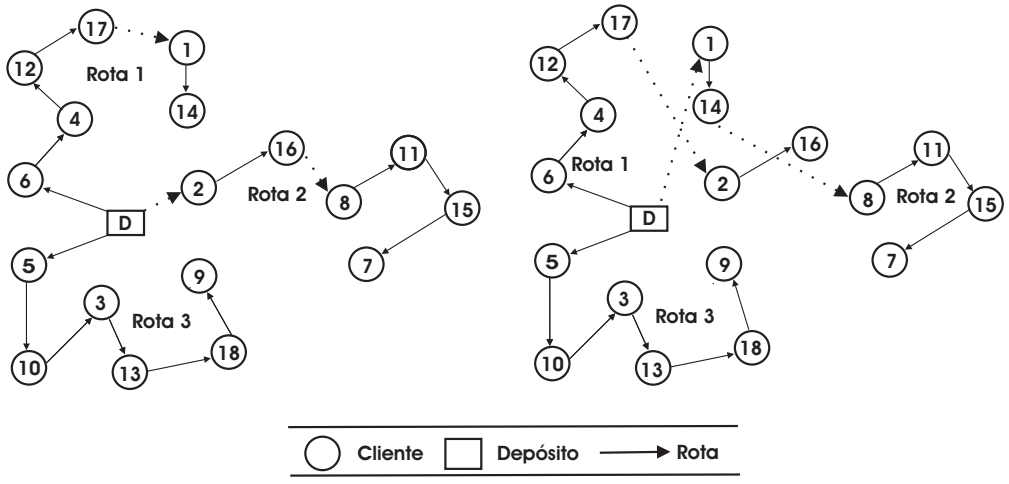


Figura 4.10: Movimento *Swap*(2,2).

4.2.4 Estratégias de Eliminação de Rotas

A estratégia ER (Elimina Rota) consiste em tentar eliminar uma rota da solução. Primeiramente se escolhe a menor rota para tentar eliminá-la por meio da retirada dos clientes pertencentes a ela. Com isso, se busca inserir os clientes removidos da rota escolhida inserindo-os nas demais rotas da solução, dando-se preferência para as maiores rotas. Ao final é escolhida a posição para realocar que resultar na obtenção do melhor valor da função objetivo. Se todos os clientes retirados da rota escolhida forem inseridos nas demais rotas, a nova solução é definida como a corrente e o procedimento

termina. Quando não é possível inserir um dos clientes removidos da rota escolhida em nenhuma das outras rotas, tal rota permanece inalterada e o processo é aplicado para as demais rotas até que seja minimizado o número de rotas ou quando todas as rotas, desde da menor para a maior, forem também analisadas para a sua eliminação. A estratégia ERFO (Elimina Rota Função Objetivo) é semelhante a ER, porém o processo termina quando todos os clientes da rota escolhida são retirados e inseridos nas demais rotas ou o valor da função objetivo é reduzido. O ERFO também termina quando todas as rotas a partir da menor para a maior são analisadas para serem eliminadas.

Tanto na estratégia ER como na ERFO, quando é feita a tentativa de reinserção de um cliente i e não é possível a sua inserção em nenhuma das rotas restantes, é feita a sua troca com um cliente j pertencente a uma das demais rotas, desde que tal operação de troca não viole nenhuma das restrições de capacidade e de tempo. Ao ser feita tal troca, se procura realocar o cliente j que foi trocado com i em outra posição da solução. É testado tal procedimento de troca entre o cliente i e todos os clientes das rotas restantes. Se houver a possibilidade de troca do cliente i, de forma que o cliente j trocado com ele seja colocado em outra posição da solução, tal troca é aplicada e os clientes i e j são reinseridos em suas novas posições; caso contrário, a solução permanece inalterada e é analisado o próximo cliente j. Quando é possível realizar este tipo de troca seguida por reinserção entre o cliente i e dois ou mais clientes pertencentes à solução, é escolhido o cliente que ao ser submetido junto com i a tal procedimento gera o menor impacto sobre o valor da função objetivo. Se não for possível a aplicação deste procedimento entre o cliente i e nenhum dos outros clientes, é criada uma nova rota e i é atendido por ela.

As Figuras 4.12 e 4.13 mostram exemplos da aplicação das estratégias ER e ERFO sobre uma solução s, que é exibida na Figura 4.11, formada por 9 clientes que são atendidos por 3 diferentes veículos. Na Figura 4.11 pode-se visualizar a retirada da rota 3 da solução, em que os clientes 5, 3 e 9 pertencentes a ela são designados para serem reinseridos em s. Por meio da Figura 4.12 é mostrado como é realizada a inserção dos clientes da rota 3 nas demais rotas da solução da Figura 4.11 utilizando a estratégia ER. Após um cliente da rota retirada ser escolhido, aleatoriamente, para ser reinserido, são analisadas todas a possíveis posições de inserção de tal cliente na solução. Neste exemplo da aplicação da estratégia ER são inseridos na solução os clientes 3, 9 e 5, nesta ordem que é definida de modo aleatório. Como foi possível a inserção de todos os três clientes nas demais rotas, resultando na eliminação de umas delas, a solução obtida é aceita e o processo é finalizado.

A Figura 4.13 ilustra a aplicação da estratégia ERFO sobre a solução da Figura 4.11. Considerando que neste caso a rota retirada da solução foi a 3 também, a estratégia ERFO é aplicada sobre a solução s, sendo feita a inserção dos clientes 5, 3 e 9 em s, nesta sequência definida aleatoriamente. Cada vez que é feita a reinserção de um dos clientes da rota retirada, assim como na estratégia ER, são analisadas todas as possíveis posições de reinserção do cliente selecionado. Como neste exemplo a ordem de inserção dos clientes foi diferente da definida no primeiro caso, os mesmos foram inseridos em posições diferentes da solução. Conforme o ilustrado, ao ser feita a tentativa de inserção do cliente 9, não foi possível inseri-lo em nenhuma das rotas restantes, houve a necessidade de se criar uma terceira rota para atendê-lo, impossibilitando a redução do número de rotas. Se neste exemplo da aplicação da

estratégia ERFO, o valor da função objetivo da solução gerada ao final fosse menor do que o da solução original, a nova solução seria aceita e o processo terminaria.

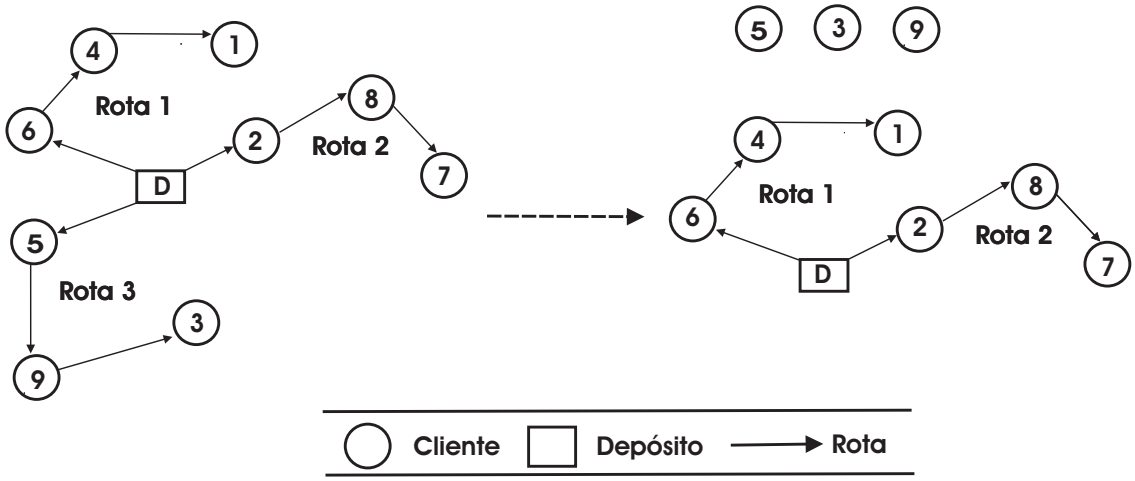


Figura 4.11: Retirada de uma rota da solução.

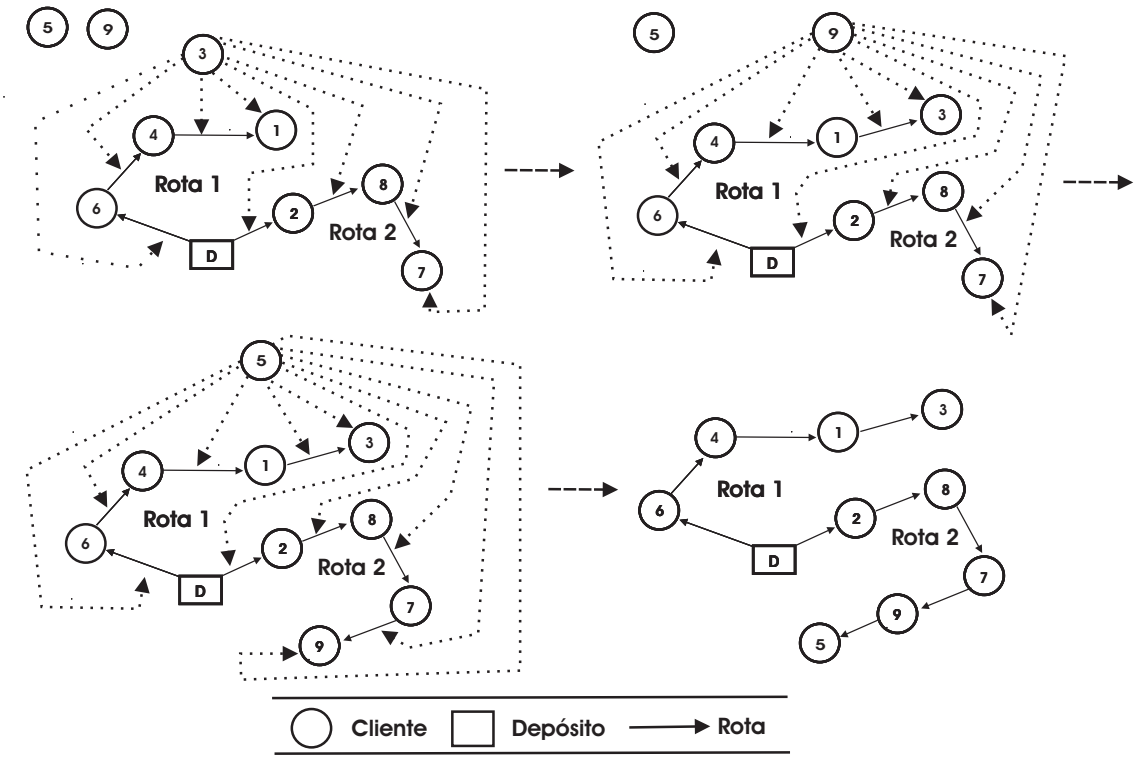


Figura 4.12: Estratégia Elimina Rota.

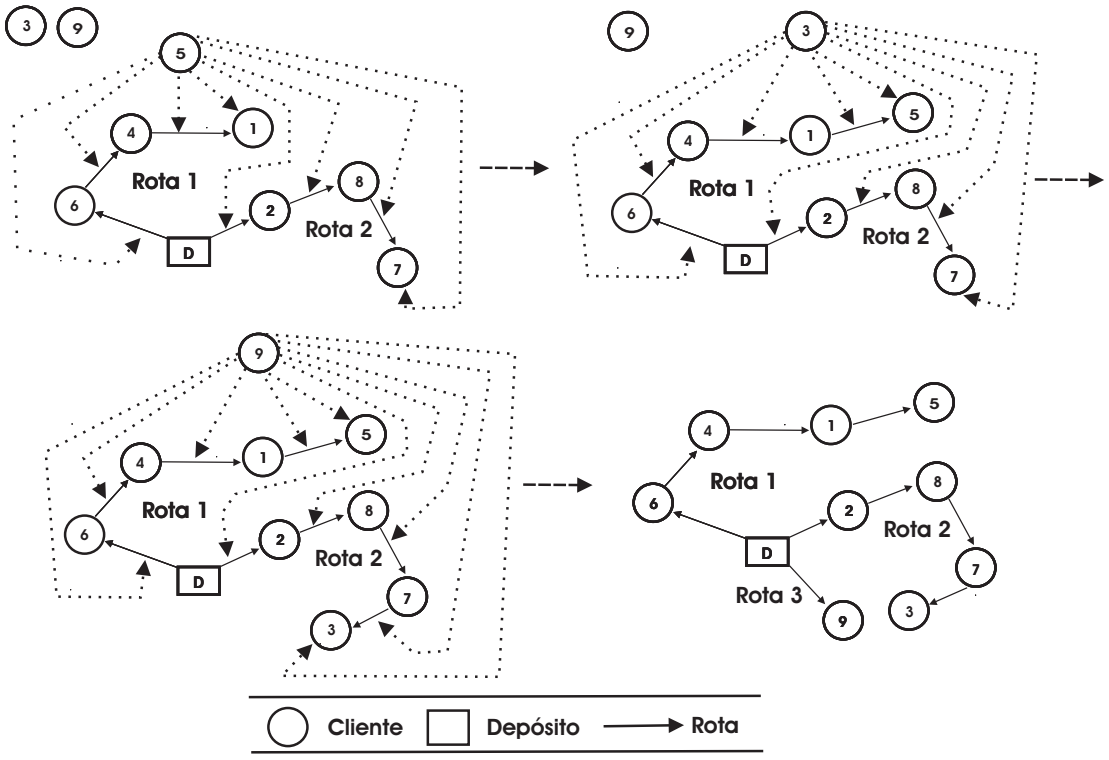


Figura 4.13: Estratégia Elimina Rota Função Objetivo.

4.3 Função de Avaliação

Uma solução $s \in S$ é avaliada pela função representada pela expressão

$$f(s) = \sum_{(ij)\in A} d_{ij}x_{ij} \tag{4.1}$$

que calcula o custo de deslocamento dos veículos, ou seja, a distância total percorrida por todos os veículos em suas respectivas rotas.

Nesta expressão, A é o conjunto das arestas (i, j). O custo de deslocamento de um cliente i para o cliente j é definido por d_{ij}. A variável x_{ij} serve para indicar se o arco (i, j) faz parte da solução, sendo que esta variável recebe o valor 1 caso o arco esteja na solução e 0, caso contrário.

4.4 Algoritmos Propostos

4.4.1 ILS

Nesta implementação, o procedimento de geração da solução inicial do algoritmo baseado em ILS foi feito pela heurística construtiva PFIH. Este algoritmo aplica diferentes níveis de perturbação na solução corrente. Foram utilizados os movimentos

Swap(1,1), *Shift'*(2), *Shift'*(3), *Shift*(1,0), *Shift*(2,0) e *Shift*(3,0). Também foram usadas as estratégias ER e ERFO. O processo de perturbação foi desenvolvido de modo que os movimentos não são escolhidos de forma simplesmente aleatória para realizar tal procedimento; ao invés disso, os movimentos e as estratégias usados para alterar a solução corrente têm um percentual de chance de escolha associado. Deste modo, ao ser feita a escolha do movimento ou estratégia a ser utilizado durante uma iteração, um dentre estes movimentos ou estratégias pode ter um percentual de chance de escolha maior. Esta forma de escolha foi definida, deste modo, para que conforme as características do problema abordado fosse possível dar prioridade de escolha para os movimentos ou estratégias que permitissem explorar o espaço de busca do problema de maneira mais intensiva, mas sem deixar de haver diversificação durante o processo de pesquisa.

Diferentemente da proposta original do ILS, a perturbação implementada consiste em repetir a aplicação de um mesmo movimento r_{max} vezes durante uma iteração, sendo $r_{\max}$ igual a um décimo do número cidades pertencentes ao problema. Isto foi feito para que seja possível analisar mais efetivamente a vizinhança da solução corrente, antes de procurar em outra região do espaço de busca. Assim, ao ser escolhido, um movimento é aplicado $r_{\max}$ vezes sobre s caso não seja obtido um vizinho s' melhor do s durante a aplicação do mesmo.

As perturbações realizadas consistem na aplicação de 2 a 9 movimentos dentre os adotados, ou seja, o número de alterações feitas sobre a solução varia entre 2 e 9. Com isso, quanto maior o nível de perturbação, maior o número de movimentos realizados. No caso das perturbações feitas por meio das estratégias ER ou ERFO, elas são aplicadas apenas uma vez, devido ao fato de seu custo computacional ser maior. A cada iteração, ao ser realizado o processo de perturbação, um movimento é escolhido e aplicado até $r_{\max}$ iterações sem melhora com o nível de perturbação corrente. Caso não seja obtida uma solução melhor do que a atual, na próxima iteração o nível de perturbação é incrementado em uma unidade. O nível de perturbação é aumentado até que se atinja o nível máximo, o que acarreta na reinicialização do nível de perturbações para 2. Quando uma nova solução é encontrada, o nível de perturbação também volta a ser igual a 2.

Após a perturbação, é aplicado o processo de busca local por meio do método VND (*Variable Neighborhood Descent*), em que são utilizados todos os movimentos descritos na seção (4.2), porém não são utilizadas as estratégias ER e o ERFO. Para realizar a busca pelo melhor vizinho no VND é utilizado o método de Descida Randômica para cada vizinhança explorada. A cada iteração do ILS, a solução obtida é avaliada por meio do procedimento de aceitação para definir qual será a próxima solução em que o procedimento de perturbação vai ser aplicado. O Algoritmo 14 apresenta o pseudocódigo da implementação realizada.

Algoritmo 14: ILS

```
   Entrada: iter_max
   Saída: Solucao
 1 início
 2     s_0 ← PFIH();        // Procedimento de geração da solução inicial feito pela heurística PFIH
 3     s ← VND(s_0);                  // Procedimento de busca local feito pela metaheurística VND
 4     iter ← 0;
 5     enquanto (iter < iter_max) faça
 6         iter ← iter + 1;
 7         s' ← Perturbacao(s, historico);
 8         s'' ← VND(s');             // Procedimento de busca local feito pela metaheurística VND
 9         s ← CriterioAceitacao(s, s', s'');
10     fim
11     Retorne Solucao
12 fim
```

4.4.2 GRASP+ILS

O GRASP+ILS se baseia no uso em conjunto das metaheurísticas GRASP e ILS, sendo a primeira responsável pela geração da solução inicial usada pela segunda.

A metaheurística GRASP foi adaptada para o PRAVJT de modo que a fase de construção foi desenvolvida com base na heurística PFIH. A cada iteração, um dos clientes ainda não atendidos é escolhido para ser inserido na solução corrente, de acordo com a ordem definida pela função (3.1) do PFIH. O algoritmo analisa todas as posições em que o cliente pode ser atendido sem que sejam violadas as restrições que compõem o PRAVJT. Todas as possíveis opções referentes às posições de inserção do cliente são armazenadas em uma lista de candidatos de forma ordenada. Estas opções são ordenadas na lista de acordo com o seu benefício, ou seja, valor da função objetivo caso o cliente seja inserido em determinada posição. Com base na função adaptativa gulosa que compõe o procedimento de construção do GRASP, são escolhidos, a partir da lista de candidatos, os melhores elementos (posições) para compor a lista de candidatos restrita (LCR).

Para controlar o nível de gulosidade e aleatoriedade da função de escolha dos elementos da LCR, é utilizado o parâmetro α com valor igual 0,5. O valor do parâmetro α foi definido de forma empírica, em que foram feitos testes para definir os mesmos. Dentre os elementos da lista de candidatos restrita, é escolhido um elemento relativo a uma das posições de inserção na solução para fixar o cliente. A cada iteração do processo de construção, são atualizadas a lista de candidatos e a lista de candidatos restrita. Caso não seja possível a inserção do cliente, é criada uma nova rota. O processo de construção é realizado até que todos os clientes sejam atendidos. Após o processo de construção, é feita a busca local para se pesquisar por um novo ótimo local. Para a realização da busca local sobre as soluções geradas pelo processo de construção, foi utilizado o método de primeira melhora. No método de busca local, são utilizados as estratégias ER e ERFO. Os Algoritmos 15 e 16 descrevem os pseudocódigos do GRASP+ILS.

Algoritmo 15: GRASP

```
   Entrada: f(.), g(.), N(.), GRASPmax, s
   Saída: Solucao
1  início
2      f* ← ∞;
3      for Iter = 1 → GRASPmax do
4          ConstrucaoGRASP_PFIH(g(.), α, s);        // Procedimento de construção baseado no PFIH
5          PrimeiraMelhora(f(.), N(.), s);          // Procedimento de busca local feito pelo método de
           primeira melhora
6          se f(s) < f* então
7              s* ← s;
8              f* ← f(s);
9          fim
10     end
11     s ← s*;
12     Retorne Solucao
13 fim
```

Algoritmo 16: GRASP+ILS

```
   Entrada: iter_max
   Saída: Solucao
1  início
2      s0 ← GRASP();                                // Solução inicial construída pelo algoritmo 15
3      s ← VND(s0);                                 // busca local feita pela metaheurística VND
4      iter ← 0;
5      enquanto (iter < iter_max) faça
6          iter ← iter + 1;
7          s' ← Perturbacao(s, historico);
8          s'' ← VND(s');                           // Procedimento de busca local feito pela metaheurística VND
9          s ← CriterioAceitacao(s, s', s'');
10     fim
11     Retorne Solucao
12 fim
```

4.4.3 HSCM+ILS

O algoritmo HSCM+ILS foi desenvolvido de modo que a metaheurística HSCM tem a função de gerar a solução inicial empregada pelo ILS.

A forma de gerar uma solução inicial é a mesma apresentada na seção 3.1.2. O número de iterações máximo foi definido de maneira empírica, sendo seu valor fixado em 500 iterações. O Algoritmo 17 descreve o pseudocódigo do HSCM+ILS.

Algoritmo 17: HSCM+ILS

```
   Entrada: iter_max
   Saída: Solucao
1  início
2      s0 ← HSCM();                                 // Solução inicial construída pelo algoritmo 3
3      s ← VND(s0);                                 // Procedimento de busca local feito pela metaheurística VND
4      iter ← 0;
5      enquanto (iter < iter_max) faça
6          iter ← iter + 1;
7          s' ← Perturbacao(s, historico);
8          s'' ← VND(s');                           // Procedimento de busca local feito pela metaheurística VND
9          s ← CriterioAceitacao(s, s', s'');
10     fim
11     Retorne Solucao
12 fim
```

4.4.4 *MAX-MIN Ant System*+ILS

O algoritmo *MAX-MIN Ant System*+ILS consiste na combinação das metaheurísticas *MAX-MIN Ant System* e ILS, sendo o primeiro responsável pela geração da solução inicial usada pelo ILS.

