

ROBERT CRISTIAN ABREU

MÉTODOS DE OTIMIZAÇÃO PARA O
PROBLEMA DE ROTEAMENTO DE VEÍCULOS
PERIÓDICO COM FROTA HETEROGÊNEA

Dissertação apresentada a Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

Viçosa
Minas Gerais-Brasil
2016

**Ficha catalográfica preparada pela Biblioteca Central da Universidade
Federal de Viçosa - Câmpus Viçosa**

T

A162m
2016

Abreu, Robert Cristian, 1989-

Métodos de otimização para o problema de roteamento de
veículos periódico com frota heterogênea / Robert Cristian
Abreu. – Viçosa, MG, 2016.
viii, 59f : il. (algumas color.) ; 29 cm.

Inclui apêndice.

Orientador: José Elias Claudio Arroyo.

Dissertação (mestrado) - Universidade Federal de Viçosa.

Referências bibliográficas: f.55-58.

1. Otimização combinatória. 2. Heurística. 3. Algoritmos.
4. Pesquisa operacional. 5. Veículos - Roteamento.

I. Universidade Federal de Viçosa. Departamento de Informática.
Programa de Pós Graduação em Ciência da Computação.

II. Título.

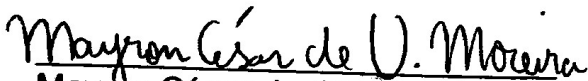
CDD 22 ed. 519.64


ROBERT CRISTIAN ABREU

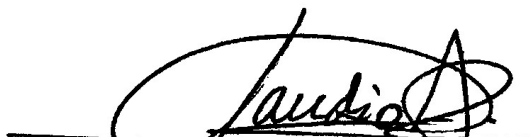
**MÉTODOS DE OTIMIZAÇÃO PARA O PROBLEMA DE
ROTEAMENTO DE VEÍCULOS PERIÓDICO COM FROTA
HETEROGÊNEA**

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 06 de julho de 2016.


Mayron César de Oliveira Moreira


André Gustavo dos Santos


José Elias Claudio Arroyo
Orientador

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus por me dar a força e saúde necessária para terminar esta jornada. Agradeço por Ele ter colocado em meu caminho as pessoas certas e os meios necessários para conclusão deste trabalho.

Aos meus pais, Heberth, in memoriam, e Silvia, pelo apoio incondicional, pelo amor e por terem me dado a oportunidade de chegar onde eu cheguei. Se estou aqui certamente foi pelo incentivo e educação que me deram durante toda a minha vida.

A minha noiva, amiga, companheira e futura esposa Leila pela persistência, pela paciência, pelos puxões de orelha e principalmente pela companhia. Obrigado por estar junto comigo, suportando os percalços e desventuras que foram aparecendo.

Aos meus amigos, Rafael, sempre pronto a me ajudar, sem ele não sei como terminaria este processo, e a Juliana, que me incentivou a começar essa luta, esteve ao meu lado em todos os momentos, sem ela eu não estaria aqui agora. Obrigado a ambos, meus irmãos de coração, por tudo que fizeram por mim.

Ao meu orientador José Elias Arroyo pela paciência e determinação. Obrigado por me ajudar durante o mestrado e mesmo antes dele na iniciação científica e trabalho de conclusão de curso. Obrigado por fazer esta caminhada comigo.

Agradeço a todos os professores do Departamento de Informática da UFV que contribuíram em minha formação e tanto me ensinaram, seja em aspectos técnicos ou experiências de vida.

Sumário

Lista de Figuras	v
Lista de Tabelas	vii
Resumo	viii
Abstract	ix
1 Introdução	1
1.1 O problema e sua importância	2
1.2 Objetivo	3
1.3 Organização do trabalho	3
2 Problema de Roteamento de Veículos Periódico	4
2.1 Trabalhos relacionados	4
2.2 Definição do problema	8
3 Métodos propostos	12
3.1 Métodos construtivos	12
3.1.1 Construtivo Cordeau	13
3.1.2 Construtivo Mortati	13
3.1.3 Construtivo Aleatório	14
3.2 Penalidade	15
3.3 Buscas locais	15
3.3.1 Movimentos intra-rota	16
3.3.2 Movimentos inter-rota	17
3.3.3 Otimiza agendas	20
3.4 Iterated Local search (ILS)	22
3.4.1 ILS-RVND	23

3.4.2	Perturbações	25
3.5	Particle Swarm Optimization (PSO)	25
3.5.1	PSO Discreto	27
3.5.2	Mutação	29
3.6	Proximity Search (PS)	30
3.6.1	Implementação	31
4	Geração de instâncias e calibração dos algoritmos	33
4.1	Instâncias geradas	34
4.2	Calibração do ILS	35
4.3	Calibração do PSO	38
5	Testes computacionais	43
5.1	Ambiente de testes	43
5.2	Avaliação das heurísticas nas instâncias geradas	44
5.2.1	Resultados para instâncias de pequeno porte	44
5.2.2	Resultados para instâncias de grande porte: Comparação entre Proximity Search e CPLEX	45
5.2.3	Resultados para instâncias de grande porte: Comparação entre todos os métodos	48
5.3	Avaliação das heurísticas em instâncias da literatura	50
6	Conclusões	53
6.1	Trabalhos futuros	53
	Referências Bibliográficas	55
A	Publicação	59