No *MAX-MIN Ant System* o modo como as formigas utilizadas constroem as suas soluções foi baseada na heurística construtiva PFIH. De acordo com a ordem de atendimento das cidades definida pela função (3.1) do PFIH, a formiga analisa todas as posições existentes onde o cliente pode ser atendido, sem que sejam violadas as restrições de capacidade e tempo, criando uma lista de candidatos referente a tais posições de inserção. Cada elemento inserido na lista de candidatos é associado com um valor referente ao custo de inserção do cliente na respectiva posição da solução corrente.

Com base na lista de candidatos criada, a formiga escolhe a posição de inserção do cliente de acordo com a expressão (3.3). Na expressão (3.3), τ_{ij} se refere a concentração de feromônios do arco (i, j) a ser usado para ligar o cliente com o depósito ou o cliente anterior a ele, na posição em que é verificada a sua inserção. O valor de η_{ij} é o custo de inserção da cidade em determinada posição da solução. Quando não é possível a inserção do cliente em nenhuma das posições existentes, a formiga cria uma nova rota para atender o cliente.

No *MAX-MIN Ant System*, a cada iteração a solução construída pela melhor formiga é refinada pelo método de primeira melhora utilizando as estratégias ER e ERFO descritas na seção 4.2. O objetivo é tentar obter soluções melhores e com um menor número de rotas.

Os parâmetros α, β, ρ, Q, γ e o número de formigas foram utilizados com os valores 1, 5, 0.5, 100, 0.5 e 10, respectivamente. Tais valores foram definidos por meio de testes empíricos. Os valores dos parâmetros τ_0, τ_{min} e τ_{max} foram definidos de acordo com equações descritas na seção 3.4.2. Os Algoritmos 19 e 18 descrevem os pseudocódigos do *MAX-MIN Ant System*+ILS.

Algoritmo 18: MAX-MIN Ant System+ILS

```
   Entrada: iter_max
   Saída: Solucao
 1 início
 2     s_0 ← MAX − MINAntSystem();          // Solução inicial construída pelo algoritmo 19
 3     s ← VND(s_0);                  // Procedimento de busca local feito pela metaheurística VND
 4     iter ← 0;
 5     enquanto (iter < iter_max) faça
 6         iter ← iter + 1;
 7         s' ← Perturbacao(s, historico);
 8         s'' ← VND(s');             // Procedimento de busca local feito pela metaheurística VND
 9         s ← CriterioAceitacao(s, s', s'');
10     fim
11     Retorne Solucao
12 fim
```

Algoritmo 19: MAX-MIN Ant System

Entrada: $iter_{\max}$, m, $NumCidades$
Saída: $Solucao$
início
  Inicialize os parâmetros $\alpha, \beta, \rho, \tau_{min}$ *e* τ_{max};
  $\tau_0 \leftarrow \tau_{max}, f(s^{gb}) \leftarrow \infty, iter \leftarrow 0$;
  Faça $\Delta\tau_{ij} \leftarrow 0$ $\tau_{ij} \leftarrow \tau_0$ $\forall$ $arco(i,j)$;
  enquanto $(iter < iter_{\max})$ **faça**
    for $k = 1 \rightarrow m$ **do**
      for $c = 1 \rightarrow NumCidades$ **do**
        Selecione a $c-$*ésima cidade conforme a ordem definida pela função 3.1 do PFIH*;
        Formiga k *analisa as posições onde o cliente* c *pode ser atendido*;
        Formiga k *seleciona a posição de inserção do cliente na solução conforme a expressão 3.3*;
      end
    end
    $s^{ib} = argmin(f(s^1), f(s^2), ..., f(s^m))$;
    $PrimeiraMelhora(f(s^{ib}), s^{ib})$; `// busca local feita pelo método de primeira melhora`
    se $f(s^{ib}) < f(s^{gb})$ **então**
      $s^{gb} \leftarrow s^{ib}$;
      $f(s^{gb}) \leftarrow f(s^{ib})$;
      Atualize os parâmetros τ_{min} *e* τ_{max} *de acordo com as equações 3.8 e 3.9, respectivamente*;
    fim
    defina s^{best} *igual a* s^{ib} *ou* s^{gb};
    for $i = 1 \rightarrow NumCidades$ **do**
      for $j = 1 \rightarrow NumCidades$ **do**
        se $arco(i,j)$ *pertence à solução* s^{best} **então**
          $\tau_{ij} = \rho\tau_{ij} + \Delta\tau_{ij}^{best}$;
          se $\tau_{ij} < \tau_{min}$ **então**
            $\tau_{ij} = \tau_{min}$;
          fim
          se $\tau_{ij} > \tau_{max}$ **então**
            $\tau_{ij} = \tau_{max}$;
          fim
        fim
      end
    end
    $iter \leftarrow iter + 1$;
  fim
  Retorne $Solucao$
fim

4.4.5 P-ACO+ILS

Foram geradas três diferentes versões do algoritmo P-ACO. Assim como em Oliveira et al. (2011), foram escolhidas as estratégias *Age-based Strategy*, *Quality-based Strategy* e *Elitist-based Strategy* do P-ACO para a implementação de cada uma dessas três versões. Deste modo, foram criados os algoritmos $P-ACO_{Age}$ (P-ACO *Age-based Strategy*), $P-ACO_{Quality}$ (P-ACO *Quality-based Strategy*) e $P-ACO_{Elitist}$ (P-ACO *Elitist-based Strategy*), baseados nas estratégias *Age*, *Quality* e *Elitist*, respectivamente. A diferença entre tais algoritmos é a forma como o arquivo de soluções e as trilhas de feromônios são atualizados, variando conforme a estratégia adotada por cada um. Foram realizadas diferentes combinações entre o ILS e cada uma das três versões do P-ACO, sendo a solução inicial de tal algoritmo gerada por uma delas.

Os procedimentos de construção, busca local e atualização das estatísticas dos algoritmos $P-ACO_{Age}$, $P-ACO_{Quality}$ e $P-ACO_{Elitist}$ foram desenvolvidos de maneira semelhante à empregada para o *MAX-MIN Ant System*.

As versões do P-ACO desenvolvidas utilizam o procedimento de reinício, sendo que após um certo número de iterações sem melhora a concentração de feromônio em

todos os componentes da matriz de feromônios volta ser igual a τ_0, como no início da execução do algoritmo. O P-ACO, assim como as demais variações do ACO, após um certo número de iterações, devido a alta concentração de feromônio em determinadas trilhas, acaba gerando soluções muito semelhantes ou até mesmo iguais, fazendo com que o P-ACO pare de convergir, sendo necessária a reinicialização da concentração de feromônio das trilhas para que possa haver uma maior diversificação ao ser feito o processo de busca e com isso obter novas e melhores soluções ao final.

Os parâmetros α, β, ρ, Q, τ_{max} e o número de formigas foram definidos de forma empírica. O valor de τ_0 é igual a $\frac{1}{n-1}$, em que n é a quantidade de cidades relativas ao problema abordado. Os valores dos parâmetros adotados para serem usados no $P-ACO_{Age}$, $P-ACO_{Quality}$ e $P-ACO_{Elitist}$ são apresentados na Tabela 4.1.

Tabela 4.1: Valores dos parâmetros definidos para as estratégias do P-ACO.

Parâmetro	$P-ACO_{Age}$	$P-ACO_{Quality}$	$P-ACO_{Elitist}$
α	3	1	1
β	7	6	5
ρ	0.5	0.5	0.5
Q	100	100	100
τ_{max}	6	1	6
Pop Size	25	10	15
Formigas	10	10	10

Os Algoritmos 20 e 21 descrevem os pseudocódigos do $P-ACO_{Age}$+ILS.

Algoritmo 20: $P-ACO_{Age}$+ILS

```
   Entrada: iter_max
   Saída: Solucao
1  início
2      s_0 ← P − ACO_Age();                          // Solução inicial construída pelo algoritmo 21
3      s ← VND(s_0);                  // Procedimento de busca local feito pela metaheurística VND
4      iter ← 0;
5      enquanto (iter < iter_max) faça
6          iter ← iter + 1;
7          s' ← Perturbacao(s, historico);
8          s'' ← VND(s');             // Procedimento de busca local feito pela metaheurística VND
9          s ← CriterioAceitacao(s, s', s'');
10     fim
11     Retorne Solucao
12 fim
```

Algoritmo 21: $P-ACO_{Age}$

Entrada: $iter_{\max}$, m, $NumCidades$
Saída: $Solucao$
início
  $Inicialize\ os\ parâmetros\ \alpha, \beta, \rho, K, iterMAXSemMelhora\ e\ \tau_{max}$;
  $\tau_0 \leftarrow 1/(NumCidades-1), f(s^{gb}) \leftarrow \infty, iter \leftarrow 0, iterSemMelhora \leftarrow 0$;
  Faça $\Delta\tau_{ij} \leftarrow 0\ \tau_{ij} \leftarrow \tau_0\ \forall\ arco(i,j)$;
  enquanto $(iter < iter_{\max})$ **faça**
    for $k = 1 \rightarrow m$ **do**
      for $c = 1 \rightarrow NumCidades$ **do**
        $Selecione\ a\ c-ésima\ cidade\ conforme\ a\ ordem\ definida\ pela\ função\ 3.1\ do\ PFIH$;
        $Formiga\ k\ analisa\ as\ posições\ onde\ o\ cliente\ c\ pode\ ser\ atendido$;
        $Formiga\ k\ seleciona\ a\ posição\ de\ inserção\ do\ cliente\ na\ solução\ conforme\ a\ expressão\ 3.3$;
      end
    end
    $s^{ib} = argmin(f(s^1), f(s^2), ..., f(s^m))$;
    $PrimeiraMelhora(f(s^{ib}), s^{ib})$; `// busca local feita pelo método de primeira melhora`
    se $f(s^{ib}) < f(s^{gb})$ **então**
      $s^{gb} \leftarrow s^{ib}$;
      $f(s^{gb}) \leftarrow f(s^{ib})$;
    fim
    $Armazene\ s^{ib}\ no\ arquivo\ de\ soluções\ P$;
    se $Se\ o\ arquivo\ |P| = K$ **então**
      $Defina\ s^{fi}\ igual\ a\ solução\ mais\ antiga\ pertecente\ a\ P$;
      $Retire\ a\ solução\ armazenada\ há\ mais\ tempo\ em\ P$;
    fim
    for $i = 1 \rightarrow NumCidades$ **do**
      for $j = 1 \rightarrow NumCidades$ **do**
        se $arco(i,j)\ pertence\ à\ solução\ s^{ib}$ **então**
          $\tau_{ij} = \tau_{ij} + \Delta.w_k$;
        fim
        se $arco(i,j)\ pertence\ à\ solução\ s^{fi}$ **então**
          $\tau_{ij} = \tau_{ij} - \Delta.w_k$;
        fim
      end
    end
    $iter \leftarrow iter + 1$;
    se $iterSemMelhora \geq iterMAXSemMelhora$ **então**
      $iterSemMelhora \leftarrow 0$;
      Faça $\Delta\tau_{ij} \leftarrow 0\ \tau_{ij} \leftarrow \tau_0\ \forall\ arco(i,j)$;
    fim
  fim
  Retorne $Solucao$
fim

Os Algoritmos 23 e 22 mostram os pseudocódigos do $P-ACO_{Quality}$+ILS.

Algoritmo 22: $P-ACO_{Quality}$+ILS

Entrada: $iter_{\max}$
Saída: $Solucao$
início
  $s_0 \leftarrow P-ACO_{Quality}()$; `// Solução inicial construída pelo algoritmo 23`
  $s \leftarrow VND(s_0)$; `// Procedimento de busca local feito pela metaheurística VND`
  $iter \leftarrow 0$;
  enquanto $(iter < iter_{max})$ **faça**
    $iter \leftarrow iter + 1$;
    $s' \leftarrow Perturbacao(s, historico)$;
    $s'' \leftarrow VND(s')$; `// Procedimento de busca local feito pela metaheurística VND`
    $s \leftarrow CriterioAceitacao(s, s', s'')$;
  fim
  Retorne $Solucao$
fim

Algoritmo 23: $P - ACO_{Quality}$

```
   Entrada: iter_max, m, NumCidades
   Saída: Solucao
 1 início
 2     Inicialize os parâmetros α, β, ρ, K, iterMAXSemMelhora e τ_max;
 3     τ_0 ← 1/(NumCidades − 1), f(s^gb) ← ∞, iter ← 0, iterSemMelhora ← 0;
 4     Faça Δτ_ij ← 0 τ_ij ← τ_0 ∀ arco(i, j);
 5     enquanto (iter < iter_max) faça
 6         for k = 1 → m do
 7             for c = 1 → NumCidades do
 8                 Selecione a c − ésima cidade conforme a ordem definida pela função 3.1 do PFIH;
 9                 Formiga k analisa as posições onde o cliente c pode ser atendido;
10                 Formiga k seleciona a posição de inserção do cliente na solução conforme a expressão 3.3;
11             end
12         end
13         s^ib = argmin(f(s^1), f(s^2), ..., f(s^m));
14         PrimeiraMelhora(f(s^ib), s^ib);        // busca local feita pelo método de primeira melhora
15         se f(s^ib) < f(s^gb) então
16             s^gb ← s^ib;
17             f(s^gb) ← f(s^ib);
18         fim
19         se Se o arquivo |P| = K então
20             se s^ib é melhor do que a pior solução armazenada em P então
21                 Armazene s^ib no arquivo de soluções P;
22                 Defina s^ws igual a pior solução pertecente a P;
23                 Retire a pior solução armazenada em P;
24                 for i = 1 → NumCidades do
25                     for j = 1 → NumCidades do
26                         se arco(i, j) pertence à solução s^ib então
27                             τ_ij = τ_ij + Δ.w_k;
28                         fim
29                         se arco(i, j) pertence à solução s^ws então
30                             τ_ij = τ_ij − Δ.w_k;
31                         fim
32                     end
33                 end
34             fim
35         fim
36         se Se o arquivo |P| < K então
37             Armazene s^ib no arquivo de soluções P;
38             for i = 1 → NumCidades do
39                 for j = 1 → NumCidades do
40                     se arco(i, j) pertence à solução s^ib então
41                         τ_ij = τ_ij + Δ.w_k;
42                     fim
43                 end
44             end
45         fim
46         iter ← iter + 1;
47         se iterSemMelhora ≥ iterMAXSemMelhora então
48             iterSemMelhora ← 0;
49             Faça Δτ_ij ← 0 τ_ij ← τ_0 ∀ arco(i, j);
50         fim
51     fim
52     Retorne Solucao
53 fim
```

Os Algoritmos 24 e 25 mostram os pseudocódigos do $P-ACO_{Elitist}$+ILS.

Algoritmo 24: $P-ACO_{Elitist}$

Entrada: $iter_{\max}$, m, $NumCidades$
Saída: $Solucao$
início
  $Inicialize\ os\ parâmetros\ \alpha, \beta, \rho, K, iterMAXSemMelhora\ e\ \tau_{max}$;
  $\tau_0 \leftarrow 1/(NumCidades-1), f(s^{gb}) \leftarrow \infty, iter \leftarrow 0, iterSemMelhora \leftarrow 0$;
  Faça $\Delta\tau_{ij} \leftarrow 0\ \tau_{ij} \leftarrow \tau_0\ \forall\ arco(i,j)$;
  enquanto $(iter < iter_{\max})$ **faça**
    for $k = 1 \rightarrow m$ **do**
      for $c = 1 \rightarrow NumCidades$ **do**
        $Selecione\ a\ c-ésima\ cidade\ conforme\ a\ ordem\ definida\ pela\ função\ 3.1\ do\ PFIH$;
        $Formiga\ k\ analisa\ as\ posições\ onde\ o\ cliente\ c\ pode\ ser\ atendido$;
        $Formiga\ k\ seleciona\ a\ posição\ de\ inserção\ do\ cliente\ na\ solução\ conforme\ a\ expressão\ 3.3$;
      end
    end
    $s^{ib} = argmin(f(s^1), f(s^2), ..., f(s^m))$;
    $PrimeiraMelhora(f(s^{ib}), s^{ib})$; `// busca local feita pelo método de primeira melhora`
    se $f(s^{ib}) < f(s^{gb})$ **então**
      $Armazene\ s^{gb}\ no\ arquivo\ de\ soluções\ P$;
      $Atualize\ a\ matriz\ de\ feromônios\ conforme\ a\ Elitist\text{-}Based\text{-}Strategy$;
      $s^{gb} \leftarrow s^{ib}$;
      $f(s^{gb}) \leftarrow f(s^{ib})$;
    fim
    se $f(s^{ib}) \geq f(s^{gb})$ **então**
      $Armazene\ s^{ib}\ no\ arquivo\ de\ soluções\ P$;
      se $Se\ o\ arquivo\ |P| = K$ **então**
        $Defina\ s^{fi}igual\ a\ solução\ mais\ antiga\ pertecente\ a\ P$;
        $Retire\ a\ solução\ armazenada\ há\ mais\ tempo\ em\ P$;
      fim
      for $i = 1 \rightarrow NumCidades$ **do**
        for $j = 1 \rightarrow NumCidades$ **do**
          se $arco(i,j)\ pertence\ à\ solução\ s^{ib}$ **então**
            $\tau_{ij} = \tau_{ij} + \Delta.w_k$;
          fim
          se $arco(i,j)\ pertence\ à\ solução\ s^{fi}$ **então**
            $\tau_{ij} = \tau_{ij} - \Delta.w_k$;
          fim
        end
      end
    fim
    $iter \leftarrow iter + 1$;
    se $iterSemMelhora \geq iterMAXSemMelhora$ **então**
      $iterSemMelhora \leftarrow 0$;
      Faça $\Delta\tau_{ij} \leftarrow 0\ \tau_{ij} \leftarrow \tau_0\ \forall\ arco(i,j)$;
    fim
  fim
  Retorne $Solucao$
fim

Algoritmo 25: $P - ACO_{Elitist}$+ILS

```
   Entrada: iter_max
   Saída: Solucao
1  início
2      s_0 ← P − ACO_Elitist();                    // Solução inicial construída pelo algoritmo 24
3      s ← VND(s_0);                  // Procedimento de busca local feito pela metaheurística VND
4      iter ← 0;
5      enquanto (iter < iter_max) faça
6          iter ← iter + 1;
7          s' ← Perturbacao(s, historico);
8          s'' ← VND(s');             // Procedimento de busca local feito pela metaheurística VND
9          s ← CriterioAceitacao(s, s', s'');
10     fim
11     Retorne Solucao
12 fim
```

Capítulo 5

Experimentos Computacionais

Neste Capítulo são apresentados os experimentos computacionais realizados com as algoritmos desenvolvidos e mostrados no Capítulo 4. A Seção 5.1 descreve as instâncias estudadas para o Problema de Roteamento Aberto de Veículos com Janelas de Tempo. A Seção 5.2 introduz questões gerais a respeito da apresentação dos resultados computacionais. As demais seções apresentam os resultados computacionais relativos à aplicação dos algoritmos desenvolvidos às instâncias de teste utilizadas.

5.1 Instâncias Propostas para o PRAVJT

As instâncias utilizadas para efetuar os testes computacionais nos algoritmos desenvolvidos, apresentados no Capítulo 4, são as propostas por Solomon (1987), adaptadas para o caso em tela do Problema de Roteamento Aberto de Veículos com Janela de Tempo.

Estas instâncias se dividem nos grupos C, R e RC. No conjunto de instâncias C, os clientes são localizados próximos uns dos outros, formando grupos ("*cluster*"). No conjunto de instâncias R, os clientes estão localizados em posições aleatórias, sem agrupamento. No caso do grupo de instâncias RC, há tanto clientes localizados próximos uns dos outros, como no conjunto C, quanto em posições aleatórias, como no conjunto R.

Os conjuntos de instâncias do tipo C1, R1 e RC1 possuem janelas de tempo com intervalos menores, ou seja, o intervalo no qual deve ser realizado o atendimento dos clientes é definido por um curto período de tempo. Deste modo, menos clientes podem fazer parte das rotas a serem atendidas pelos veículos, sendo necessário um maior número de veículos para atender todos os clientes.

Já os conjuntos de instâncias tipo C2, R2 e RC2 possuem janelas de tempo com intervalos largas, de modo que o intervalo de tempo para o atendimento dos clientes é um largo período de tempo. Neste caso, um maior número de clientes pode ser atendido por rota, utilizando-se, portanto, um menor número de veículos para o atendimento do número total de clientes.

Todos estes conjuntos de instâncias possuem $n = 100$ clientes. A capacidade Q dos veículos nos conjuntos de instâncias usadas é igual a 200 unidades, enquanto que nos conjuntos C2, R2 e RC2 a capacidade dos veículos é igual a 700, 100 e 1000 unidades, respectivamente.

Também é utilizado para os testes computacionais o conjunto de instâncias apresentado em Homberger e Gehring (1999). Este conjunto de instâncias mantém as principais características das instâncias de Solomon (1987), porém, o número de clientes que tais instâncias possuem é de 200, 400, 600, 800 e 1000. Na Tabela 5.1, é apresentado um exemplo de como são apresentadas as informações contidas nas instâncias de Solomon.

Tabela 5.1: Informações das instâncias de Solomon

N° Cidades						
Capac. Veículo						
Id. Cidade	Coord. X	Coord. Y	Demanda	Início JT	Fim JT	TS

Nesta Tabela, tem-se que:

- Nº Cidades: representa o número de cidades da instância;
- Cap. Veículo: capacidade de carga máxima do veículo;
- Id. Cidade: número de identificação da cidade;
- Coord. X e Coord. Y: coordenadas geográficas que definem as localizações das cidades;
- Demanda: demanda requerida por uma cidade;
- Inicio JT e Fim JT: intervalo de tempo definido para uma cidade;
- TS: tempo de serviço gasto para se atender em uma cidade.

A Tabela 5.2 ilustra o exemplo de uma instância para o PRAVJT.

Tabela 5.2: Exemplo de uma instância para o PRAVJT

100						
200						
0	35	35	0	0		0
1	41	49	10	161	171	10
2	35	17	7	50	60	10
⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	18	18	17	185	195	10

5.2 Definições para os Resultados Computacionais

Os algoritmos propostos foram desenvolvidos em linguagem C++, usando o compilador gnu GCC. Os algoritmos desenvolvidos foram executados trinta vezes para cada uma das instâncias utilizadas em um computador com processador Pentium Intel(R) Core(TM)2 Quad Q8400, com clock de 2,66 GHz, memória RAM de 3,7 GB, sob a plataforma Windows Seven Ultimate. Todos os algoritmos foram testados

utilizando as 56 instâncias de Solomon (1987). O algoritmo MMAS+ILS também foi aplicado sobre os conjuntos de instâncias de Homberger e Gehring (1999).

São apresentados nas tabelas de resultados o desvio da melhor solução (DMS) e o desvio do resultado médio (DRM). O DMS se refere à melhora dos resultados obtidos em relação ao melhor resultado presente na literatura. O DRM equivale ao desvio da média das soluções obtidas em relação à melhor solução da literatura. O DMS e o DRM são calculados de acordo com as expressões 5.1 e 5.2, respectivamente.

$$DMS = \frac{fo^* - \min\limits_{i=1,\cdots,30}\{fo_i\}}{fo^*} \tag{5.1}$$

$$DRM = \frac{fo^* - \frac{1}{30}\left(\sum\limits_{i=1}^{30} fo_i\right)}{fo^*} \tag{5.2}$$

Nestas expressões, fo^* refere-se ao melhor resultado encontrado por Repoussis et al. (2009), valores de referência para esta dissertação, e fo_i ao resultado encontrado na i-ésima execução do algoritmo proposto. Observe que valores negativos para DMS e DRM indicam que os resultados encontrados estão acima dos valores encontrados em Repoussis et al. (2009).

As tabelas que vão de 5.15 a 5.14 apresentam, para cada uma das implementações feitas, os dados referentes aos resultados obtidos através destas implementações. Em todas as tabelas, a coluna 1 refere-se ao nome da instância e os dados das colunas 2 e 3 são relativos à distância total e ao número de veículos da melhor solução presente na literatura, respectivamente. As demais colunas apresentam os dados acerca dos resultados obtidos pelo algoritmo desenvolvido, ao qual a tabela se refere, que são a Distância Total (DT), Número de Veículos (NV), Média da Distância Total (MDT), Desvio da Melhor Solução (DMS) e Desvio do Resultado Médio (DRM), respectivamente.

Os valores em negrito mostram os resultados que são superiores aos encontrados em Repoussis et al. (2009). Os valores são comparados considerando-se, em primeiro lugar, o número de veículos da solução e, em seguida, a distância total percorrida, de modo que melhores soluções apresentam menor número de veículos e, após, menores distâncias totais.

5.3 Resultados para o ILS

Ao ser testado, utilizando o conjunto de instâncias R1, o algoritmo ILS apresentou soluções melhores que as apresentadas na literatura em 50% das instâncias testadas. O algoritmo ILS também foi aplicado sobre o conjunto R2. No entanto, apesar de ter apresentado resultados próximos, o ILS não foi capaz de encontrar soluções melhores que as existentes na literatura neste caso. Quando foi usado para solucionar os conjuntos C1 e C2, o ILS encontrou novas soluções em 100% e 75% dos instâncias testadas, respectivamente. O algoritmo ILS demonstrou também ser eficiente ao ser

testado para solucionar as instâncias do conjunto RC1, encontrando soluções melhores que as da literatura em 75% dos casos. Já para o conjunto RC2, o ILS apresentou resultados melhores para apenas 12, 5% dos casos. Os resultados obtidos pelo ILS ao ser aplicado sobre as instâncias de Solomon são apresentados pelas Tabelas 5.3 a 5.8.

Tabela 5.3: Resultados para o conjunto de instâncias R1 de Solomon

	Repoussis (2009)		**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R101	1192,85	19	**1195,13**	**18**	1198,853	-0,002	-0,005
R102	1079,39	17	**1046,4**	**17**	1050,547	0,031	0,027
R103	1016,78	13	**986,16**	**13**	978,302	0,030	0,038
R104	869,63	9	**801**	**9**	821,050	0,079	0,056
R105	1055,04	14	**1082,37**	**13**	1077,518	-0,026	-0,021
R106	1000,95	12	**988,3**	**12**	1009,312	0,013	-0,008
R107	912,99	10	925,5	10	936,445	-0,014	-0,026
R108	760,3	9	787,6	9	763,125	-0,036	-0,004
R109	934,53	11	981,45	11	931,178	-0,050	0,004
R110	846,49	10	883,08	10	863,410	-0,043	-0,020
R111	895,21	10	906,85	10	899,670	-0,013	-0,005
R112	811,73	9	753,07	10	783,365	0,072	0,035

Tabela 5.4: Resultados para o conjunto de instâncias C1 de Solomon

	Repoussis (2009)		**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C101	556,18	10	**556,03**	**10**	558,329	0,0003	-0,004
C102	556,18	10	**556,03**	**10**	570,724	0,0003	-0,026
C103	556,18	10	**556,03**	**10**	577,914	0,0003	-0,039
C104	555,41	10	**555,27**	**10**	619,601	0,0003	-0,116
C105	556,18	10	**556,03**	**10**	556,030	0,0003	0,0003
C106	556,18	10	**556,03**	**10**	567,016	0,0003	-0,019
C107	556,18	10	**556,03**	**10**	567,016	0,0003	-0,019
C108	555,8	10	**555,65**	**10**	566,971	0,0003	-0,020
C109	555,8	10	**555,65**	**10**	584,334	0,0003	-0,051

Tabela 5.5: Resultados para o conjunto de instâncias RC1 de Solomon

	Repoussis (2009)		**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC101	1227,37	14	**1131,64**	**14**	1147,581	0,078	0,065
RC102	1203,05	12	1043,87	13	1079,016	0,132	0,1031
RC103	923,15	11	**920,89**	**11**	939,693	0,002448	-0,018
RC104	787,02	10	801,52	10	826,6773	-0,018	-0,050
RC105	1195,2	13	**1052,92**	**13**	1067,507	0,119	0,107
RC106	1095,65	11	**1004,88**	**11**	993,331	0,083	0,093
RC107	861,28	11	**844,95**	**11**	894,114	0,019	-0,038
RC108	831,09	10	**804,79**	**10**	840,350	0,032	-0,011

Tabela 5.6: Resultados para o conjunto de instâncias R2 de Solomon

	Repoussis (2009)		**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R201	1182,43	4	1214,68	4	1264,276	-0,027	-0,069
R202	1149,59	3	1038,07	4	1071,988	0,097	0,068
R203	889,12	3	933,96	3	925,466	-0,050	-0,041
R204	801,46	2	839,52	2	781,899	-0,048	0,024
R205	943,33	3	995,76	3	1035,032	-0,056	-0,097
R206	869,27	3	895,74	3	934,980	-0,031	-0,076
R207	857,08	2	795,54	3	824,702	0,072	0,038
R208	700,53	2	725,69	2	762,633	-0,036	-0,089
R209	851,69	3	855,15	3	985,922	-0,004	-0,158
R210	892,45	3	922,8	3	965,811	-0,034	-0,082
R211	886,9	2	771,89	3	809,558	0,130	0,087

Tabela 5.7: Resultados para o conjunto de instâncias C2 de Solomon

	Repoussis (2009)		**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C201	548,51	3	**548,3**	**3**	548,3	0,0004	0,0004
C202	548,51	3	**548,3**	**3**	550,274	0,0004	-0,003
C203	548,13	3	635,74	3	760,761	-0,160	-0,388
C204	547,55	3	651,1	3	697,035	-0,189	-0,273
C205	545,83	3	**545,61**	**3**	545,610	0,0004	0,0004
C206	545,45	3	**545,22**	**3**	597,256	0,0004	-0,095
C207	545,24	3	**545,01**	**3**	554,331	0,0004	-0,017
C208	545,28	3	**545,05**	**3**	546,749	0,0004	-0,003

Tabela 5.8: Resultados para o conjunto de instâncias RC2 de Solomon

	Repoussis (2009)		ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC201	1303,73	4	1345,6	4	1378,063	-0,032	-0,057
RC202	1321,43	3	1112,3	4	1134,795	0,158	0,141
RC203	993,29	3	1027,37	3	1079,595	-0,034	-0,087
RC204	718,97	3	796,82	3	852,841	-0,108	-0,186
RC205	1189,84	4	**986,9**	**3**	1044,667	0,171	0,122
RC206	1091,79	3	1118,31	3	1173,076	-0,024	-0,074
RC207	998,7	3	1030,02	3	106,766	-0,031	0,893
RC208	769,4	3	835,93	3	890,128	-0,086	-0,156

5.4 Resultados para o GRASP+ILS

O algoritmo GRASP+ILS foi capaz de encontrar novas soluções para mais de 58,33% das instâncias do conjunto R1, apesar de não ter encontrado soluções melhores do que as apresentadas na literatura para o conjunto R2, os resultados alcançados pelo algoritmo foram próximos. Para os conjuntos de instâncias C1 e C2, o GRASP+ILS foi capaz de encontrar novas soluções em 100% e 87,5% das instâncias testadas, respectivamente. Ao ser aplicado sobre os problemas-teste das instâncias RC1 e RC2, o GRASP+ILS apresentou novas soluções em 87,5% e 12,5% das instâncias testadas, respectivamente. Nas tabelas que vão desde a 5.9 até a 5.14 apresentam os resultados alcançados pelo GRASP+ILS ao ser aplicado sobre as instâncias de Solomon.

Tabela 5.9: Resultados para o conjunto de instâncias R1 de Solomon

	Repoussis (2009)		GRASP+ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R101	1192,85	19	**1195,13**	**18**	1199,365	-0,002	-0,006
R102	1079,39	17	**1046,29**	**17**	1050,915	0,031	0,030
R103	1016,78	13	**984,97**	**13**	1000,117	0,031	0,020
R104	869,63	9	**791,43**	**9**	799,878	0,090	0,080
R105	1055,04	14	**1123,85**	**13**	1115,668	-0,065	-0,060
R106	1000,95	12	**994,82**	**12**	1006,872	0,006	-0,006
R107	912,99	10	921,37	10	907,565	-0,009	0,006
R108	760,30	9	777,60	9	773,134	-0,023	-0,017
R109	934,53	11	954,19	11	935,387	-0,021	-0,001
R110	846,49	10	904,16	10	878,7697	-0,068	-0,040
R111	895,21	10	**887,21**	**10**	901,427	0,009	-0,007
R112	811,73	9	750,21	10	791,091	0,076	0,030

Tabela 5.10: Resultados para o conjunto de instâncias C1 de Solomon

	Repoussis (2009)		**GRASP+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C101	556,18	10	**556,03**	**10**	559,384	0,0003	-0,006
C102	556,18	10	**556,03**	**10**	559,945	0,0003	-0,007
C103	556,18	10	**556,03**	**10**	579,096	0,0003	-0,041
C104	555,41	10	**555,27**	**10**	604,409	0,0003	-0,088
C105	556,18	10	**556,03**	**10**	557,248	0,0003	-0,002
C106	556,18	10	**556,03**	**10**	563,753	0,0003	-0,014
C107	556,18	10	**556,03**	**10**	560,600	0,0003	-0,008
C108	555,80	10	**555,65**	**10**	566,295	0,0003	-0,019
C109	555,80	10	**555,65**	**10**	574,455	0,0003	-0,034

Tabela 5.11: Resultados para o conjunto de instâncias RC1 de Solomon

	Repoussis (2009)		**GRASP+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC101	1227,37	**14**	**1131,64**	**14**	1151,544	0,078	0,062
RC102	1203,05	**12**	**1168,31**	**12**	1118,133	0,030	0,071
RC103	923,15	**11**	**911,60**	**11**	952,058	0,013	-0,031
RC104	787,02	10	794,40	10	836,686	-0,010	-0,063
RC105	1195,20	**13**	**1044,40**	**13**	1053,389	0,130	0,119
RC106	1095,65	**11**	**993,29**	**11**	990,503	0,093	0,096
RC107	861,28	**11**	**846,53**	**11**	872,061	0,017	-0,013
RC108	831,09	**10**	**824,12**	**10**	838,376	0,008	-0,009

Tabela 5.12: Resultados para o conjunto de instâncias R2 de Solomon

	Repoussis (2009)		**GRASP+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R201	1182,43	4	1198,06	4	1238,979	-0,013	-0,048
R202	1149,59	3	1179,32	3	1108,580	-0,026	0,036
R203	889,12	3	923,51	3	956,334	-0,039	-0,076
R204	801,46	2	760,20	3	1055,781	0,052	-0,320
R205	943,33	3	1008,34	3	1055,781	-0,069	-0,119
R206	869,27	3	894,39	3	932,758	-0,029	-0,073
R207	857,08	2	790,98	3	819,687	0,077	0,044
R208	700,53	2	703,40	2	740,440	-0,004	-0,057
R209	851,69	3	876,52	3	926,996	-0,030	-0,088
R210	892,45	3	901,07	3	963,509	-0,010	-0,080
R211	886,90	2	761,22	3	805,376	0,142	0,092

Tabela 5.13: Resultados para o conjunto de instâncias C2 de Solomon

	Repoussis (2009)		**GRASP+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C201	548,51	3	**548,30**	**3**	548,300	0,0004	0,0004
C202	548,51	3	**548,30**	**3**	552,199	0,0004	-0,007
C203	548,13	3	**630,60**	**3**	737,442	-0,151	-0,345
C204	547,55	3	618,54	3	697,1397	-0,130	-0,273
C205	545,83	3	**545,61**	**3**	545,610	0,0004	0,0004
C206	545,45	3	**545,22**	**3**	550,973	0,0004	-0,010
C207	545,24	3	**545,01**	**3**	557,519	0,0004	-0,023
C208	545,28	3	**545,05**	**3**	545,050	0,0004	0,0004

Tabela 5.14: Resultados para o conjunto de instâncias RC2 de Solomon

	Repoussis (2009)		**GRASP+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC201	1303,73	4	1349,95	4	1384,184	-0,036	-0,062
RC202	1321,43	3	1349,95	3	1161,308	-0,023	0,121
RC203	993,29	3	1019,77	3	1075,073	-0,027	-0,082
RC204	718,97	3	785,72	3	835,305	-0,09284	-0,1618
RC205	1189,84	4	**992,60**	**3**	1048,727	0,166	0,119
RC206	1091,79	3	1098,27	3	1167,916	-0,006	-0,070
RC207	998,70	3	1018,68	3	1097,985	-0,020	-0,010
RC208	769,40	3	838,34	3	890,483	-0,090	-0,160

5.5 Resultados para o HSCM+ILS

O algoritmo HSCM+ILS foi capaz de encontrar novas soluções para 58,33% das instâncias do conjunto R1. No caso do conjunto R2, não foram encontradas soluções melhores do que as apresentadas na literatura, tendo, no entanto, resultados próximos aos já apresentados na literatura. Para os conjuntos de instâncias C1 e C2, o HSCM+ILS foi capaz de encontrar novas soluções em 100% e 87,5% das instâncias testadas, respectivamente. Ao ser aplicado sobre os problemas-teste das instâncias RC1 e RC2, o HSCM+ILS apresentou novas soluções em 62,5% e 12,5% das instâncias testadas, respectivamente.

As Tabelas 5.15 a 5.20 apresentam os resultados completos obtidos pelo HSCM+ILS ao ser aplicado sobre as instâncias de Solomon.

Tabela 5.15: Resultados para o conjunto de instâncias R1 de Solomon

	Repoussis (2009)		**HSCM+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R101	1192,85	19	**1195,13**	**18**	1194,741	-0,002	-0,002
R102	1079,39	17	**1046,79**	**17**	1052,254	0,030	0,025
R103	1016,78	13	**982,42**	**13**	904,722	0,034	0,110
R104	869,63	9	**809,11**	**9**	815,149	0,069	0,063
R105	1055,04	14	**1090,62**	**13**	1053,704	-0,034	0,001
R106	1000,95	12	**979,64**	**12**	1003,990	0,021	-0,003
R107	912,99	10	946,06	10	928,160	-0,036	-0,017
R108	760,3	9	780,31	9	768,484	-0,026	-0,011
R109	934,53	11	912,18	12	934,768	0,024	-0,0003
R110	846,49	10	898,12	10	872,952	-0,061	-0,031
R111	895,21	10	**872,96**	**10**	909,150	0,025	-0,016
R112	811,73	9	826,96	9	775,224	-0,019	0,0450

Tabela 5.16: Resultados para o conjunto de instâncias C1 de Solomon

	Repoussis (2009)		**HSCM+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C101	556,18	10	**556,03**	**10**	556,030	0,0003	0,0003
C102	556,18	10	**556,03**	**10**	556,613	0,0003	-0,001
C103	556,18	10	**556,03**	**10**	556,481	0,0003	-0,001
C104	555,41	10	**555,27**	**10**	577,498	0,0003	-0,040
C105	556,18	10	**556,03**	**10**	556,030	0,0003	0,0003
C106	556,18	10	**556,03**	**10**	557,266	0,0003	-0,002
C107	556,18	10	**556,03**	**10**	556,030	0,0003	0,0003
C108	555,8	10	**555,65**	**10**	555,650	0,0003	0,0003
C109	555,8	10	**555,65**	**10**	566,791	0,0003	-0,020

Tabela 5.17: Resultados para o conjunto de instâncias RC1 de Solomon

	Repoussis (2009)		**HSCM+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC101	1227,37	14	**1130,06**	**14**	1148,189	0,079	0,065
RC102	1203,05	12	**1071,65**	**12**	1070,614	0,109222	0,110
RC103	923,15	11	931,97	11	980,392	-0,010	-0,0620
RC104	787,02	10	806,19	10	856,267	-0,024	-0,088
RC105	1195,2	13	**1047,62**	**13**	1080,260	0,123	0,096
RC106	1095,65	11	**997,22**	**11**	1005,687	0,090	0,082
RC107	861,28	11	**841,77**	**11**	883,550	0,023	-0,026
RC108	831,09	10	841,27	10	851,1777	-0,012	-0,024

Tabela 5.18: Resultados para o conjunto de instâncias R2 de Solomon

	Repoussis (2009)		**HSCM+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R201	1182,43	4	1188,48	4	1208,857	-0,005	-0,022
R202	1149,59	3	1035,47	4	1070,826	0,099	0,069
R203	889,12	3	913,72	3	947,1173	-0,027	-0,065
R204	801,46	2	731,48	3	747,538	0,087	0,067
R205	943,33	3	1000,41	3	1062,727	-0,061	-0,127
R206	869,27	3	881,46	3	935,114	-0,014	-0,076
R207	857,08	2	803,33	3	839,858	0,063	0,020
R208	700,53	2	714,89	2	726,826	-0,021	-0,038
R209	851,69	3	874,78	3	915,714	-0,027	-0,075
R210	892,45	3	917,74	3	938,078	-0,028	-0,051
R211	886,9	2	773,91	3	799,154	0,127	0,099

Tabela 5.19: Resultados para o conjunto de instâncias C2 de Solomon

	Repoussis (2009)		HSCM+ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C201	548,51	3	**548,3**	3	548,300	0,0004	0,0004
C202	548,51	3	**548,3**	3	548,300	0,0004	0,0004
C203	548,13	3	**546,99**	3	548,564	0,002	-0,001
C204	547,55	3	549,26	3	553,914	-0,003	-0,012
C205	545,83	3	**545,61**	3	545,610	0,0004	0,0004
C206	545,45	3	**545,22**	3	545,220	0,0004	0,0004
C207	545,24	3	**545,01**	3	545,010	0,0004	0,0004
C208	545,28	3	**545,05**	3	545,050	0,0004	0,0004

Tabela 5.20: Resultados para o conjunto de instâncias RC2 de Solomon

	Repoussis (2009)		HSCM+ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC201	1303,73	4	1343,77	4	1148,189	-0,031	0,119
RC202	1321,43	3	1095,83	4	1070,614	0,171	0,190
RC203	993,29	3	1039,15	3	1081,340	-0,046	-0,089
RC204	718,97	3	816,05	3	863,985	-0,135	-0,202
RC205	1189,84	4	**974,4**	**3**	1065,376	0,181	0,105
RC206	1091,79	3	1140,86	3	1174,450	-0,050	-0,076
RC207	998,7	3	1049,26	3	1107,443	-0,051	-0,109
RC208	769,4	3	845,48	3	875,379	-0,010	-0,138

5.6 Resultados para o *MAX-MIN Ant System*+ILS

Nesta abordagem, o algoritmo MMAS+ILS demonstrou ser competitivo, sendo capaz de encontrar soluções melhores que as existentes na literatura para mais de $55,35\%$ das instâncias. Ao ser aplicado sobre os conjuntos de instâncias R1 e R2, o algoritmo MMAS+ILS encontrou novas soluções em mais de $58,33\%$ das instâncias do conjunto R1. Já em relação ao conjunto R2, o algoritmo MMAS+ILS não foi capaz de encontrar soluções melhores que as presentes na literatura, apesar dos resultados para alguns dos testes terem sido próximos aos melhores existentes. No caso dos conjuntos C1 e C2, o algoritmo encontrou soluções melhores do que as da literatura para mais de 100% e $87,5\%$ dos casos, respectivamente. Em relação aos conjuntos RC1 e RC2, o algoritmo MMAS+ILS encontrou novas soluções em mais de 75% e 25% dos casos. Os resultados alcançados pelo algoritmo MMAS+ILS ao ser aplicado sobre as instâncias de Solomon são apresentados pelas Tabelas 5.21 a 5.26.