Lista de Figuras

3.1	Configuração inicial da solução.	16
3.2	Exemplo de movimento Swap(1).	17
3.3	Exemplo de movimento 2-Opt.	17
3.4	Exemplo de movimento Or-Opt(1).	18
3.5	Exemplo de movimento Swap(1,1).	18
3.6	Exemplo de movimento Shift(1).	19
3.7	Exemplo de movimento Cross.	19
3.8	Exemplo de movimento Radial.	20
3.9	Cálculo do vetor velocidade da partícula i . Onde p_i é a melhor posição já encontrada pela partícula i , g é a posição da melhor partícula da nuvem e x_i é a posição atual da partícula i . Adaptado: de Lacerda [2007].	27
3.10	Exemplo de distância de Hamming entre x e x'	31
4.1	Valor médio da F.O para cada número de repetições do ILS.	36
4.2	GAP entre o valor inicial dos métodos construtivos para o ILS.	36
4.3	GAP entre o valor final dos métodos construtivos para o ILS.	37
4.4	GAP entre o valor da função objetivo para cada tipo de perturbação.	37
4.5	GAP entre o valor inicial dos métodos construtivos para o PSO.	39
4.6	GAP entre o valor final dos métodos construtivos para o PSO.	39
4.7	Valor médio da F.O para cada combinação de W , c_1 e c_2	40
4.8	Imagem ampliada da primeira metade do gráfico da Figura 4.7 para melhor visualização.	40
4.9	Imagem ampliada da segunda metade do gráfico da Figura 4.7 para melhor visualização.	41
4.10	Valor médio da F.O para cada número de repetições do PSO.	41
4.11	Valor médio da F.O para cada número de partículas do PSO.	42

5.1	Gráfico de Médias e intervalos HSD de Tukey com nível de confiança de 95% para a comparação dos resultados encontrados pelos três métodos. . .	45
5.2	Evolução dos métodos no tempo.	48
5.3	Gráfico de Médias e intervalos HSD de Tukey com nível de confiança de 95% para a comparação dos resultados encontrados pelos três métodos. . .	49

Lista de Tabelas

5.1	Médias dos RPDs (por grupo de instâncias de 10 a 30 clientes) dos algoritmos comparados para o melhor resultado encontrado.	44
5.2	Média de RPI por grupo de instâncias.	47
5.3	Média geométrica do primal integral por intervalo de tempo.	48
5.4	Médias dos RPDs (por grupo de instâncias de 50 a 100 clientes) dos algoritmos comparados para o melhor resultado encontrado.	49
5.5	Comparação do resultado da função objetivo entre as heurísticas.	51
5.6	Comparação do resultado da função objetivo entre as heurísticas para as instâncias com limite de distância.	52

Resumo

ABREU, Robert Cristian, M.Sc., Universidade Federal de Viçosa, julho de 2016. **Métodos de otimização para o Problema de Roteamento de Veículos Periódico com frota heterogênea.** Orientador: José Elias Cláudio Arroyo.

Definição

O Problema de Roteamento de Veículos (PRV) é um problema clássico de Otimização Combinatória bastante estudado na literatura devido a sua importância prática. O PRV Periódico (PRVP), abordado neste trabalho, é uma variante do PRV no qual um conjunto de **clientes devem ser visitados uma ou mais vezes** para atender suas demandas durante um horizonte de tempo composto de vários dias. **Os dias de visita/atendimento** não são fixados a priori. Uma lista de dias possíveis (agenda de visitas) é associada a cada cliente. O objetivo é determinar os dias de visita de cada cliente e as rotas dos veículos para cada dia do horizonte de tal maneira que a distância total de percurso dos veículos e os custos associados com utilização dos mesmos sejam **minimizados**.

O PRVP é um problema que pertence à classe NP-difícil. Neste trabalho, para resolvê-lo, são desenvolvidos três métodos de otimização: **Proximity Search (PS)**, **Iterated Local Search (ILS)** e **Particle Swarm Optimization (PSO)**. PS é um método genérico que faz uso do modelo de **Programação Inteira do problema para melhorar iterativamente uma solução inicial**. Em vez de modificar as restrições do modelo com o objetivo de reduzir o espaço de busca, o PS modifica a função objetivo do modelo para tornar a busca mais fácil. Os métodos ILS e PSO são **meta-heurísticas de busca em vizinhança e populacional/evolutiva**, respectivamente.