Tabela 5.21: Resultados para o conjunto de instâncias R1 de Solomon

	Repoussis (2009)		**MMAS+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R101	1192,85	19	**1195,13**	**18**	1193,12	0,00	0,00
R102	1079,39	17	**1046,46**	**17**	1051,21	0,03	0,03
R103	1016,78	13	**981,43**	**13**	987,78	0,03	0,03
R104	869,63	9	**804,86**	**9**	805,79	0,07	0,07
R105	1055,04	14	**1091,63**	**13**	1068,00	-0,03	-0,01
R106	1000,95	12	**982,17**	**12**	1006,53	0,02	-0,01
R107	912,99	10	**899,05**	**10**	914,91	0,02	0,00
R108	760,3	9	776,87	9	767,34	-0,02	-0,01
R109	934,53	11	958,27	11	934,36	-0,03	0,00
R110	846,49	10	889,86	10	866,48	-0,05	-0,02
R111	895,21	10	901,19	10	895,10	-0,01	0,00
R112	811,73	9	832,88	9	787,49	-0,03	0,03

Tabela 5.22: Resultados para o conjunto de instâncias C1 de Solomon

	Repoussis (2009)		**MMAS+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C101	556,18	10	**556,03**	**10**	581,34	0,00	-0,05
C102	556,18	10	**556,03**	**10**	586,19	0,00	-0,05
C103	556,18	10	**556,03**	**10**	602,48	0,00	-0,08
C104	555,41	10	**555,27**	**10**	631,03	0,00	-0,14
C105	556,18	10	**556,03**	**10**	560,31	0,00	-0,01
C106	556,18	10	**556,03**	**10**	562,41	0,00	-0,01
C107	556,18	10	**556,03**	**10**	620,72	0,00	-0,12
C108	555,8	10	**555,65**	**10**	558,97	0,00	-0,01
C109	555,8	10	**555,65**	**10**	577,38	0,00	-0,04

Tabela 5.23: Resultados para o conjunto de instâncias RC1 de Solomon

	Repoussis (2009)		**MMAS+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC101	1227,37	14	**1130,36**	**14**	1164,74	0,08	0,05
RC102	1203,05	12	**1149,76**	**12**	1056,78	0,04	0,12
RC103	923,15	11	**906,73**	**11**	954,25	0,02	-0,03
RC104	787,02	10	799,56	10	830,67	-0,02	-0,06
RC105	1195,2	13	**1038,78**	**13**	1055,68	0,13	0,12
RC106	1095,65	11	**997,35**	**11**	989,09	0,09	0,10
RC107	861,28	11	**842,09**	**11**	885,07	0,02	-0,03
RC108	831,09	10	834,83	10	835,53	0,00	-0,01

Tabela 5.24: Resultados para o conjunto de instâncias R2 de Solomon

	Repoussis (2009)		MMAS+ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R201	1182,43	4	1188,95	4	1249,216	-0,006	-0,057
R202	1149,59	3	1216,8	3	1086,9	-0,058	0,055
R203	889,12	3	919,46	3	991,087	-0,034	-0,115
R204	801,46	2	747,75	3	799,778	0,067	0,002
R205	943,33	3	1002,13	3	1056,531	-0,062	-0,120
R206	869,27	3	905,18	3	965,142	-0,041	-0,110
R207	857,08	2	809,73	3	853,525	0,055	0,004
R208	700,53	2	709,27	2	757,382	-0,012	-0,081
R209	851,69	3	875,27	3	934,326	-0,028	-0,097
R210	892,45	3	920,37	3	986,549	-0,031	-0,105
R211	886,9	2	782,78	3	823,564	0,117	0,071

Tabela 5.25: Resultados para o conjunto de instâncias C2 de Solomon

	Repoussis (2009)		MMAS+ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C201	548,51	3	**548,3**	**3**	638,163	0,0004	-0,164
C202	548,51	3	**548,3**	**3**	704,690	0,0004	-0,284
C203	548,13	3	**546,99**	**3**	695,691	0,002	-0,269
C204	547,55	3	553,67	3	743,106	-0,011	-0,357
C205	545,83	3	**545,61**	**3**	673,598	0,0004	-0,234
C206	545,45	3	**545,22**	**3**	636,965	0,0004	-0,167
C207	545,24	3	**545,01**	**3**	674,920	0,0004	-0,237
C208	545,28	3	**545,05**	**3**	674,920	0,0004	-0,238

Tabela 5.26: Resultados para o conjunto de instâncias RC2 de Solomon

	Repoussis (2009)		MMAS+ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC201	1303,73	4	1323,07	4	1406,193	-0,015	-0,079
RC202	1321,43	3	1370,85	3	1203,867	-0,037	0,089
RC203	993,29	3	999,87	3	1105,628	-0,007	-0,113
RC204	718,97	3	781,14	3	841,062	-0,086	-0,169
RC205	1189,84	4	**980,01**	**3**	1051,388	0,176	0,116
RC206	1091,79	3	**1090,43**	**3**	1194,957	0,001	-0,094
RC207	998,7	3	1016,33	3	1089,903	-0,018	-0,091
RC208	769,4	3	818,78	3	883,278	-0,064	-0,148

5.6.1 Resultados para as Instâncias com 200 Clientes

Ao ser aplicado sobre os conjuntos de instâncias R1, C1 e RC1 propostas por Homberger e Gehring (1999) que possuem 200 clientes, o algoritmo MMAS+ILS foi capaz de encontrar soluções melhores que as presentes na literatura em 100% das instâncias de todos os conjuntos testados. Os resultados obtidos pelo MMAS+ILS são mostrados nas Tabelas 5.27, 5.28 e 5.29.

Tabela 5.27: Resultados para as instâncias do conjunto R1 com 200 clientes

	Repoussis (2006)		**MMAS+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R121	5799,10	20	**4233,8**	**19**	4115,633	0,270	0,980
R122	4820,93	18	**3596,59**	**18**	3796,837	0,254	0,970
R123	4308,11	18	**3008,23**	**18**	3228,739	0,302	0,975
R124	3433,66	18	**2679,45**	**18**	2911,413	0,220	0,966
R125	4936,86	18	**3633,56**	**18**	3850,191	0,264	0,981
R126	4801,26	18	**3301,48**	**18**	3486,327	0,312	0,973
R127	4376,27	18	**2778,88**	**18**	3031,850	0,365	0,972
R128	3516,08	18	**2629,35**	**18**	2802,799	0,252	0,967
R129	4353,28	18	**3359,54**	**18**	3534,648	0,228	0,976
R1210	4017,23	18	**2889,73**	**18**	3113,269	0,281	0,966

Tabela 5.28: Resultados para as instâncias do conjunto C1 com 200 clientes

	Repoussis (2006)		**MMAS+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C121	2044,80	20	**2491,93**	**18**	2655,859	-0,218	-0,299
C122	2989,41	19	**2197,71**	**18**	2572,245	0,264	0,140
C123	2894,41	19	**2001,79**	**18**	2425,124	0,308	0,162
C124	2437,72	18	**1963,6**	**18**	2254,671	0,194	0,075
C125	2186,22	20	**2242,58**	**18**	2601,037	-0,026	-0,190
C126	2728,33	22	**2240,81**	**18**	2459,333	0,179	0,099
C127	2212,82	20	**2256,22**	**18**	2586,367	-0,020	-0,169
C128	2649,75	20	**2033,65**	**18**	2365,497	0,233	0,107
C129	2773,60	19	**2023,97**	**18**	2278,357	0,270	0,178
C1210	2577,78	19	**2022,45**	**18**	2243,921	0,215	0,130

5.6.2 Resultados para as Instâncias com 400 Clientes

O algoritmo MMAS+ILS também foi testado utilizando as instâncias pertencentes aos conjuntos R1, C1 e RC1 propostas por Homberger e Gehring (1999) que possuem 400 clientes. Em relação a estas instâncias, o algoritmo MMAS+ILS foi capaz de encontrar soluções melhores que as apresentadas na literatura em 100% dos casos. Os dados relativos aos resultados obtidos pelo algoritmo MMAS+ILS podem ser vistos por intermédio das Tabelas 5.30, 5.31 e 5.32.

Tabela 5.29: Resultados para as instâncias do conjunto RC1 com 200 clientes

	Repoussis (2006)		MMAS+ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC121	3987,91	19	**2623,42**	**19**	2796,566	0,342	0,299
RC122	4092,80	18	**2882,13**	**18**	3092,627	0,296	0,244
RC123	3631,56	18	**2816,98**	**18**	3040,057	0,224	0,163
RC124	3237,43	18	**2533,81**	**18**	2866,736	0,217	0,115
RC125	3799,42	18	**2883,65**	**18**	2729,255	0,241	0,282
RC126	3824,80	18	**2806,68**	**18**	2756,503	0,266	0,279
RC127	3726,23	18	**2541,34**	**18**	2833,171	0,318	0,240
RC128	3480,78	18	**2567,69**	**18**	2728,686	0,262	0,216
RC129	3394,62	18	**2493,8**	**18**	2692,781	0,265	0,207
RC1210	3278,08	18	**2432,8**	**18**	6591,113	0,258	-1,011

Tabela 5.30: Resultados para as instâncias do conjunto R1 com 400 clientes

	Repoussis (2006)		MMAS+ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R141	12718,18	40	**8848,44**	**37**	8895,786	0,304	0,981
R142	11816,22	36	**8082,15**	**36**	8224,813	0,316	0,967
R143	10119,41	36	**7442,21**	**36**	7922,680	0,265	0,968
R144	8316,25	36	**6933,06**	**36**	7387,429	0,166	0,958
R145	10646,18	36	**8137,49**	**36**	7963,298	0,236	0,973
R146	11401,47	36	**7620,47**	**36**	8266,420	0,332	0,972
R147	9190,60	36	**7176,62**	**36**	7697,906	0,219	0,973
R148	8173,47	36	**7099,12**	**36**	7459,158	0,132	0,968
R149	9926,96	36	**7747,35**	**36**	7925,979	0,220	0,962
R1410	9311,09	36	**7164,51**	**36**	7594,684	0,231	0,980

Tabela 5.31: Resultados para as instâncias do conjunto C1 com 400 clientes

	Repoussis (2006)		MMAS+ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C141	4941,35	40	**5614,10**	**37**	6324,120	-0,136	-0,280
C142	10347,64	38	**7196,51**	**36**	7997,137	0,305	0,227
C143	7500,22	37	**6341,56**	**36**	7142,774	0,155	0,048
C144	5927,36	37	**5510,42**	**36**	6123,614	0,0703	-0,033
C145	5578,89	40	**6161,20**	**36**	6187,662	-0,104	-0,109
C146	6671,50	41	**6608,04**	**36**	5894,871	0,010	0,116
C147	5596,08	40	**5011,14**	**37**	5894,871	0,105	-0,053
C148	6786,02	39	**6057,38**	**36**	6698,987	0,107	0,013
C149	7406,01	37	**5644,95**	**36**	6297,471	0,234	0,150
C1410	-	-	591280	36	6540,274	-	-

Tabela 5.32: Resultados para as instâncias do conjunto RC1 com 400 clientes

	Repoussis (2006)		MMAS+ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC141	8944,75	37	**6471,32**	**37**	6582,802	0,276	0,264
RC142	10135,74	36	**6550,28**	**36**	6518,602	0,354	0,357
RC143	9109,80	36	**6609,30**	**36**	7154,930	0,274	0,215
RC144	7323,19	36	**6388,37**	**36**	7103,379	0,128	0,030
RC145	8872,65	37	**6130,81**	**37**	6427,967	0,309	0,276
RC146	8623,46	37	**6066,44**	**37**	6318,209	0,297	0,267
RC147	9163,29	36	**6445,50**	**36**	6303,691	0,297	0,312
RC148	8575,29	36	**6317,54**	**36**	6236,029	0,263	0,273
RC149	8440,29	36	**6212,28**	**36**	6130,677	0,264	0,274
RC1410	8164,35	36	**5905,30**	**36**	6178,479	0,277	0,243

5.6.3 Resultados para as Instâncias com 600 Clientes

O algoritmo MMAS+ILS foi aplicado sobre os conjuntos de instâncias R1, C1 e RC1 propostos por Homberger e Gehring (1999) que possuem 600 clientes. Dentre as instâncias pertencentes aos conjuntos R1 e RC1 que foram utilizadas para os testes, o algoritmo MMAS+ILS conseguiu encontrar novas soluções em 70% dos casos. O algoritmo MMAS+ILS encontrou novas soluções em 50% e 90% das instâncias dos conjuntos R1 e RC1, respectivamente. Apesar de também de sido aplicado sobre as instâncias do conjunto C1, não foi possível comparar os resultados obtidos pelo algoritmo desenvolvido com outro método, devido a ausência de dados na literatura para tal conjunto de instâncias. As Tabelas 5.33, 5.34 e 5.35 apresentam os resultados obtidos pelo MMAS+ILS para tais conjuntos de instâncias.

Tabela 5.33: Resultados para as instâncias do conjunto R1 com 600 clientes

	Repoussis (2006)		MMAS+ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R161	27407,71	60	**17897,00**	**55**	18189,350	0,347	0,336
R162	28147,30	54	16140,40	55	17115,930	0,427	0,392
R163	22750,62	54	14766,70	55	15524,280	0,351	0,318
R164	17249,73	54	**15204,60**	**54**	1593,813	0,119	0,908
R165	20699,50	55	**16505,40**	**55**	414,666	0,203	0,980
R166	25450,71	54	15263,20	55	470,394	0,400	0,983
R167	22901,71	54	**18219,00**	**54**	1231,304	0,205	0,946
R168	15459,61	54	**14682,60**	**54**	1596,830	0,050	0,897
R169	21458,64	54	15963,60	55	367,637	0,256	0,983
R1610	19606,41	54	14998,30	55	463,094	0,235	0,976

Tabela 5.34: Resultados para as instâncias do conjunto C1 com 600 clientes

	Repoussis (2006)		MMAS+ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C161	-	-	10685,3	56	11045,21	-	-
C162	-	-	11436,4	56	12738,57	-	-
C163	-	-	10246,1	56	11442,35	-	-
C164	-	-	9538,32	56	10147,09	-	-
C165	-	-	9979,84	56	10717,31	-	-
C166	-	-	10087,2	56	11367,35	-	-
C167	-	-	9904,51	56	10668,08	-	-
C168	-	-	10362,3	56	11095,08	-	-
C169	-	-	9637,38	56	11095,08	-	-
C1610	-	-	9922,41	56	10754,09	-	-

Tabela 5.35: Resultados para as instâncias do conjunto RC1 com 600 clientes

	Repoussis (2006)		MMAS+ILS				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC161	20315,74	55	12915,80	56	13337,570	0,364	0,343
RC162	19476,73	55	**13702,20**	**55**	14307,360	0,296	0,265
RC163	18304,66	55	**14049,20**	**55**	15411,990	0,232	0,158
RC164	15904,62	55	**14161,70**	**55**	15561,170	0,110	0,0216
RC165	19547,47	55	**14039,10**	**55**	12085,350	0,282	0,382
RC166	18489,34	55	**13957,40**	**55**	13371,140	0,245	0,277
RC167	18278,75	55	**13709,00**	**55**	13197,760	0,250	0,278
RC168	18147,58	55	**12679,10**	**55**	13238,800	0,301	0,271
RC169	16606,75	55	**12878,80**	**55**	13099,650	0,225	0,211
RC1610	16288,81	55	**12807,20**	**55**	13385,130	0,229	0,194

5.7 Resultados para o P-ACO

As subseções seguintes apresentam os resultados referentes à aplicação do algoritmo P-ACO às instâncias de Solomon. São apresentados resultados referentes às estratégias *Age-based Strategy*, *Quality-based Strategy* e *Elitist-based Strategy*.

5.7.1 Resultados para o P-ACO *Age-based Strategy*+ILS

Ao ser testado utilizando as instâncias propostas por Solomon (1987), o algoritmo P-ACO*Age-based Strategy* + ILS (P-ACO$_{\text{Age}}$ + ILS) apresentou bons resultados. O algoritmo foi capaz de encontrar soluções melhores do que as presentes na literatura em 53,6% das instâncias.

O P-ACO$_{\text{Age}}$ + ILS encontrou novas soluções, em relação às apresentadas na literatura, em 50% das instâncias do conjunto R1, porém, apesar de encontrados soluções com valores próximos em muitos casos, tal algoritmo não foi capaz de alcançar os resultados referentes ao conjunto R2.

O P-ACO$_{\text{Age}}$ + ILS foi capaz de encontrar soluções melhores do que as presentes na literatura em 100% e 87,5% das instâncias pertencentes aos conjuntos C1 e C2, respectivamente. No caso dos conjuntos RC1 e RC2, o P-ACO$_{\text{Age}}$ + ILS conseguiu encontrar soluções melhores do que as existentes na literatura em 87,5% e 12,5% das instâncias pertencentes a tais conjuntos respectivamente.

As tabelas 5.36 a 5.41 apresentam os resultados obtidos pelo P-ACO$_{\text{Age}}$ + ILS.

Tabela 5.36: Resultados para o conjunto de instâncias R1 de Solomon

	Repoussis (2009)		$P-ACO_{Age}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R101	1192,85	19	**1195,13**	**18**	1194,818	-0,0019	-0,002
R102	1079,39	17	**1046,37**	**17**	1053,744	0,031	0,024
R103	1016,78	13	**989,29**	**13**	997,291	0,027	0,0192
R104	869,63	9	**802,16**	**9**	802,568	0,078	0,077
R105	1055,04	14	**1099,22**	**13**	1076,022	-0,042	-0,020
R106	1000,95	12	**997,81**	**12**	1023,282	0,003	-0,022
R107	912,99	10	915,08	10	915,384	-0,002	-0,003
R108	760,3	9	772,27	9	763,403	-0,016	-0,004
R109	934,53	11	950,62	11	930,447	-0,017	0,004
R110	846,49	10	813,15	11	855,140	0,040	-0,010
R111	895,21	10	921,8	10	883,577	-0,030	0,0130
R112	811,73	9	907,8	9	783,746	-0,118	0,035

Tabela 5.37: Resultados para o conjunto de instâncias C1 de Solomon

	Repoussis (2009)		$P-ACO_{Age}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C101	556,18	10	**556,03**	**10**	570,728	0,0003	-0,026
C102	556,18	10	**556,03**	**10**	571,678	0,0003	-0,028
C103	556,18	10	**556,03**	**10**	611,377	0,0003	-0,099
C104	555,41	10	**555,27**	**10**	631,632	0,0003	-0,137
C105	556,18	10	**556,03**	**10**	581,010	0,0003	-0,045
C106	556,18	10	**556,03**	**10**	565,682	0,0003	-0,017
C107	556,18	10	**556,03**	**10**	585,409	0,0003	-0,053
C108	555,80	10	**555,65**	**10**	559,089	0,0003	-0,006
C109	555,80	10	**555,65**	**10**	582,919	0,0003	-0,049

Tabela 5.38: Resultados para o conjunto de instâncias RC1 de Solomon

	Repoussis (2009)		$P-ACO_{Age}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC101	1227,37	14	**1130,06**	**14**	1155,821	0,079	0,058
RC102	1203,05	12	**1120,45**	**12**	1071,102	0,069	0,110
RC103	923,15	11	**919,44**	**11**	949,693	0,004	-0,029
RC104	787,02	10	800,25	10	845,4377	-0,017	-0,074
RC105	1195,20	13	**1045,77**	**13**	1053,565	0,125	0,119
RC106	1095,65	11	**993,29**	**11**	1003,386	0,093	0,084
RC107	861,28	11	**844,95**	**11**	900,270	0,019	-0,045
RC108	831,09	10	**814,73**	**10**	842,565	0,020	-0,014

Tabela 5.39: Resultados para o conjunto de instâncias R2 de Solomon

	Repoussis (2009)		$P-ACO_{Age}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R201	1182,43	4	1198,85	4	1241,309	-0,014	-0,050
R202	1149,59	3	1231,33	3	1084,744	-0,071	0,056
R203	889,12	3	906,56	3	978,687	-0,020	-0,101
R204	801,46	2	736,75	3	785,458	0,081	0,020
R205	943,33	3	1001,74	3	1056,446	-0,062	-0,120
R206	869,27	3	889,27	3	944,502	-0,023	-0,087
R207	857,08	2	796,70	3	846,088	0,071	0,013
R208	700,53	2	720,04	2	761,923	-0,030	-0,088
R209	851,69	3	882,21	3	929,598	-0,035	-0,092
R210	892,45	3	922,18	3	986,642	-0,033	-0,110
R211	886,90	2	777,31	3	819,928	0,124	0,076

Tabela 5.40: Resultados para o conjunto de instâncias C2 de Solomon

	Repoussis (2009)		$P-ACO_{Age}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C201	548,51	3	**548,30**	**3**	551,317	0,0004	-0,005
C202	548,51	3	**548,30**	**3**	668,450	0,0004	-0,219
C203	548,13	3	**546,99**	**3**	725,720	0,002	-0,324
C204	547,55	3	576,91	3	756,053	-0,054	-0,381
C205	545,83	3	**545,61**	**3**	573,432	0,0004	-0,051
C206	545,45	3	**545,22**	**3**	608,482	0,0004	-0,116
C207	545,24	3	**545,01**	**3**	561,125	0,0004	-0,029
C208	545,28	3	**545,05**	**3**	608,106	0,0004	-0,115

Tabela 5.41: Resultados para o conjunto de instâncias RC2 de Solomon

	Repoussis (2009)		$P-ACO_{Age}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC201	1303,73	4	1317,57	4	1394,187	-0,011	-0,069
RC202	1321,43	3	1118,51	3	1175,711	0,154	0,110
RC203	993,29	3	1039,54	3	1103,424	-0,047	-0,111
RC204	718,97	3	767,19	3	829,903	-105,707	-0,154
RC205	1189,84	4	**1009,37**	**3**	1057,262	0,153	0,111
RC206	1091,79	3	1094,12	3	1176,853	-0,002	-0,078
RC207	998,70	3	1025,89	3	1083,020	-0,027	-0,084
RC208	769,40	3	798,73	3	883,681	-0,038	-0,149

5.7.2 Resultados para o P-ACO *Quality-based Strategy*+ILS

O algoritmo P-ACO *Quality-based Strategy* + ILS (P-$ACO_{Quality}$ + ILS) demonstrou bons resultados, obtendo novas soluções em 51,8% das instâncias testadas. O P-$ACO_{Quality}$ + ILS conseguir encontrar soluções melhores do que as apresentadas na literatura em 58,33% das instâncias pertencentes ao conjunto R1, enquanto que tal algoritmo não conseguiu alcançar as soluções presentes na literatura que são referentes ao conjunto R2.

O P-$ACO_{Quality}$ + ILS foi capaz de encontrar soluções melhores do que as existentes na literatura em 88,89% e 75% para as instâncias que pertencem aos conjuntos C1 e C2, respectivamente.

O P-$ACO_{Quality}$ + ILS também encontrou soluções melhores do que as presentes na literatura em 87,5% e 12,5% das instâncias pertencentes aos conjuntos RC1 e RC2.

As tabelas 5.42a 5.47 apresentam os resultados obtidos pelo P-$ACO_{Quality}$ + ILS.

Tabela 5.42: Resultados para o conjunto de instâncias R1 de Solomon

	Repoussis (2009)		$P-ACO_{Quality}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R101	1192,85	19	**1196,49**	**18**	1206,369	-0,003	-0,011
R102	1079,39	17	**1046,37**	**17**	1053,556	0,031	0,024
R103	1016,78	13	**984,88**	**13**	997,049	0,031	0,019
R104	869,63	9	**802,75**	**9**	799,862	0,077	0,080
R105	1055,04	14	**1089,99**	**13**	1081,412	-0,033	-0,025
R106	1000,95	12	**995,21**	**12**	1021,925	0,006	-0,021
R107	912,99	10	923,20	10	926,451	-0,011	-0,015
R108	760,30	9	770,87	9	761,065	-0,014	-0,001
R109	934,53	11	938,34	11	932,116	-0,004	0,003
R110	846,49	10	917,17	10	866,803	-0,084	-0,024
R111	895,21	10	**890,35**	**10**	893,617	0,005	0,002
R112	811,73	9	743,45	10	779,838	0,084	0,039

Tabela 5.43: Resultados para o conjunto de instâncias C1 de Solomon

	Repoussis (2009)		$P-ACO_{Quality}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C101	556,18	10	**556,03**	**10**	580,784	0,0003	-0,044
C102	556,18	10	**556,03**	**10**	570,697	0,0003	-0,026
C103	556,18	10	**556,03**	**10**	596,775	0,0003	-0,073
C104	555,41	10	555,65	10	621,1907	-0,0004	-0,118
C105	556,18	10	**556,03**	**10**	575,628	0,0003	-0,035
C106	556,18	10	**556,03**	**10**	575,941	0,0003	-0,036
C107	556,18	10	**556,03**	**10**	565,055	0,0003	-0,016
C108	555,80	10	**555,65**	**10**	568,823	0,0003	-0,023
C109	555,80	10	**555,65**	**10**	569,841	0,0003	-0,025

Tabela 5.44: Resultados para o conjunto de instâncias RC1 de Solomon

	Repoussis (2009)		$P-ACO_{Quality}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC101	1227,37	14	**1130,06**	**14**	1151,183	0,079	0,062
RC102	1203,05	12	**1080,42**	**12**	1087,302	0,102	0,096
RC103	923,15	11	**911,21**	**11**	953,458	0,013	-0,033
RC104	787,02	10	809,33	10	848,1783	-0,028	-0,078
RC105	1195,20	13	**1044,40**	**13**	1054,752	0,126	0,118
RC106	1095,65	11	**996,50**	**11**	995,759	0,091	0,091
RC107	861,28	11	**844,95**	**11**	860,853	0,019	0,0005
RC108	831,09	10	**810,07**	**10**	836,277	0,025	-0,006

Tabela 5.45: Resultados para o conjunto de instâncias R2 de Solomon

	Repoussis (2009)		$P-ACO_{Quality}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R201	1182,43	4	1197,52	4	1235,680	-0,013	-0,045
R202	1149,59	3	1030,66	4	1088,828	0,104	0,053
R203	889,12	3	928,18	3	980,044	-0,044	-0,102
R204	801,46	2	749,86	3	786,658	0,064	0,019
R205	943,33	3	983,58	3	945,574	-0,043	-0,002
R206	869,27	3	896,30	3	945,574	-0,031	-0,088
R207	857,08	2	914,11	2	849,767	-0,067	0,009
R208	700,53	2	724,96	2	761,984	-0,035	-0,088
R209	851,69	3	882,91	3	938,002	-0,037	-0,101
R210	892,45	3	923,59	3	978,154	-0,035	-0,096
R211	886,90	2	786,23	3	817,511	0,114	0,078

Tabela 5.46: Resultados para o conjunto de instâncias C2 de Solomon

	Repoussis (2009)		$P-ACO_{Quality}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C201	548,51	3	**548,30**	**3**	548,300	0,0004	0,0004
C202	548,51	3	**548,30**	**3**	680,762	0,0004	-0,241
C203	548,13	3	618,83	3	732,977	-0,129	-0,337
C204	547,55	3	559,59	3	749,916	-0,022	-0,370
C205	545,83	3	**545,61**	**3**	545,610	0,0004	0,0004
C206	545,45	3	**545,22**	**3**	621,163	0,0004	-0,139
C207	545,24	3	**545,01**	**3**	561,525	0,0004	-0,030
C208	545,28	3	**545,05**	**3**	547,602	0,0004	-0,004

Tabela 5.47: Resultados para o conjunto de instâncias RC2 de Solomon

	Repoussis (2009)		$P-ACO_{Quality}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC201	1303,73	4	1321,47	4	1396,616	-0,014	-0,071
RC202	1321,43	3	1348,57	3	1181,011	-0,021	0,106
RC203	993,29	3	996,50	3	1108,635	-0,003	-0,116
RC204	718,97	3	753,80	3	830,495	-0,048	-0,155
RC205	1189,84	4	**994,25**	**3**	1045,116	0,164	0,122
RC206	1091,79	3	1106,35	3	1160,445	-0,013	-0,063
RC207	998,70	3	1031,25	3	1090,787	-0,033	-0,092
RC208	769,40	3	821,31	3	883,165	-0,068	-0,148

5.7.3 Resultados para o P-ACO *Elitist-based Strategy*+ILS

O algoritmo P-ACO*Elitist-based Strategy* + ILS (P-ACO$_{\text{Elitist}}$ + ILS)também demonstrou ser competitivo, conseguindo encontrar soluções melhores do que as presentes na literatura em 57,14% das instâncias testadas.

O P-ACO$_{\text{Elitist}}$ + ILS obteve soluções melhores do que as na literatura em 75% das instâncias do conjunto R1, mas, apesar de ter alcançado soluções com valores próximos tal algoritmo não foi capaz de encontrar novas soluções para as instâncias pertencentes ao conjunto R2.

O P-ACO$_{\text{Elitist}}$ + ILS encontrou soluções melhores do que as existentes na literatura em 100% e 75% das instâncias pertencentes aos conjuntos C1 e C2, respectivamente. No caso dos conjuntos RC1 e RC2 o algoritmo P-ACO$_{\text{Elitist}}$ + ILS foi capaz de encontrar novas soluções em 87,5% e 12,5% das instâncias das instâncias que pertencem a tais conjuntos, respectivamente.

As tabelas 5.48 a 5.53 apresentam os resultados obtidos pelo P-ACO$_{\text{Elitist}}$ + ILS.

Tabela 5.48: Resultados para o conjunto de instâncias R1 de Solomon

	Repoussis (2009)		$P-ACO_{Elitist}$**+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R101	1192,85	19	**1195,85**	**18**	1207,300	-0,003	-0,012
R102	1079,39	17	**1046,46**	**17**	1055,373	0,031	0,022
R103	1016,78	13	**982,97**	**13**	994,984	0,033	0,021
R104	869,63	9	**796,73**	**9**	804,457	0,084	0,075
R105	1055,04	14	**1098,53**	**13**	1060,158	-0,041	-0,005
R106	1000,95	12	**993,55**	**12**	1019,221	0,007	-0,018
R107	912,99	10	915	10	941,492	-0,002	-0,031
R108	760,30	9	785	9	770,479	-0,033	-0,013
R109	934,53	11	**931,72**	**11**	926,642	0,003	0,008
R110	846,49	10	885,16	10	866,842	-0,046	-0,024
R111	895,21	10	**880,33**	**10**	890,478	0,017	0,005
R112	811,73	9	**807,12**	**9**	785,639	0,006	0,032

Tabela 5.49: Resultados para o conjunto de instâncias C1 de Solomon

	Repoussis (2009)		$P-ACO_{Elitist}$**+ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C101	556,18	10	**556,03**	**10**	564,154	0,0003	-0,014
C102	556,18	10	**556,03**	**10**	562,801	0,0003	-0,012
C103	556,18	10	**556,03**	**10**	577,283	0,0003	-0,038
C104	555,41	10	**555,27**	**10**	617,529	0,0003	-0,112
C105	556,18	10	**556,03**	**10**	564,732	0,0003	-0,015
C106	556,18	10	**556,03**	**10**	565,085	0,0003	-0,016
C107	556,18	10	**556,03**	**10**	583,718	0,0003	-0,050
C108	555,80	10	**555,65**	**10**	564,970	0,0003	-0,017
C109	555,80	10	**555,65**	**10**	579,230	0,0003	-0,042

Tabela 5.50: Resultados para o conjunto de instâncias RC1 de Solomon

	Repoussis (2009)		$P-ACO_{Elitist}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC101	1227,37	14	**1130,06**	**14**	1156,261	0,079	0,058
RC102	1203,05	12	**1143,76**	**12**	1069,691	0,049	0,111
RC103	923,15	11	**910,76**	**11**	938,533	0,013	-0,017
RC104	787,02	10	797,95	10	858,890	-0,014	-0,091
RC105	1195,20	13	**1044,40**	**13**	1049,766	0,126	0,122
RC106	1095,65	11	**993,29**	**11**	982,422	0,093	0,103
RC107	861,28	11	**844,95**	**11**	886,543	0,019	-0,030
RC108	831,09	10	**807,24**	**10**	835,019	0,029	-0,005

Tabela 5.51: Resultados para o conjunto de instâncias R2 de Solomon

	Repoussis (2009)		$P-ACO_{Elitist}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
R201	1182,43	4	1204,75	4	1235,569	-0,019	-0,050
R202	1149,59	3	1047,18	4	1087,557	0,089	0,054
R203	889,12	3	921,39	3	988,625	-0,036	-0,112
R204	801,46	2	735,07	3	792,496	0,083	0,0112
R205	943,33	3	987,50	3	1053,181	-0,047	-0,117
R206	869,27	3	909,84	3	952,863	-0,047	-0,096
R207	857,08	2	805,24	3	852,093	0,061	0,006
R208	700,53	2	717,08	2	755,894	-0,024	-0,079
R209	851,69	3	866,45	3	933,906	-0,017	-0,097
R210	892,45	3	923,00	3	975,509	-0,034	-0,093
R211	886,90	2	792,46	3	823,761	0,107	0,071

Tabela 5.52: Resultados para o conjunto de instâncias C2 de Solomon

	Repoussis (2009)		$P-ACO_{Elitist}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
C201	548,51	3	**548,30**	**3**	548,30	0,0004	0,0004
C202	548,51	3	**548,30**	**3**	648,624	0,0004	-0,183
C203	548,13	3	555,39	3	727,934	-0,013	-0,328
C204	547,55	3	567,38	3	727,934	-0,036	-0,329
C205	545,83	3	**545,61**	**3**	614,130	0,0004	-0,125
C206	545,45	3	**545,22**	**3**	646,870	0,0004	-0,186
C207	545,24	3	**545,01**	**3**	631,825	0,0004	-0,159
C208	545,28	3	**545,05**	**3**	672,032	0,0004	-0,233

Tabela 5.53: Resultados para o conjunto de instâncias RC2 de Solomon

	Repoussis (2009)		$P-ACO_{Elitist}$+**ILS**				
Conjunto	**DT**	**NV**	**DT**	**NV**	**MDT**	**DMS**	**DRM**
RC201	1303,73	4	1311,02	4	1392,76	-0,006	-0,068
RC202	1321,43	3	1403,74	3	1191,107	-0,062	0,099
RC203	993,29	3	1036,79	3	1098,362	-0,044	-0,106
RC204	718,97	3	770,25	3	833,837	-0,071	-0,160
RC205	1189,84	4	**992,01**	**3**	1042,307	0,166	0,124
RC206	1091,79	3	1132,14	3	1194,075	-0,037	-0,094
RC207	998,70	3	1012,44	3	1082,060	-0,014	-0,084
RC208	769,40	3	824,02	3	880,008	-0,071	-0,144

Capítulo 6

Análise dos Experimentos Computacionais

Este Capítulo apresenta uma análise dos resultados computacionais encontrados, mostrados no Capítulo 5. A Seção 6.1 mostra uma análise acerca do desempenho computacional destes algoritmos. Já a Seção 6.3 mostra uma análise estatística destes resultados.

6.1 Desempenho Computacional

Com a finalidade de comparar o desempenho dos algoritmos que foram desenvolvidos para solucionar o Problema de Roteamento Aberto de Veículos com Janela de Tempo (PRAJT), foi utilizado o Teste de Probabilidade Empírica. De acordo com Aiex et al. (2002), o Teste de Probabilidade Empírica consiste em analisar a eficiência de algoritmos para alcançar uma determinada solução, em que são considerados o tempo gasto e o número de vezes que estes algoritmos foram capazes de chegar ao resultado estabelecido.

Para a realização dos testes, foram escolhidas 6 diferentes instâncias, cada uma delas pertencente a um dos subconjuntos de instâncias de Solomon. Foram escolhidas as instâncias R102, C104, RC104, R205, C204 e RC204, que pertencem aos subconjuntos R1, C1, RC1, R2, C2 e RC2, respectivamente. Estas instâncias foram escolhidas pois apresentaram maior dificuldade para alcançar bons resultados que as demais instâncias de seus respectivos subconjuntos, quando todos ou a maioria dos algoritmos desenvolvidos foram aplicados sobre elas. Deste modo, os maiores desvios em relação à melhor solução da literatura (DMS) foram obtidos quando os algoritmos foram testados sobre essas instâncias, que, por sua vez, foram, então, escolhidas para os testes.

Tabela 6.1: Resultados encontrados para as instâncias escolhidas para Testes de Probabilidade Empírica

Instância	NV	DT
R102	17	1046,29
C104	10	555,27
RC104	10	794,40
R205	2	914,11
C204	3	549,26
RC204	3	767,19

Durante a realização dos experimentos por meio dos algoritmos desenvolvidos, dentre os resultados obtidos por eles, de acordo com a Tabela 6.1, as melhores soluções alcançadas paras as instâncias R102, C104, RC104, R205, C204 e RC204 foram:

Os valores das soluções alvo foram definidos por meio da escolha da pior solução ótima dentre as obtidas pelos algoritmos aplicados sobre cada uma das instâncias escolhidas. O valor da distância total percorrida pelos veículos utilizados nas soluções tem um aumento de 10% para que os valores das soluções alvo possam ter chance de serem alcançados por todos os algoritmos. Todos os algoritmos utilizados foram executados cerca de 30 vezes para cada uma das instâncias testadas. As execuções com tempo repetido foram descartadas. A Tabela 6.2 apresenta os dados das soluções alvo definidas para a realização dos testes.

Tabela 6.2: Dados das soluções alvo.

Instância	NV	DT
R102	17	1052,25
C104	10	621,19
RC104	10	848,17
R205	3	1055,78
C204	3	697,03
RC204	3	863,98

A lista abaixo apresenta as legendas utilizadas nos gráficos relativos aos testes feitos.

i) MMAS: MMAS + ILS

ii) GRASP: GRASP + ILS

iii) ILS: ILS

iv) HSCM: HSCM + ILS

v) PACO$_A$: P-ACO$_{Age}$ + ILS

vi) PACO$_Q$: P-ACO$_{\text{Quality}}$ + ILS

vii) PACO$_E$: P-ACO$_{\text{Elitist}}$ + ILS

Os Testes de Probabilidade Empírica em relação a cada uma das seis instâncias estão apresentados a seguir, na ordem R102, C104, RC104, R205, C204, RC204.

Teste de Probabilidade Empírica para a instância R102

A Figura 6.1 mostra o resultado do Teste de Probabilidade Empírica para a instância R102.

Por meio da comparação entre os algoritmos aplicados sobre esta instância, percebe-se que o algoritmo ILS demonstrou ser o mais rápido dentre as diferentes configurações desenvolvidas, enquanto que os algoritmos P-ACO$_{\text{Age}}$ + ILS, P-ACO$_{\text{Quality}}$ + ILS e P-ACO$_{\text{Elitist}}$ + ILS foram os que demandaram maior tempo computacional para atingir o alvo. Além disso, por meio do gráfico, pode ser visto que os algoritmos P-ACO$_{\text{Age}}$ + ILS, P-ACO$_{\text{Quality}}$ + ILS e P-ACO$_{\text{Elitist}}$ + ILS tiveram desempenhos semelhantes. Deve ser observado que, neste caso, todos os algoritmos foram capazes de alcançar o alvo em 100% de suas respectivas execuções.

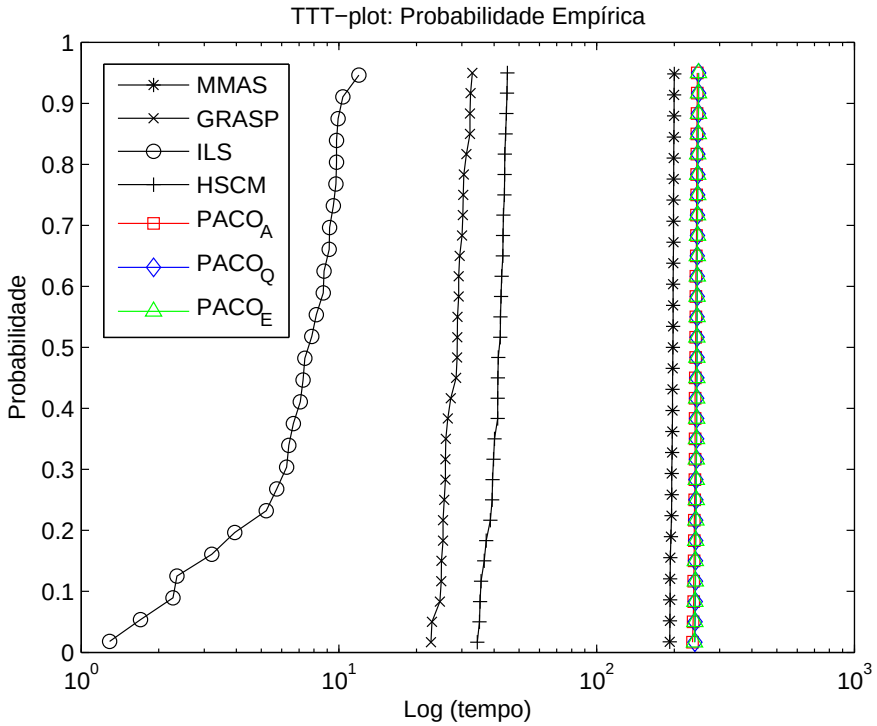


Figura 6.1: Teste de Probabilidade Empírica para a instância R102.

Teste de Probabilidade Empírica para a instância C104

A Figura 6.2 mostra o resultado do Teste de Probabilidade Empírica para a instância C104.

Conforme os testes feitos com a instância C104, o ILS foi o algoritmo que teve o menor computacional, alcançando o alvo em 80% das execuções. O HSCM+ILS foi o algoritmo mais eficaz, sendo capaz de alcançar o alvo em 100% das execuções. O P-ACO$_{\text{Elitist}}$ + ILS e o GRASP + ILS também demonstraram um desempenho eficiente, alcançado o alvo em 90% e 93% das execuções, respectivamente. Os algoritmos MMAS + ILS, P-ACO$_{\text{Age}}$ + ILS e P-ACO$_{\text{Quality}}$ + ILS alcançaram o alvo em 83%, 70% e 83% das execuções, respectivamente. Por intermédio do gráfico, é possível também verificar que o P-ACO$_{\text{Age}}$ + ILS, além de ter tido um dos maiores custos computacionais, foi o menos eficiente dentre os algoritmos para atingir o alvo.

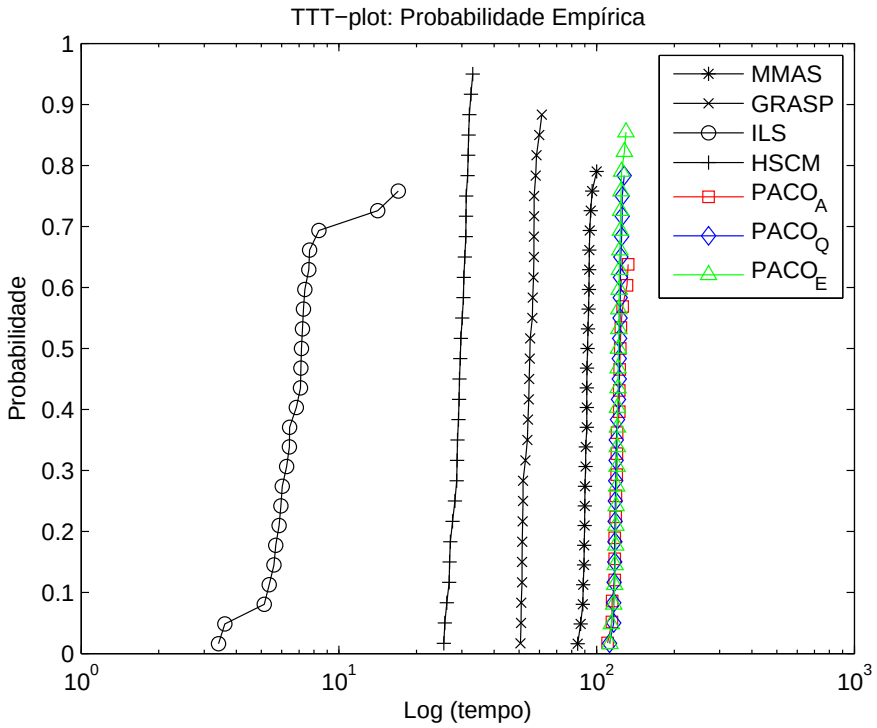


Figura 6.2: Time-to-target para a instância C104.

Teste de Probabilidade Empírica para a instância RC104

A Figura 6.3 mostra o resultado do Teste de Probabilidade Empírica para a instância RC104.

De acordo com os testes realizados usando a instância RC104, é possível verificar que o algoritmo ILS demonstrou ter uma convergência mais rápida do que os demais algoritmos, apesar de ter sido o menos eficaz, alcançado o alvo em apenas 43% das execuções. O algoritmo GRASP + ILS apresentou a maior eficiência, sendo capaz de alcançar o alvo em 73% das execuções. O o algoritmo HSCM + ILS foi capaz de alcançar o alvo em 66% das execuções. O o algoritmo MMAS + ILS alcançou o alvo em 63% das execuções. Os algoritmos P-ACO$_{\text{Age}}$ + ILS, PACO$_{\text{Quality}}$ + ILS e P-ACO$_{\text{Elitist}}$ + ILS alcançaram o alvo em 56%, 63% e 66% das execuções, respectivamente. Deve-se ressaltar que estes três algortimos tiveram custos de tempo computacional semelhante, maiores, no entanto, que os demais algoritmos. Ao se comparar o GRASP + ILS com o HSCM + ILS, percebe-se que, com uma probabilidade de alcançar o alvo próxima de 40%, o GRASP + ILS demonstrou ser mais rápido que o HSCM + ILS.

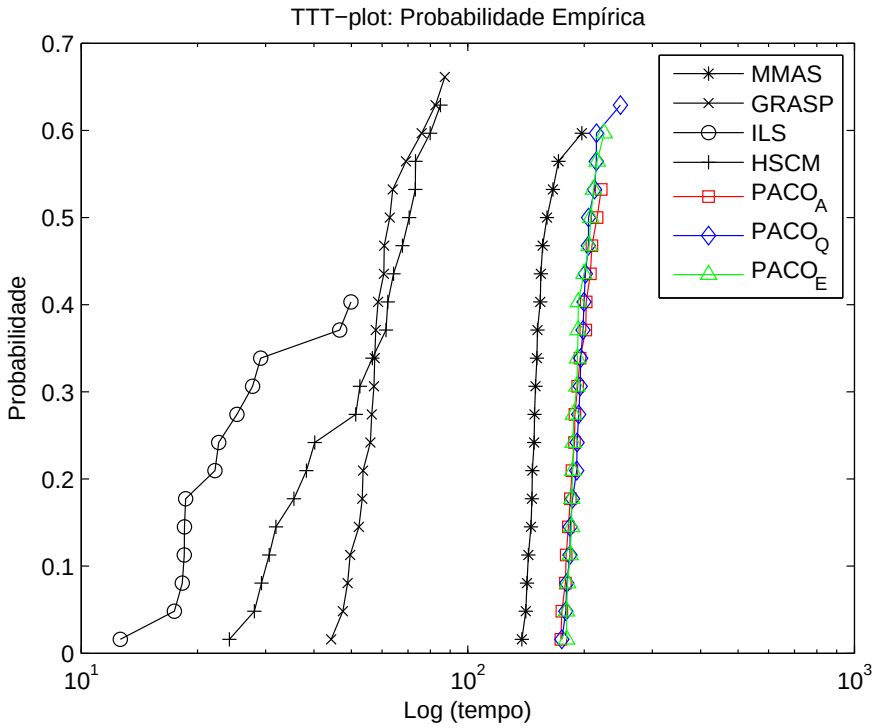


Figura 6.3: Time-to-target para a instância RC104.

Teste de Probabilidade Empírica para a instância R205

A Figura 6.3 mostra o resultado do Teste de Probabilidade Empírica para a instância R205.

Segundo os resultados dos testes obtidos usando a instância R205, pode ser visto que o algoritmo ILS, além de ter alcançado o alvo em 100% das execuções, teve o menor custo de tempo computacional. Os demais algoritmos testados também demonstraram ser eficazes para esta instância, sendo capazes de alcançar o alvo em 100% das execuções. Em relação ao tempo computacional, o HSCM + ILS teve o segundo menor custo; já os algoritmos P-ACO$_{\text{Age}}$ + ILS, P-ACO$_{\text{Quality}}$ + ILS e P-ACO$_{\text{Elitist}}$ + ILS tiveram os maiores custos.

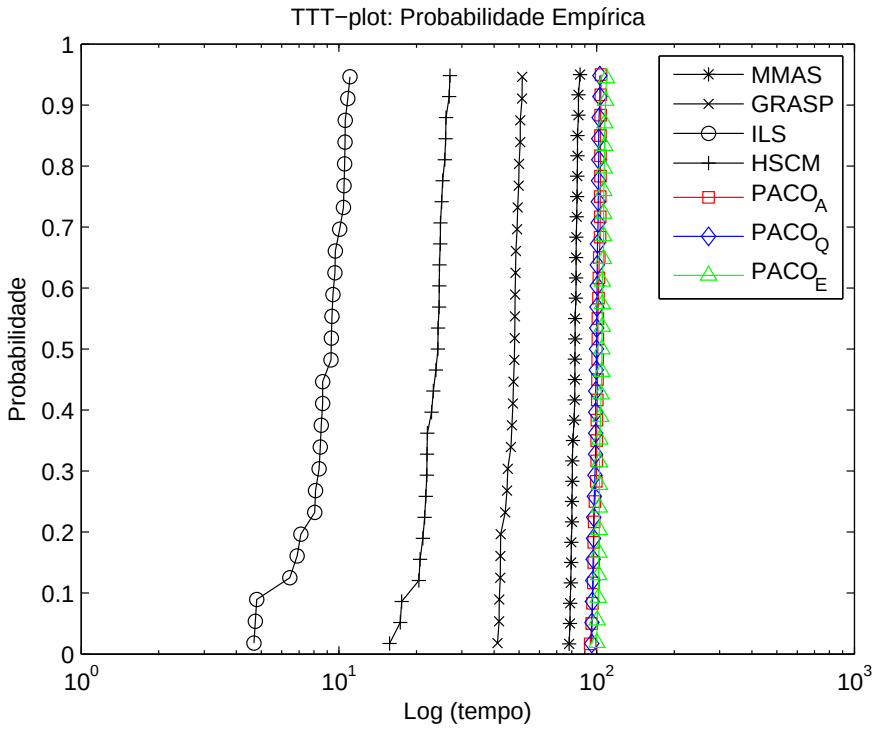


Figura 6.4: Time-to-target para a instância R205.

Teste de Probabilidade Empírica para a instância C204

A Figura 6.3 mostra o resultado do Teste de Probabilidade Empírica para a instância C204.

Ao analisar os resultados referentes aos testes em que foi utilizada a instância C204, percebe-se que algoritmo HSCM + ILS demonstrou ser o algoritmo mais eficiente, alcançando o alvo em 95% das execuções. Os algoritmos ILS e GRASP + ILS alcançam o alvo com probabilidades de aproximadamente 90% e 85%, respectivamente. O algoritmo MMAS + ILS, por sua vez, alcançou o alvo em 63%. das execuções. Em relação ao tempo, o algoritmo P-ACO$_{\text{Elitist}}$ + ILS teve o maior custo computacional, principalmente para alcançar o alvo com probabilidade superior a 43%, aproximadamente. Mesmo neste caso, também não demonstrou ser muito eficaz alcançando o alvo em apenas 53% das execuções. O algoritmo P-ACO$_{\text{Quality}}$ + ILS conseguiu alcançar o alvo em 53% das execuções. O P-ACO$_{\text{Age}}$ + ILS mostrou ser o algoritmo menos eficiente, alcançando o alvo em apenas 43% das execuções.

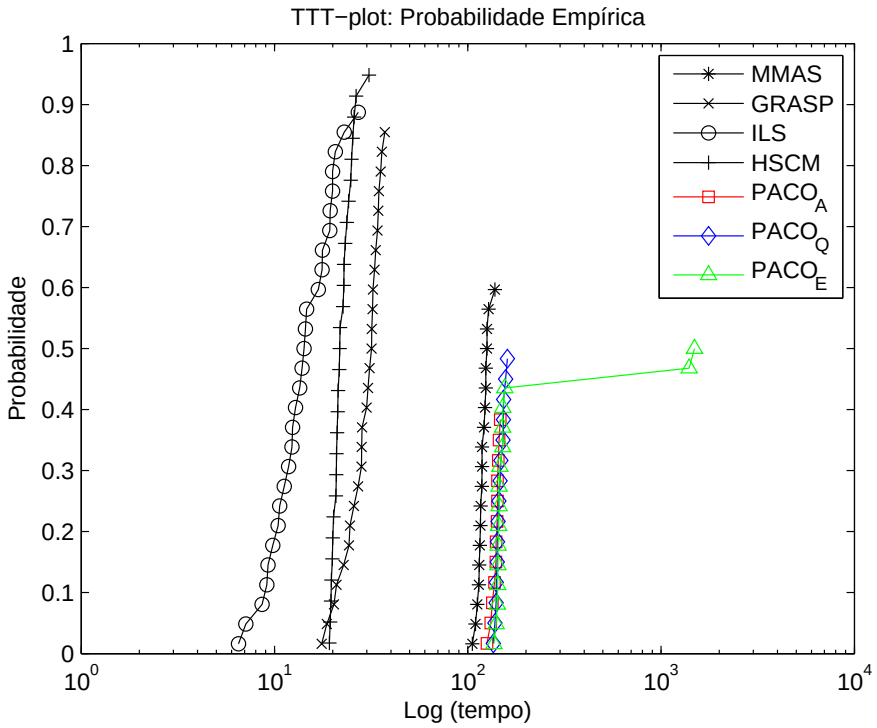


Figura 6.5: Time-to-target para a instância C204.

Teste de Probabilidade Empírica para a instância RC204

A Figura 6.3 mostra o resultado do Teste de Probabilidade Empírica para a instância RC204.

Em relação aos resultados sobre os testes feitos para a instância RC204, o ILS mostrou ser o algoritmo com o melhor desempenho, tendo o menor custo de tempo computacional. Sobre a eficiência, todos os algoritmos testados mostraram ser capazes de alcançar o alvo em 100% das execuções. Ao se analisar o gráfico, é também possível perceber que, para alcançar o alvo com probabilidade acima de 10%, aproximadamente, apesar de terem apresentado custos de tempo computacional com valores próximos, o algoritmo GRASP + ILS apresentou um custo de tempo menor do que o algoritmo HSCM + ILS. Nesse teste, apesar de terem demonstrado eficiência, os algoritmos P-ACO$_{\text{Age}}$ + ILS, P-ACO$_{\text{Quality}}$ + ILS e P-ACO$_{\text{Elitist}}$ + ILS tiveram custos computacionais semelhantes, sendo os maiores dentre os algoritmos testados para atingir o alvo.

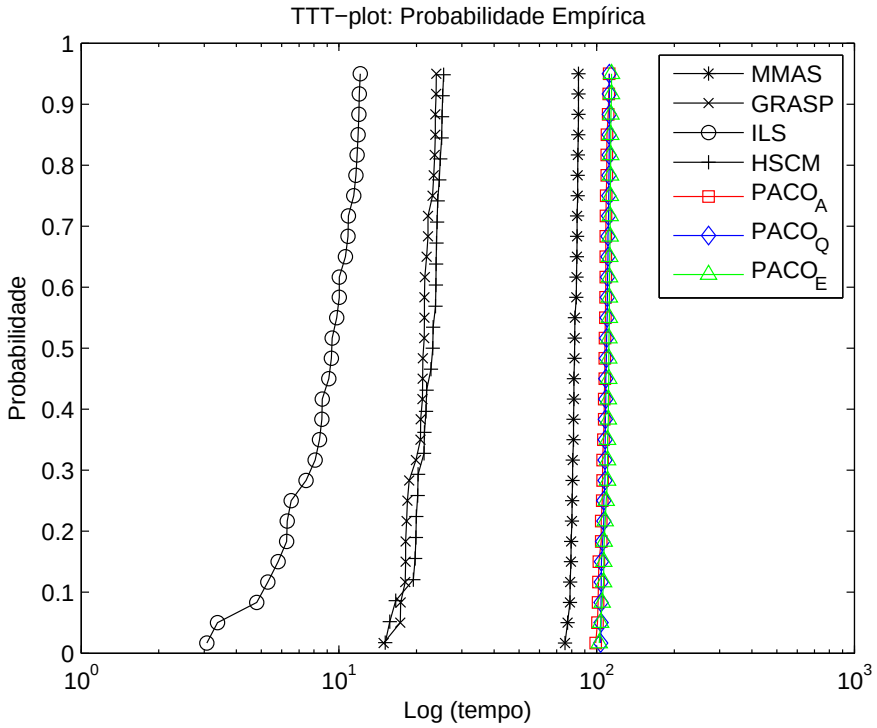


Figura 6.6: Time-to-target para a instância RC204.

6.2 Dados Gerais das Soluções

6.2.1 Geração de Novas Soluções

De acordo com os dados obtidos por meio dos testes, em que foram aplicados os algoritmos desenvolvidos nesse trabalho sobre as instâncias propostas por Solomon (1987), foi possível observar que, ao serem aplicados sobre determinados conjuntos, certos algoritmos conseguiram obter um número maior de soluções melhores do que as presentes na literatura. As figuras a seguir mostram os gráficos com os percentuais de soluções melhores que as existentes na literatura que cada algoritmo foi capaz de gerar para os conjuntos de instâncias utilizados.

Na Figura 6.7 são apresentados os percentuais das soluções melhores que as presentes na literatura que foram geradas pelos algoritmos durante os testes feitos usando os conjuntos R1 e R2. No caso do conjunto R1, percebe-se que todos os algoritmos foram capazes de encontrar, para pelo menos 50% das instâncias, soluções melhores que as existentes na literatura. O algoritmo P-ACO$_{\text{Elitist}}$ + ILS foi o que encontrou maior número de novas soluções, no caso, em 75% dos testes. Para o conjunto R2, nenhum dos algoritmos desenvolvidos conseguiu pelo menos alcançar o resultados vistos na literatura.

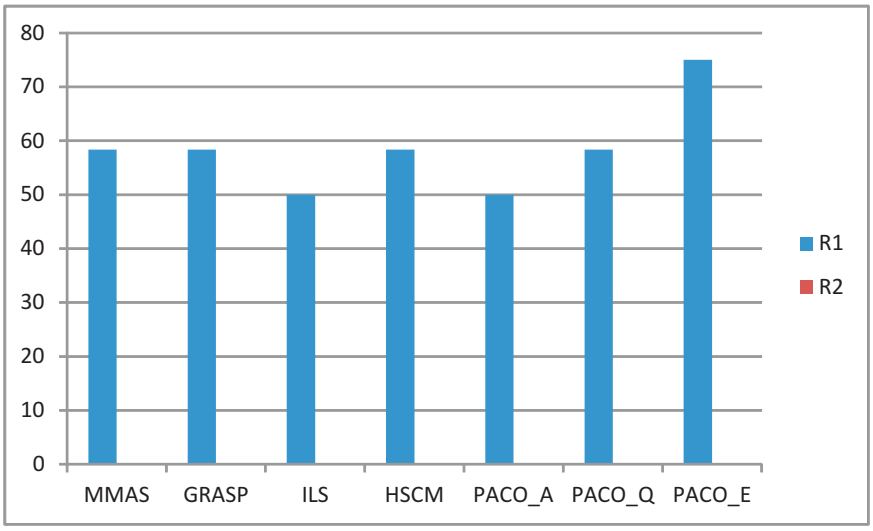


Figura 6.7: Percentual de soluções novas encontradas para os conjuntos R1 e R2.

Com base no gráfico mostrado na Figura 6.8, é possível observar os percentuais de novas soluções que os algoritmos desenvolvidos foram capazes de encontrar, quando aplicados sobres os conjuntos C1 e C2. Percebe-se que a maioria dos algoritmos foi capaz de encontrar boas soluções para 100% das instâncias do conjunto C1, com exceção do algoritmo P-ACO$_{\text{Elitist}}$ + ILS, que encontrou boas soluções para 88,89%. Em relação às instâncias do conjunto C2, todos os algoritmos encontraram boas soluções para a maior partes das instâncias deste conjunto. Para o conjunto C2, os algoritmos MMAS + ILS, GRASP + ILS, HSCM + ILS e P-ACO$_{\text{Age}}$ + ILS tiveram 87,5% de aproveitamento e os algoritmos ILS, P-ACO$_{\text{Quality}}$ + ILS e $P-ACO_{Elitist}$+ILS obtiveram 75% de aproveitamento.

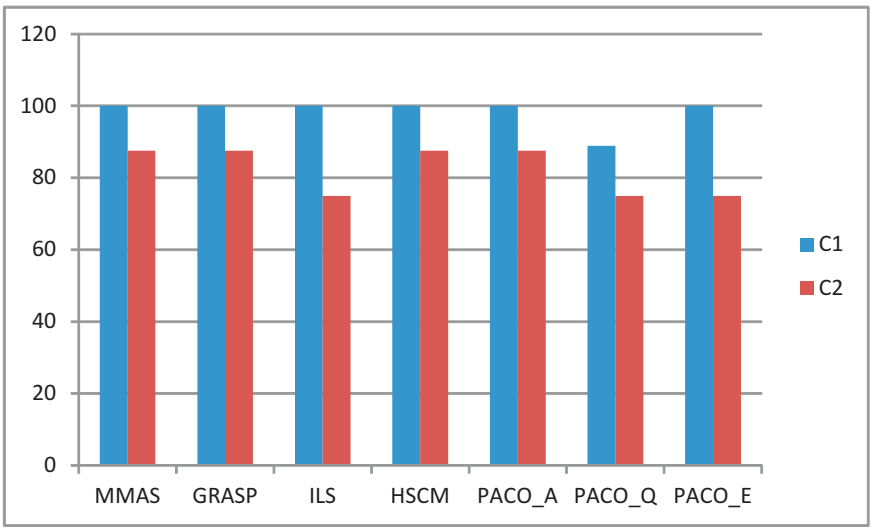


Figura 6.8: Percentual de novas soluções encontradas para os conjuntos C1 e C2.

Na Figura 6.9 são mostrados os gráficos com os percentuais de soluções encontradas pelos algoritmos testados para os conjuntos RC1 e RC2. Para as instâncias do conjunto RC1, os algoritmos usados demonstraram um bom aproveitamento, sendo capazes de encontrar boas soluções na maioria dos casos. Os algoritmos GRASP + ILS, P-ACO$_{\text{Age}}$ + ILS, P-ACO$_{\text{Quality}}$ + ILS e P-ACO$_{\text{Elitist}}$ + ILS obtiveram boas soluções para 87,5% das instâncias. Já os algoritmos MMAS + ILS, ILS e HSCM + ILS obtiveram boas soluções em 75%, 75% e 62,5% dessas instâncias, respectivamente. No caso das instâncias do conjunto RC2, estes algoritmos não encontraram bom desempenho, tendo boas soluções para apenas 12,5% das soluções pertencentes a esse conjunto, com exceção do algoritmo MMAS + ILS, que foi capaz de encontrar boas soluções em 25% das instâncias.

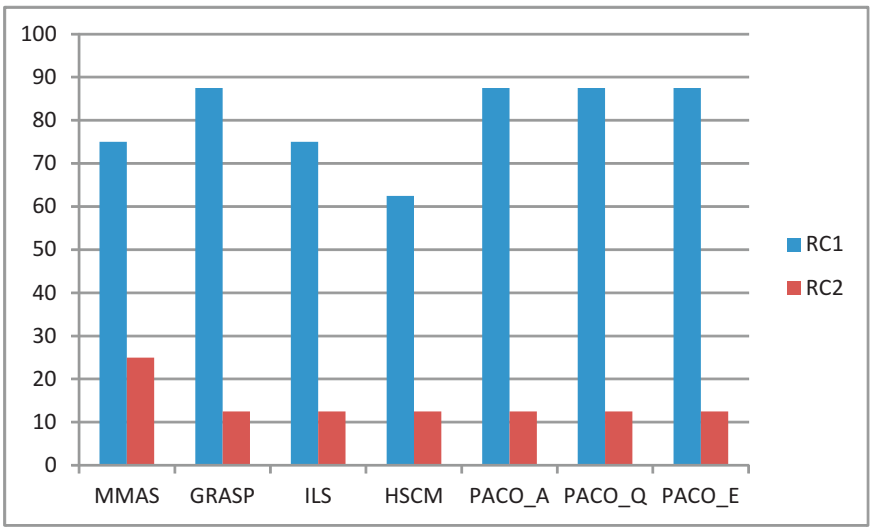


Figura 6.9: Percentual de novas soluções encontradas para os conjuntos RC1 e RC2.

6.2.2 Sumário de Resultados

Nas Tabelas 6.3 e 6.4 são apresentados os resultados obtidos, para cada conjunto das instâncias de Solomon, por Repoussis et al. (2009) e pelos algoritmos implementados. Na Tabela 6.5 mostra o somatório dos resultados gerais em que pode ser visto que, considerando os resultados obtidos para todas as 56 instâncias de Solomon, o MMAS+ILS obteve umas das menores somas em relação ao número de veículos e a menor distância percorrida.

Tabela 6.3: Dados Gerais dos Resultados paras os Conjuntos R1, C1 e RC1

	Conjuntos					
	R1		**C1**		**RC1**	
Algoritmos	**DT**	**NV**	**DT**	**NV**	**DT**	**NV**
Repoussis (2009)	11375,89	143	5004,09	90	8123,81	92
MMAS+ILS	11359,80	141	**5002,75**	**90**	7699,46	92
ILS	11336,91	142	**5002,75**	**90**	7605,46	93
GRASP+ILS	11331,23	142	**5002,75**	**90**	7714,29	92
HSCM+ILS	11340,30	142	**5002,75**	**90**	7667,75	92
$P-ACO_{Age}$+**ILS**	11410,70	142	**5002,75**	**90**	7668,94	92
$P-ACO_{Quality}$+**ILS**	11299,07	142	5003,13	90	**7626,94**	**92**
$P-ACO_{Elitist}$+**ILS**	**11318,42**	**141**	**5002,75**	**90**	7672,41	92

Tabela 6.4: Dados Gerais dos Resultados paras os Conjuntos R2, C2 e RC2

	Conjuntos					
	R2		**C2**		**RC2**	
Algoritmos	**DT**	**NV**	**DT**	**NV**	**DT**	**NV**
Repoussis (2009)	**10023,85**	**30**	4374,5	24	8387,15	26
MMAS+ILS	10077,69	33	4378,15	24	8380,48	25
ILS	9988,80	33	4564,33	24	8253,25	26
GRASP+ILS	9997,01	33	4526,63	24	8453,28	25
HSCM+ILS	9835,67	34	**4373,74**	**24**	8304,80	26
$P-ACO_{Age}$+**ILS**	10062,94	33	4401,39	24	**8170,92**	**25**
$P-ACO_{Quality}$+**ILS**	10017,90	33	4455,91	24	8373,50	25
$P-ACO_{Elitist}$+**ILS**	9909,96	34	4400,26	24	8482,41	25

Tabela 6.5: Dados Gerais dos Resultados

	Total	
Algoritmos	**DT**	**NV**
Repoussis (2009)	47289,29	405
MMAS+ILS	**46898,33**	**405**
ILS	46751,50	408
GRASP+ILS	47025,19	406
HSCM+ILS	46525,01	408
$P-ACO_{Age}$+**ILS**	46717,64	406
$P-ACO_{Quality}$+**ILS**	46776,45	406
$P-ACO_{Elitist}$+**ILS**	46786,21	406

6.3 Análise Estatística e Teste Kruskal-Wallis

6.3.1 Apresentação do Teste Kruskal-Wallis

Com a finalidade de comparar os resultados obtidos pelos algoritmos desenvolvidos neste trabalho, foi utilizado o Teste de Kruskal-Wallis. Segundo Field (2009), o método de Kruskal-Wallis é um teste não-paramétrico, que serve para testar a hipótese de que diferentes amostras provém da mesma população ou de populações idênticas. Esse teste é usado para verificar se várias amostras possuem funções de distribuição iguais. Diferente de outros métodos, este teste não exige que as amostras analisadas tenham distribuições normais. Quando é feita a aplicação do Teste de Kruskak-Wallis, a estatística de teste H é calculada, cuja distribuição pode ser aproximada pela distribuição χ-quadrado, desde que cada amostra contenha pelo menos cinco observações. Ao ser aplicada a distribuição χ-quadrado, o valor do grau de liberdade é igual a $k-1$, em que k é a quantidade de amostras. Na Tabela 6.6, é apresentada a notação usada no Teste de Kruskal-Wallis (Triola, 2008).

Tabela 6.6: Notação usada no Teste de Kruskal-Wallis.

Variável	Descrição
N	número total de observações em todas as amostras combinadas
k	número de amostras
R_i	soma dos postos da amostra i
η_i	número de observações na amostra i

Para Triola (2008), dentre as etapas necessárias para a realização do Teste de Kruskal-Wallis, é necessário, primeiramente, atribuir um posto para cada uma das observações de todas as amostras, sendo o valor do posto iniciando do valor mais baixo e indo para o mais alto. Quanto ocorrem empates, é atribuído, para cada uma das observações que possuem valores iguais, um valor referente à média dos postos envolvidos. Ao ser definido o valor do posto de cada uma das observações, é determinado, para cada amostra, o seu tamanho e a soma dos postos de suas respectivas observações. Com base nos valores referentes à soma dos postos e do tamanho de cada amostra, é realizado o cálculo da estatística de teste H, que tem distribuição χ-quadrado e com $k-1$ graus de liberdade. Para a realização desse teste, é também considerado um valor crítico X^2, que se obtém na tabela Distribuição χ-Quadrado de valores X^2. Para se especificar o valor crítico adotado, é levado em conta o grau de liberdade $k-1$ e o nível de significância α, que, por padrão, tem o valor igual a 0.05, quando o problema não possui alguma condição específica que faça com que deva ser adotado outro valor para α.

De acordo com a notação apresentada na Tabela 6.6, o valor de H é calculado conforme a expressão:

$$H = \frac{12}{N\,(N+1)}\left(\frac{R_1^2}{\eta_1} + \frac{R_2^2}{\eta_2} + \cdots + \frac{R_n^2}{\eta_n}\right) - 3\,(N+1) \tag{6.1}$$

A estatística teste H pode ser considerada uma medida da variância das somas dos postos em que, quando os valores dos postos são distribuídos de forma equivalente, o valor de H é pequeno, mas quando a diferença entre as amostra é grande, o valor de H se torna alto. Como o teste de Kruskal-Wallis trabalha com duas hipóteses, sendo uma que a diferença entre as amostras é nula (hipótese nula) e a outra de que há uma diferença entre pelo menos duas amostras (hipótese alternativa), quando o valor de H é alto, a primeira hipótese, por consequência, é rejeitada. Resumindo, com a definição do valor crítico e o cálculo de H pode ser feita a verificação se a hipótese nula deve ou não ser rejeitada. Quando o valor de H é superior ao valor crítico X^2, considera-se que a estatística teste H está na região crítica delimitada por X^2 e a hipótese nula deve ser rejeitada. Maiores detalhes sobre o Teste de Kruskal-Wallis podem ser vistos em Field (2009).

6.3.2 Aplicação do Teste Kruskal-Wallis

Como já foi foi descrito na Seção 6.3, o Teste de Kruskal-Wallis foi usado para comparar os resultados obtidos pelos algoritmos implementados nesse trabalho, com o objetivo de se verificar a possibilidade de haver uma diferença significativa entre os mesmos. Com isso, foram escolhidas 6 diferentes instâncias, cada uma pertencente a um dos 6 diferentes conjuntos de instâncias com 100 clientes de Solomon. Assim como foi feito para a realização dos testes de probabilidade empírica, foram escolhidas as instâncias R102, C104, RC104, R205, C204 e RC204. Nas tabelas 6.8 e 6.9, são apresentados os resultados obtidos por meio do Teste de Kruskal-Wallis, em que foram feitas comparações entre os resultados obtidos por todos os algoritmos usados para as instâncias escolhidas, tanto em relação ao número de veículos quanto pela distância total percorrida referentes a cada solução. Desta forma, em cada tabela, é informado se, entre cada par de algoritmos, a diferença de variância das soluções em relação ao número de veículos ou pela distância total percorrida foram significantes.

Nesta seção, nas tabelas 6.8, 6.9 e nas Figuras 6.10 a 6.15, para identificar quais são os dados respectivos aos algoritmos desenvolvidos, são utilizadas diferentes legendas, que são apresentadas na Tabela 6.7:

Tabela 6.7: Legendas.

Legenda	Algoritmo
i) MMAS	MMAS+ILS
ii) GRASP	GRASP+ILS
iii) ILS	ILS
iv) HSCM	HSCM+ILS
v) $PACO_A$	$P-ACO_{Age}$+ILS
vi) $PACO_Q$	$P-ACO_{Quality}$+ILS
vii) $PACO_E$	$P-ACO_{Elitist}$+ILS

Tabela 6.8: Resultados para o Teste de Kruskal-Wallis

	Instâncias					
	R102		**C104**		**RC104**	
Agoritmos Comparados	**Distância**	**Veículos**	**Distância**	**Veículos**	**Distância**	**Veículos**
GRASP × **HSCM**	Não	Não	Não	Não	Não	Não
GRASP × **ILS**	Não	Não	Não	Não	Não	Não
GRASP × **MMAS**	Não	Não	Não	Não	Não	Não
GRASP × $PACO_A$	Não	Não	Não	Não	Não	Não
GRASP × $PACO_E$	Não	Não	Não	Não	Não	Não
GRASP × $PACO_Q$	Não	Não	Não	Não	Não	Não
HSCM × **ILS**	Não	Não	Não	Não	**Sim**	Não
HSCM × **MMAS**	Não	Não	**Sim**	Não	**Sim**	Não
HSCM × $PACO_A$	Não	Não	**Sim**	Não	Não	Não
HSCM × $PACO_E$	Não	Não	**Sim**	Não	Não	Não
HSCM × $PACO_Q$	Não	Não	**Sim**	Não	Não	Não
ILS × **MMAS**	Não	Não	Não	Não	Não	Não
ILS × $PACO_A$	Não	Não	Não	Não	Não	Não
ILS × $PACO_E$	Não	Não	Não	Não	**Sim**	Não
ILS × $PACO_Q$	Não	Não	Não	Não	Não	Não
MMAS × $PACO_A$	Não	Não	Não	Não	Não	Não
MMAS × $PACO_E$	Não	Não	Não	Não	**Sim**	Não
MMAS × $PACO_Q$	Não	Não	Não	Não	Não	Não
$PACO_A$ × $PACO_E$	Não	Não	Não	Não	Não	Não
$PACO_A$ × $PACO_Q$	Não	Não	Não	Não	Não	Não
$PACO_E$ × $PACO_Q$	Não	Não	Não	Não	Não	Não

Tabela 6.9: Resultados para o Teste de Kruskal-Wallis

	R205		**C204**		**RC204**	
Agoritmos Comparados	**Distância**	**Veículos**	**Distância**	**Veículos**	**Distância**	**Veículos**
GRASP × **HSCM**	Não	Não	**Sim**	Não	**Sim**	Não
GRASP × **ILS**	Não	Não	Não	Não	Não	Não
GRASP × **MMAS**	Não	Não	Não	Não	Não	Não
GRASP × $PACO_A$	Não	Não	Não	Não	Não	Não
GRASP × $PACO_E$	Não	Não	Não	Não	Não	Não
GRASP × $PACO_Q$	**Sim**	Não	Não	Não	Não	Não
HSCM × **ILS**	Não	Não	**Sim**	Não	Não	Não
HSCM × **MMAS**	Não	Não	**Sim**	Não	Não	Não
HSCM × $PACO_A$	Não	Não	**Sim**	Não	**Sim**	Não
HSCM × $PACO_E$	Não	Não	**Sim**	Não	**Sim**	Não
HSCM × $PACO_Q$	**Sim**	Não	**Sim**	Não	**Sim**	Não
ILS × **MMAS**	Não	Não	Não	Não	Não	Não
ILS × $PACO_A$	Não	Não	Não	Não	Não	Não
ILS × $PACO_E$	Não	Não	Não	Não	Não	Não
ILS × $PACO_Q$	**Sim**	Não	Não	Não	Não	Não
MMAS × $PACO_A$	Não	Não	Não	Não	Não	Não
MMAS × $PACO_E$	Não	Não	Não	Não	Não	Não
MMAS × $PACO_Q$	**Sim**	Não	Não	Não	Não	Não
$PACO_A$ × $PACO_E$	Não	Não	Não	Não	Não	Não
$PACO_A$ × $PACO_Q$	**Sim**	Não	Não	Não	Não	Não
$PACO_E$ × $PACO_Q$	**Sim**	Não	Não	Não	Não	Não

6.4 Análise dos Resultados através de Gráficos Box-Plot

Tendo em vista os resultados apresentados pelas tabelas 6.8 e 6.9, percebe-se que, considerando-se apenas o número de veículos, em nenhum dos testes foi visto que houve uma diferença significativa entre os algoritmos implementados ao serem aplicados sobre as instâncias escolhidas. Ao se considerar, também, a questão da distância total percorrida pelos veículos, é possível observar que os resultados entre os algoritmos, para algumas das instâncias usadas, demonstraram ter uma diferença significativa entre si. No caso do conjunto de instâncias R102, de acordo os resultados do Teste de Kruskal-Wallis, não houve uma diferença significativa entre os resultados apresentados pelos algoritmos tanto em relação ao número de veículos quanto em relação à distância total percorrida. A Figura 6.10 apresenta os gráficos de caixa relativos à variabilidade dos resultados dos algoritmos quando aplicados sobre a instância R102.

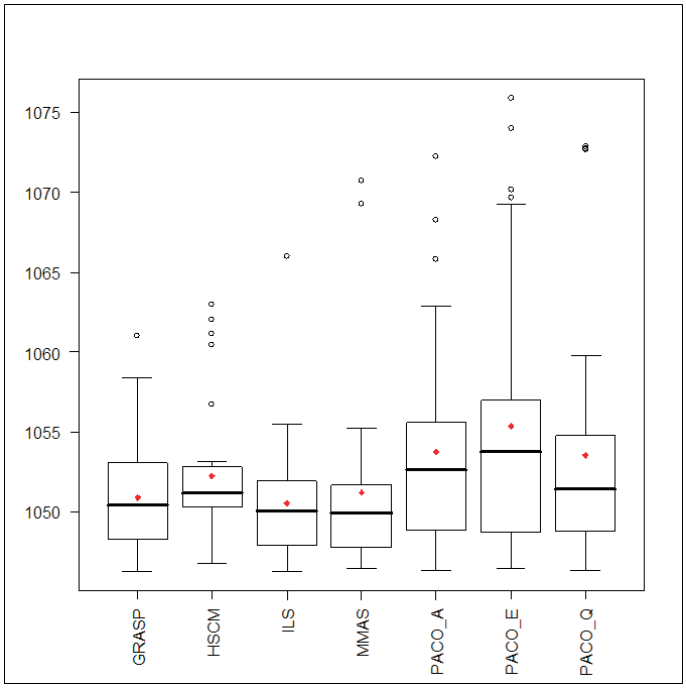


Figura 6.10: Gráficos BoxPlot acerca da distância para o conjunto R102.

Para o conjunto de instâncias C104, ao se comparar os resultados do algoritmo HSCM+ILS com os dos algoritmos MMAS, $P-ACO_{Age}$+ILS, $P-ACO_{Quality}$+ILS e $P-ACO_{Elitist}$+ILS, verificou-se que houve uma diferença significativa entre eles. Por meio dos gráficos na Figura 6.11, pode se confirmar que, de acordo com a distribuição dos resultados, os valores apresentados pelo algoritmo HSCM+ILS são, praticamente todos, inferiores se comparados com os dos algoritmos com que houve uma diferença significativa. Nota-se que o terceiro quartil *boxplot* do algoritmo HSCM+ILS, por exemplo, é menor ou pelo menos pouco superior aos valores acerca do primeiro quartil dos algoritmos MMAS, $P-ACO_{Age}$+ILS, $P-ACO_{Quality}$+ILS e $P-ACO_{Elitist}$+ILS. Os valores referentes à mediana e à média dos algoritmos MMAS, $P-ACO_{Age}$+ILS, $P-ACO_{Quality}$+ILS e $P-ACO_{Elitist}$+ILS foram todos superiores ao terceiro quartil do HSCM+ILS. A Figura 6.11 apresenta os gráficos de caixa relativos à variabilidade dos resultados dos algoritmos quando aplicados sobre a instância C104.

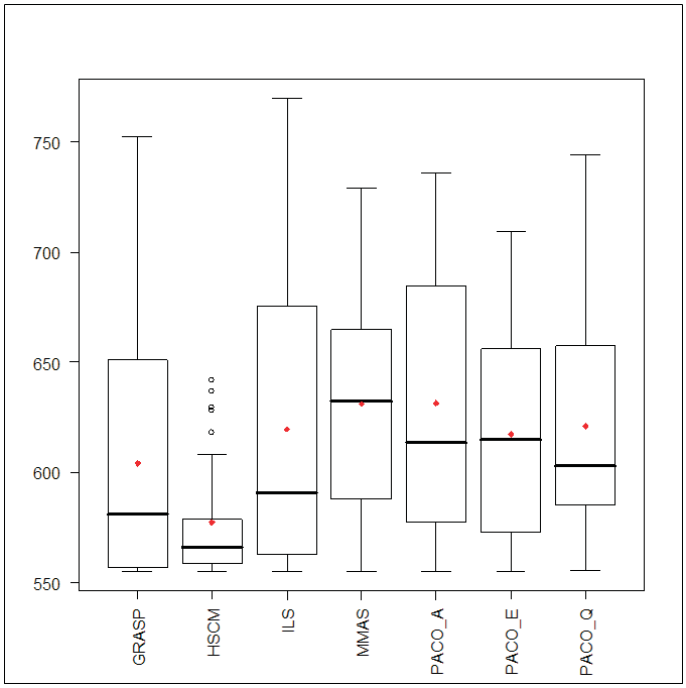


Figura 6.11: Gráficos BoxPlot acerca da distância para o conjunto C104.

De acordo a Tabela 6.8, verificou-se que houve uma diferença significativa ao se comparar os algoritmos HSCM+ILS com os algoritmos ILS e MMAS+ILS, e o algoritmo $P - ACO_{Elitist}$+ILS com o ILS e o MMAS+ILS, quando aplicados sobre a instância RC104. Com o auxílio dos gráficos mostrados na Figura 6.12, pode-se averiguar que, conforme a distribuição dos resultados, as maiores diferenças são entre os algoritmos que, pelo Teste de Kruskal-Wallis, foram constatados que possuem diferenças significativas. Além disso, os algoritmos HSCM+ILS e o $P - ACO_{Elitist}$+ILS possuem grande parte de suas soluções com os valores mais altos em relação aos dos demais algoritmos, enquanto que a maioria das soluções do ILS e do MMAS+ILS possuem os menores valores. A Figura 6.12 apresenta os gráficos de caixa relativos à variabilidade dos resultados dos algoritmos, quando aplicados sobre a instância RC104.

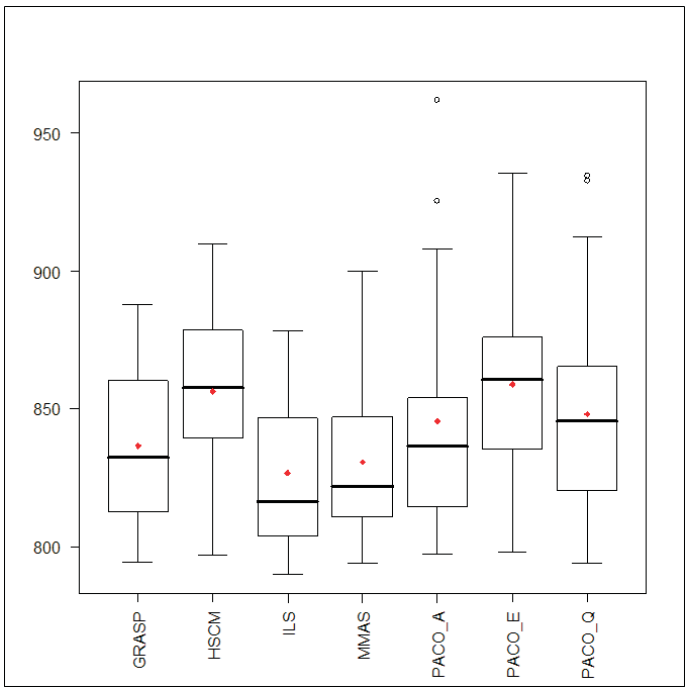


Figura 6.12: Gráficos BoxPlot acerca da distância para o conjunto RC104.

Conforme a Tabela 6.9, percebe-se que houve uma diferença significativa ao se comparar os resultados do algoritmo $P-ACO_{Quality}$+ILS com os de todos os demais algoritmos ao serem testados utilizando a instância R205. Ao se analisar os gráficos da Figura 6.13, observa-se que, de acordo com a distribuição dos valores das soluções geradas pelos algoritmos, a maior parte das soluções do $P-ACO_{Quality}$+ILS apresentaram valores inferiores aos da maioria dos demais algoritmos, tanto é que o valor máximo obtido por ele é inferior ao do primeiro quartil dos gráficos de todos os outros algoritmos, o que ajuda a reforçar os resultados que indicam a diferença significativa entre tais soluções. A Figura 6.13 apresenta os gráficos de caixa relativos à variabilidade dos resultados dos algoritmos quando aplicados sobre a instância R205.

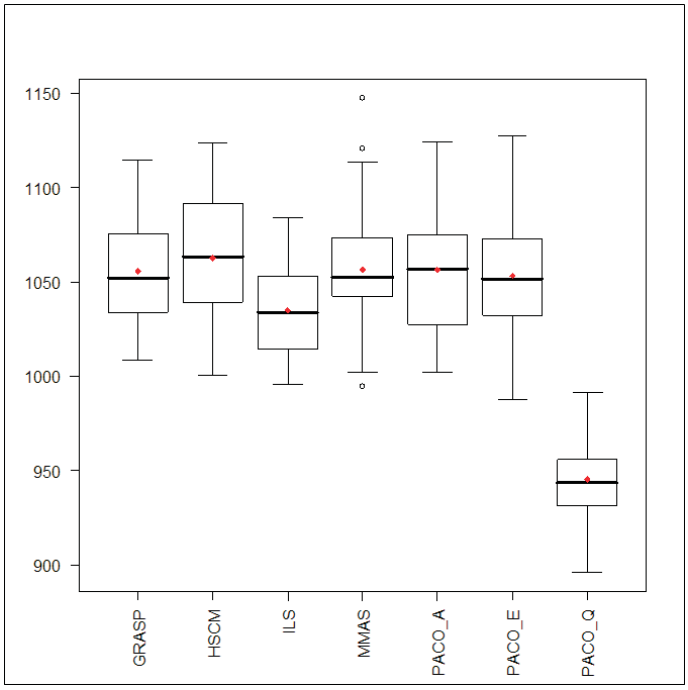


Figura 6.13: Gráficos BoxPlot acerca da distância para o conjunto R205.

Conforme a Tabela 6.9, em relação ao conjunto de instâncias C204, viu-se que há uma diferença significativa, ao se comparar as soluções do HSCM+ILS com as dos demais algoritmos implementados. Por meio dos gráficos da Figura 6.14, que apresentam a distribuição dos valores, pode ser visto que, além da baixa variabilidade, em que a maioria dos resultados do HSCM+ILS são muito próximos da mediana, os mesmos sãos significativamente inferiores aos das demais soluções, sendo que o valor máximo do HSCM+ILS é menor do que o primeiro quartil dos gráficos referentes as soluções dos outros algoritmos. A Figura 6.14 apresenta os gráficos de caixa relativos a variabilidade dos resultados dos algoritmos quando aplicados sobre a instância C204.

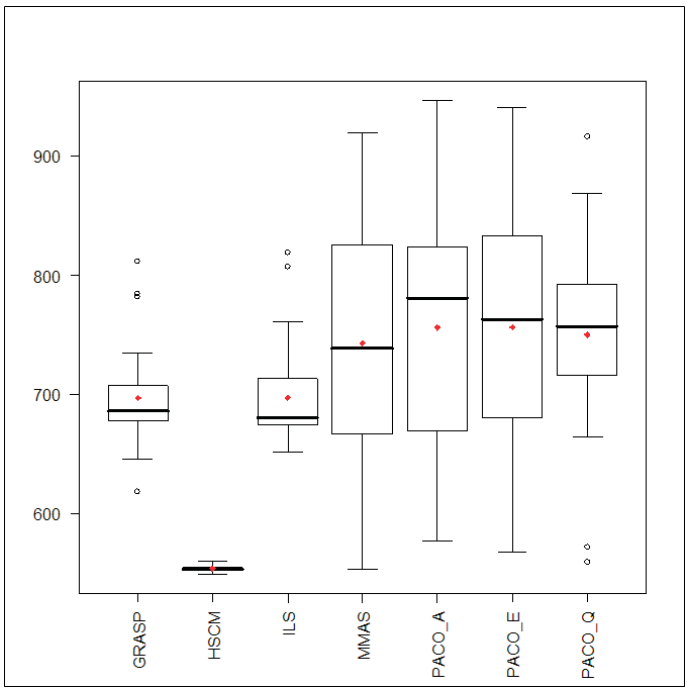


Figura 6.14: Gráficos BoxPlot acerca da distância para o conjunto C204.

Segundo a Tabela 6.9, pode ser visto que, em relação aos resultados obtidos para a instância RC204, os valores das soluções do HSCM+ILS apresentaram diferenças significativas ao serem comparadas com as dos algoritmos GRASP+ILS, $P-ACO_{Age}$+ILS, $P-ACO_{Quality}$+ILS e $P-ACO_{Elitist}$+ILS. Na Figura 6.15 são apresentados os gráficos acerca da distribuição dos valores das soluções dos algoritmos para a instância RC204. Ao se analisar os gráficos, percebe-se que, de acordo com a sua distribuição, grande parte dos valores das soluções do HSCM+ILS são maiores do que as dos algoritmos GRASP+ILS, $P-ACO_{Age}$+ILS, $P-ACO_{Quality}$+ILS e $P-ACO_{Elitist}$+ILS. Valores como o primeiro quartil do gráfico referente aos resultados do HSCM+ILS são maiores do que as medianas de todos os algoritmos que, conforme o Teste de Kruskal+Wallis, apresentaram resultados com diferença significativa ao serem comparados com os de tal algoritmo. A Figura 6.15 apresenta os gráficos de caixa relativos a variabilidade dos resultados dos algoritmos quando aplicados sobre a instância RC204.

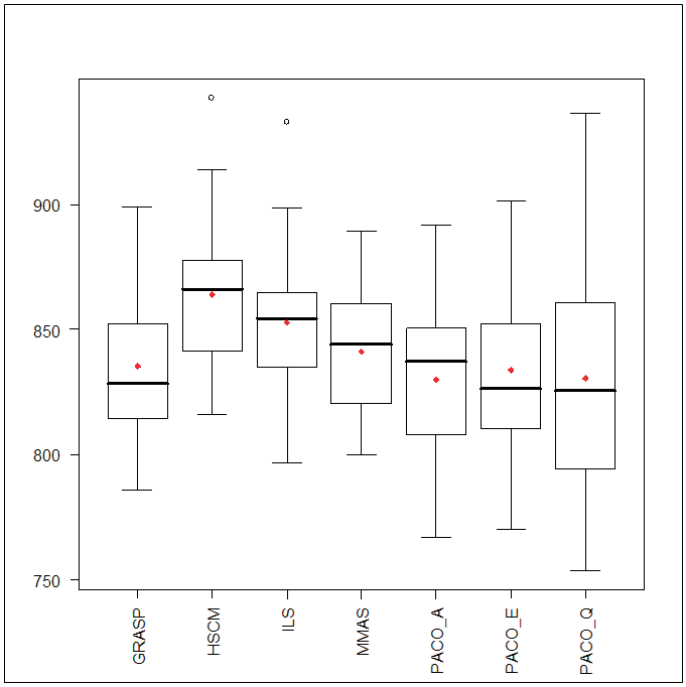


Figura 6.15: Gráficos BoxPlot acerca da distância para o conjunto RC204.

Capítulo 7

Considerações Finais

Esta dissertação apresentou um estudo acerca da aplicação de metaheurísticas ao Problema de Roteamento Aberto de Veículos com Janelas de Tempo (PRAVJT). O PRAVJT é uma variação do Problema de Roteamento de Veículos com Janela de Tempo em que os veículos não necessitam retornar ao depósito após servirem o último cliente.

Para solucionar o PRAVJT, foram implementados os algoritmos MMAS+ILS, GRASP+ILS, ILS, HSCM+ILS, $P-ACO_{Age}$+ILS, $P-ACO_{Quality}$+ILS e $P-ACO_{Elitist}$+ILS. Os algoritmos desenvolvidos são baseados na combinação entre as metaheurísticas Colônia de Formigas, GRASP, ILS, HSCM e PFIH. Por meio deste trabalho, foi possível desenvolver diferentes técnicas, de forma adaptada, para resolver o PRAVJT, como é o caso das estratégias ER e ERFO, que serviram com meio de auxílio na exploração do espaço de busca, permitindo a obtenção de soluções de maior qualidade.

Além disso, também foi proposta uma nova heurística para a construção de soluções para problemas de roteamento de veículos, a heurística HSCM, que tem como diferencial a questão de levar em conta as principais características do Problema de Roteamento de Veículos com Janelas de Tempo para definir a forma de se construir determinadas soluções. Diferente do PFIH, que realiza a inserção dos clientes nas rotas, focando apenas na questão da minimização da distância total percorrida e da não violação das restrições que compõem o problema, o HSCM busca também construir soluções que tenha o menor número possível de rotas criadas por meio da definição de um número de rotas inicial, possibilitando entre os veículos uma distribuição mais uniforme do atendimento dos clientes.

Com a finalidade de testar a metodologia proposta foram utilizados os problemas teste de Solomon com 100 clientes que são formados por 56 instâncias que se dividem em seis diferentes conjuntos, conforme as suas características. Também foi utilizado para a efetuação dos testes parte das instâncias de Homberger e Gehring.

De uma maneira geral, utilizando as instâncias de Solomon, os algoritmos desenvolvidos apresentaram resultados promissores, sendo capazes de encontrar soluções melhores que as presentes na literatura em pelo menos 50% das instâncias testadas, como é o caso do ILS, que utilizou apenas o PFIH como meio de geração da solução inicial.

O HSCM+ILS gerou soluções melhores que as das literatura em 51,8% dos casos, demonstrando um rendimento um pouco melhor do que o ILS.

O $P - ACO_{Quality}$+ILS obteve um rendimento similar ao HSCM+ILS, gerando soluções superiores aos da literatura para 51,8% das instâncias.

O $P - ACO_{Age}$+ILS, para 53,6% das instâncias testadas, apresentou resultados de maior qualidade do que os da literatura. Os algoritmos MMAS+ILS e GRASP+ILS conseguiram alcançar bons resultados, encontrado soluções melhores que as da literatura para 55,35% dos problemas teste. O $P - ACO_{Elitist}$+ILS mostrou um rendimento superior aos demais algoritmos, alcançando para 57,14% das instâncias resultados com maior qualidade, se comparadas as da literatura.

Ao ser aplicado sobre os conjuntos R1, C1 e RC1 das instâncias de Homberger e Gehring com 200, 400 e 600 clientes, o MMAS+ILS demonstrou ser eficiente papa trabalhar com instâncias maiores, obtendo para as instâncias com 200 e 400 clientes utilizadas soluções melhores do que as apresentadas na literatura em 100% dos casos. Para os conjuntos R1 e RC1 das instâncias com 600 clientes o MMAS+ILS apresentou ser eficiente também, em que ele alcançou soluções melhores do que as da literatura em 70% dos casos.

Para avaliar o desempenho dos algoritmos implementados, foram feitos diferentes testes com eles em relação ao seu desempenho computacional, capacidade de alcançar boas soluções e estabilidade. Por meio do teste de probabilidade empírica, em que foram usadas as instâncias R102, C104, RC104, R205, C204 e RC204, pode-se observar a eficiência dos algoritmos, levando-se em conta o tempo computacional e a probabilidade de alcançar boas soluções. Neste caso, pode ser visto que, em relação ao tempo computacional, o ILS foi melhor do que os demais algoritmos na maioria dos testes, enquanto que os algoritmos $P - ACO_{Age}$+ILS, $P - ACO_{Quality}$+ILS e $P - ACO_{Elitist}$+ILS foram os que tiveram maior custo.

Quanto à eficiência para se alcançar soluções com determinado nível de qualidade (soluções alvo), verificou-se que para os problemas teste R102, R205 e RC204 todos os algoritmos alçaram o alvo em 100% das execuções. De acordo com os resultados obtidos pelos algoritmos quando aplicados ao problema teste C104, todos eles demonstraram bom desempenho, alcançado o alvo em pelo menos 70% das execuções. Conforme os resultados obtidos para a instância RC104, a maioria dos algoritmos testados não demonstraram tão bom desempenho como no teste anterior, sendo que apenas o GRASP+ILS foi capaz de alcançar a solução alvo em mais de 73% das execuções, enquanto que algoritmos como o ILS alcançaram o alvo em apenas 43% das execuções. Segundo os resultados dos testes referentes a instância C204, os algoritmos HSCM+ILS, GRASP+ILS e ILS tiveram um bom desempenho alcançando a solução alvo em mais de 90% das execuções, já os algoritmos como o $P - ACO_{Age}$+ILS não apresentaram um bom desempenho, alcançando a solução alvo em apenas 43% dos casos.

Para comparar os algoritmos de maneira a verificar se há uma diferença significativa entre as soluções obtidas por eles e, se houver, verificar quais algoritmos dentre eles tiveram um melhor desempenho, foram usados em conjunto o Teste de Kruskal-Wallis e o gráfico de diagramas (BoxPlot). Com isso, verificou-se que, para a instância R102, por exemplo, não foi constatado haver uma significativa diferente entre as soluções geradas pelos algoritmos. Para os problemas teste C104 e C204, percebeu-se que, em relação aos demais algoritmos, o HSCM+ILS demonstrou resultados com diferença significativa, em que, por meio dos gráfico de caixas, foi observado que as suas soluções, em relação a distância percorrida, foram melhores.

Ao se analisar os resultados obtidos para a instância RC104, foi indicada que o HSCM+ILS e o $P-ACO_{Elitist}$+ILS possuem uma diferença significativa em relação aos algoritmos ILS e MMAS+ILS. Desta forma, foi verificado, por meio da análise dos gráficos, que o ILS e o MMAS+ILS foram melhores que o HSCM+ILS e o $P-ACO_{Elitist}$+ILS. No caso da instância R205, foi visto que, com a ajuda do teste estatístico e da análise gráfica, que o $P-ACO_{Quality}$+ILS foi melhor do que os demais algoritmos.

Em relação aos resultados acerca da instância RC204, viu-se que o HSCM+ILS gerou resultados piores do que os algoritmos GRASP+ILS, $P-ACO_{Age}$+ILS, $P-ACO_{Quality}$+ILS e $P-ACO_{Elitist}$+ILS.

Com base nos estudos realizados, percebeu-se que muitas das técnicas desenvolvidas, aos serem usadas para auxiliar na resolução do PRAVJT, foram de grande importância na obtenção de resultados satisfatórios. Além disso, devido ao pequeno número de trabalhos existentes na literatura que tratam desse tipo de problema em específico, espera-se que o presente estudo possa contribuir para a realização de novas pesquisas acerca do PRAVJT e suas variantes.

7.1 Publicações Originárias desta Dissertação

Os trabalhos listados a seguir, originados das metodologias propostas para o Problema de Roteamento Aberto de Veículos, foram aceitos para apresentação em conferências nacionais:

1. COSTA,J.M., Souza, S.R., Souza, M.J.F., . *Uso Das Metaheurísticas GRASP e ILS para a Resolução do Problema de Roteamento Aberto de Veículos com Janelas Tempo*, **CILAMCE**, 2011, Ouro Preto.

2. Costa,J.M., Souza, S.R., Souza, M.J.F., . *Uso de Metaheurísticas para Solucionar o Problema de Roteamento Aberto de Veículos com Janelas de Tempo*, **XIX Congresso Brasileiro de Automática**, 2012, Campina Grande.

3. COSTA,J.M., Souza, S.R., Souza, M.J.F. . *Estudo da Aplicação de Metaheurísticas ao Problema de Roteamento Aberto de Veículos com Janelas de Tempo*, **XVI Congresso Latino-Iberoamericano de Investigación Operativa/XLIV Simpósio Brasileiro de Pesquisa Operacional**, 2012, Rio de Janeiro.

4. COSTA,J.M., Oliveira, M.X., Souza, S.R., Souza, M.J.F., . *Um Estudo da Solução do Problema de Roteamento Aberto de Veículos com Janela de Tempo utilizando Metaheurísticas*, **XV Simpósio de Pesquisa Operacional e Logística da Marinha**, 2012, Rio de Janeiro.

5. COSTA,J.M., Souza, S.R., Souza, M.J.F. . *Heurísticas GRASP e ILS aplicadas ao Problema de Roteamento Aberto de Veículos com Janelas de Tempo*, **XXXII Encontro Nacional de Engenharia de Produção**, 2012, Bento Gonçalves.

7.2 Trabalhos Futuros

Em relação ao PRAVJT e à continuidade da linha de pesquisa aqui empreendida, as sugestões para trabalhos futuros são:

- Implementar outras heurísticas construtivas como o GENIUS, por exemplo;
- Implementar novas estruturas de vizinhança;
- Testar o uso de um método de busca local no HSCM;
- Trabalhar com aplicações baseadas em problemas de roteamento aberto de veículos reais;
- Implementar outros métodos de refinamento;
- Implementar outros procedimentos de construção para os algoritmos baseados na metaheurística Colônia de Formigas;

Referências Bibliográficas

Aiex, R. M.; Resende, M. G. C. e Ribeiro, C. C. (2002). Probability distribution of solution time in grasp: An experimental investigation. *Jornal of Heuristics*, v. 8, p. 343–373.

Aksen, D.; Ozyurt, Z. e Aras, N. (2007). Open vehicle routing problem with driver nodes and time deadlines. *Journal of the Operational Research Society*, v. 59, p. 1223–1234.

Blum, C. (2004). The hyper-cube framework for ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, v. 34, n. 2, p. 1161–1172.

Blum, C.; Puchingerb, J.; Günter, R. e Roli, A. (2011). Hybrid metaheuristic in combinatorial optimization: A survey. *Applied Soft Computing*, v. 11, p. 4135–4151.

Brandão, J. (2004). A tabu search algorithm for the open vehicle routing problem. *European Journal of Operation Research*, v. 157, p. 552–564.

Costa, J. M.; Oliveira, M. X.; Souza, S. R. e Souza, M. J. F. (2012). Um estudo da solução do problema de roteamento aberto de veículos com janela de tempo utilizando metaheurísticas. *Anais do XV Simpósio de Pesquisa Operacional e Logística da Marinha*, Rio de Janeiro, RJ.

Cristofides, N.; Mingozzi, A. e Toth, P. (1979). The vehicle routing problem. Cristofides, N.; Mingozzi, A.; Toth, P. e Sandi, C., editors, *Combinatorial Optimization*, p. 313–338. Wiley, Chichester.

Deneubourg, J. L.; S. Aron, S. Goss e Pasteels, J.-M. (1990). The selforganizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, v. 3, p. 159.

Derigs, U. e Reuter, K. (2009). A simple and efficient tabu search heuristic for solving the open vehicle routing problem. *Journal of the Operational Research Society*, v. 60, p. 1658–1669.

Dorigo, M.; Birattari, M. e Stützle, T. (2006)a. Ant colony optimization. *IEEE Computational Intelligence Magazine*, v. 1(4), p. 28–39, 2006.

Dorigo, M.; Birattari, M. e Stützle, Thomas. (2006)b. Ant colony optimization: Artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, v. 1(4), p. 28–39.

Dorigo, M.; Maniezzo, V. e Colorni, A. (1991). Positive feedback as a search strategy. *Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy.*

Dorigo, M.; Maniezzo, V. e Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, v. 26(1), p. 29–41.

Dorigo, M. e Stützle, T. (2003). The ant colony optimization metaheuristic: Algorithms, applications, and advances. Glover, In F. e Kochenberger, G. A., editors, *Handbook of Metaheuristics*, Capítulo 9, p. 251–285. Kluwer Academic Publishers.

Dorigo, M. e Stützle, T. (2004). *Ant Colony Optimization.* MIT Press.

Dueck, G. e Scheuer, T. (1990). Threshold accepting: a general purpose algorithm appearing superior to simulated annealing. *Journal of Computacional Physics*, v. 90, p. 161–175.

Fenghua, Duan. e Xiaonian, He. (2010). Iovrpstw and its improved genetic algorithm. *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on*, p. 933–936.

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

Field, Andy. P. (2009). *Descobrindo a estatística usando o SPSS.* Artmed, 2 edição.

Fisher, M. (1994). Optimal solution of vehicle routing problems minimum using k-trees. *Operations Research*, v. 42, p. 626–642.

Fu, Z.; Eglese, R. e Li, LYO. (2005). A new tabu search heuristic for the open vehicle routing problem. *Journal of the Operational Research Society*, v. 56, p. 267–274.

Gendreau, M.; Hertz, A. e Laporte, G. (1992). New insertion and post-optimization procedures for the traveling salesman problem. *Operations Research*, v. 40, p. 1086–1094.

Glover, F. e Laguna, M. (1997). *Tabu Search.* Kluwer Academic Publishers, Norwell, MA, USA.

Goldberg, D. E. (1989). *Genetics algorithms in search, optimization and machine learning.* Addison Wesley Longman, Reading, Massachusetts, USA.

Goss, S.; Aron, S.; Deneubourg, J. L. e Pasteels, J. M. (1989). Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, v. 76, p. 579–581.

Guiyun, Li. (2009). An improved ant colony algorithm for open vehicle routing problem with time windows. *International Conference on Information Management, Innovation Management and Industrial Engineering*, v. 2, p. 616–619.

Guntsch, M. *Ant Algorithms in Stochastic and Multi-Criteria Environments.* PhD thesis, Universität Fridericiana zu Kariscruhe, (2004).

Hertz, A. e Widmer, M. (2003). Guidelines for the use of meta-heuristics in combinatorial optimization. *European Journal of Operation Research Society*, v. 151, p. 247–252.

Homberger, J. e Gehring, H. (1999). Two-evolunary meta heuristics for the vehicle routing problem with time windows. *INFOR*, v. 37, p. 297–318.

Ioannou, G.; Kritikos, MN. e Prastacos, GP. (2001). A greedy look-ahead heuristic for the vehicle routing problem with time windows. *Journal of the Operational Research Society*, v. 52, p. 523–537.

Kirkpatrick, S.; Gelatt, C. D. e Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, v. 220, p. 671–680.

Letchford, A.N.; Lysgaard, J. e Eglese, R.W. (2006). A branch-and-cut algorithm for the capacitated open vehicle routing problem. *Journal of the Operational Research Society*, v. 58, p. 1642–1651.

Li, F.; Golden, B. e Wasil, E. (2007). The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Computers and Operations Research*, v. 34, p. 2918–2930.

Li, X-Y e Tian, P. (2006). Ant colony system for the open vehicle routing problem. Dorigo, M. et al., editor, *ANTS 2006*, p. 356–363. Proceedings for 5th International Workshop ont Ant Colony Optimization and Swarm Intelligence; Springer: Berlin.

Li, X.Y.; Tian, P. e Leung, S.C.H. (2009). An ant colony optimization metaheuristic hybridized with tabu search for open vehicle routing problems. *Journal of the Operational Research Society*, v. 60, p. 1012–1025.

Lourenço, H. R.; Martin, O. C. e Stützle, Thomas. (2003). Iterated local search. Glover, G.F. e Kochenberger, editor, *Handbook of Metaheuristics*, p. 321–353. Kluwer Academic Publishers, Boston.

MirHassani, S.A. e Abolghasemi, N. (2011). A particle swarm optimization algorithm for open vehicle routing problem. *Expert Systems with Applications*, v. 38, p. 11547–11551.

Mladenovic, N. e Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, v. 24, p. 1097–1100.

Oliveira, S.; Hussin, M. S.; Stützle, T.; Roli, A. e Dorigo, M. (2011). A detailed analysis of the population-based ant colony optimization algorithm for the TSP and the QAP. *Technical Report TR/IRIDIA/2011-006, IRIDIA, ULB, 2011.*

Repoussis, P.P.; Tarantilis, C.D. e Ioannou, G. (2006). The open vehicle routing problem with time windows. *Journal of the Operational Research Society*, v. 58, p. 1–13.

Repoussis, P.P.; Tarantilis, C.D. e Ioannou, G. (2009). An evolutionary algorithm for the open vehicle routing problem with time windows. *Bio-inspired algorithms for the vehicle routing problems*, v. 161, p. 55–75.

Salari, Majid.; Toth, Paolo. e Tramontani, Andrea. (2010). An ilp improvement procedure for the open vehicle routing problem. *Computers & Operations Research*, v. 37, p. 2106–2120.

Sarikilis, D. e Powell, S. (2000). A heuristic method for the open vehicle routing problem. *Journal of the Operational Research Society*, v. 51, p. 564573–.

Schrage, L. (1981). Formulation and structure of more complex/realistic routing and scheduling problems. *Networks*, v. 11, p. 229–232.

Solomon, MM. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, v. 35, p. 2544–265.

Souza, M. J. F. (2009). *Inteligência Computacional para Otimização. Notas de aula*.

Stützle, T. *Local Search Algorithms for Combinatorial Problems - Analysis, Algorithms and New Applications*. PhD thesis, Infix, Sankt Augustin, Technischen Universität Darmstadt, Computer Science Department, (1998).

Stützle, Thomas. e Hoos, H. H. (1996). Improving the ant system: A detailed report on the max-min ant system. *FG Intellektik, FB Informatik, TU Darmstadt, Germany, Tech. Rep.*

Stützle, Thomas. e Hoos, H. H. (1997). The max-min ant system and local search for the traveling salesman problem. *IEEE International Conference on Evolutionary Computation, T. Bäck et al., Eds. IEEE Press, Piscataway, NJ*, p. 309–314.

Stützle, Thomas. e Hoos, H. H. (1999). Max-min ant system and local search for combinatorial optimization problems. S. Voss, I.H. Osman C. RoucairolS. Martello, editor, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, p. 313–329. Kluwer Academic Publishers.

Stützle, Thomas. e Hoos, H. H. (2000). Max-min ant system. *Future Generation Computer Systems*, v. 16, p. 889–914.

Tan, K.; Lee, L. H.; Zhu, Q. L. e Ou, Q. (2001). Heuristic methods for vehicle routing problems with time windows. *Artificial Intelligence in Engineering*, v. 15, p. 285–295.

Tarantilis, C.; Diakoulaki, D. e Kiranoudis, C. (2004)a. Combination of geographical information system and efficient routing algorithms for real life distribution operations. *European Journal of Operation Research*, v. 152, p. 437–453.

Tarantilis, C.; Kiranoudis, C. e Prastacos, G. (2004)b. A threshold accepting approach to the open vehicle routing problem. *RAIRO - Operational Research*, v. 38, p. 345–360.

Tarantilis, CD.; Ioannou, G; Kiranoudis, CT e Prastacos, GP. (2005). Solving the open vehicle routeing problem via a single parameter metaheuristic algorithm. *Journal of the Operational Research Society*, v. 56, p. 588–596.

Toth, Paolo. e Vigo, Daniele. (2002). *The Vehicle Routing Problem*. Society for Industrial e Applied Mathematics.

Triola, Mario. F. (2008). *Introdução à estatística*. Rio de Janeiro: LTC - Livros Técnicos e Científicos, 10 edição.

Yu, S.; Ding, C. e Zhu, K. (2011). A hybrid ga -ts algorithm for open vehicle routing optimization of coal mines material. *Expert Systems with Applications*, v. 39, p. 10568–10573.

Ziviani, N. (2004). *Projeto de Algoritmos: Com implementações em Pascal e C*. Thomson Learning, 2 edição.