Os desempenhos dos métodos propostos são analisados em instâncias de pequeno e grande porte geradas neste trabalho, e também **em instâncias disponíveis na literatura**. O desempenho do PS é comparado com o solver **CPLEX**, que resolve o modelo original do problema. As meta-heurísticas desenvolvidas são comparadas entre si e também são comparadas com algumas heurísticas da literatura. Os experimentos computacionais mostram que os métodos propostos são eficientes, competitivos e rápidos.

Abstract

ABREU, Robert Cristian, M.Sc., Universidade Federal de Viçosa, July of 2016. **Optimization methods for the Periodic Vehicle Routing Problem with heterogeneous fleet.** Adviser: José Elias Cláudio Arroyo.

The Vehicle Routing Problem (VRP) is a classic problem of Combinatorial Optimization extensively studied in the literature because of its practical importance. The Periodic VRP (PVRP), discussed in this work, is a variant of VRP in which a group of customers should be visited one or more times to meet their demands over a time horizon composed of several days. The days to visit a customer are not initially fixed. A possible list of days (visits schedule) is associated with each customer. The objective problem is to determine the set of days to visit each customer and determine the routes of the vehicles for each day of the planning horizon such that the total distance of the vehicle route and costs associated with use of them are minimized.

The PVRP is a problem that belongs to the NP-hard class. In this work, three optimization methods are developed to solve the problem: Proximity Search (PS), Iterated Local Search (ILS) and Particle Swarm Optimization (PSO). PS is a generic method that makes use of the Integer Programming Problem model to iteratively improve an initial solution. Instead of modifying the model restrictions intended to reduce the search space, the PS changes the objective function to make the search easier. The ILS and PSO methods are meta-heuristic search in the neighborhood and population / evolutionary, respectively.

The performances of the proposed methods are analyzed using small and large instances generated in this work, and also for instances available in the literature. The performance of PS is compared with CPLEX solver, which solves the original problem formulation. The developed meta-heuristics are compared to each other and are also compared with some heuristics from the literature. The computational experiments show that the proposed methods are efficient, competitive and fast.

Capítulo 1

Introdução

O Problema de Roteamento de Veículos (PRV), conhecido na literatura inglesa como *Vehicle Routing Problem* (VRP), é um problema clássico da área de Pesquisa Operacional. Este problema foi apresentado pela primeira vez por Dantzig & Ramser [1959], que estudaram distribuição de gasolina em estações de venda de combustíveis, e desde então tem sido bastante estudado graças a sua importância nas áreas logística e economia. O PRV clássico consiste em encontrar rotas para os veículos disponíveis de modo que todos os pontos, que podem ser clientes ou fornecedores, sejam plenamente atendidos e a distância total percorrida seja a menor possível, minimizando assim os custos de transporte. Todos os veículos iniciam e terminam suas rotas no depósito.

O grande interesse pelo estudo de PRV ocorre, segundo Baldacci & Mingozzi [2009], devido a sua grande aplicação em problemas reais de entrega e coleta, distribuição e logística. Além disto, Toth & Vigo [2002] afirmam que os sistemas de distribuição representam, dentro do custo final dos bens distribuídos, valores que vão de 10 a 20%, e reduzir estes valores representaria um aumento na receita líquida. Ademais, os métodos de resolução deste problema, como por exemplo, aplicação de heurísticas ou a algum método exato (*Branch and Bound*, *Branch and Cut*, entre outros), são muito eficientes no sentido operacional, ou seja, essas técnicas tem sido implementadas na vida real em diferentes áreas de transporte e se mostrado muito efetivas.

Visando atender a um mercado cada vez mais exigente e competitivo, diversas variações do PRV foram criadas. Cada variação contempla uma determinada característica da situação encontrada no mundo real. Dentre as diversas variações existentes, podemos citar algumas:

- PRV Capacitado ou Clássico: nesta variação do PRV, todos os clientes e as demandas são conhecidas, e a priori, não podem ser divididas. Os veículos são

homogêneos, ou seja, possuem as mesmas características, e existe apenas um depósito. As restrições que são impostas estão relacionadas somente a capacidade dos veículos.

- PRV Coleta e Entrega: esta variação é apenas uma generalização do PRV Capacitado, no qual cada cliente possui uma oferta a ser coletada e uma demanda a ser atendida. Durante toda a rota, a capacidade máxima dos veículos não pode ser ultrapassada.
- PRV Janelas de Tempo (PRVJT): esta variação possui as mesmas restrições do PRV Capacitado, porém é adicionado a cada cliente uma janela de tempo. O veículo deve iniciar o serviço dentro deste intervalo de tempo e permanecer até que este termine. Se o veículo chegar fora da sua janela de tempo, este deve esperar seu início.
- PRV com Múltiplos Depósitos: esta variação, diferente do Problema Capacitado, define a rota dos clientes e define de qual depósito sairá as entregas. Os veículos podem reabastecer em outros depósitos, porém devem sempre retornar ao final de suas rotas ao depósito de origem.

A variação estudada neste trabalho é o Problema de Roteamento de Veículos Periódico (PRVP). No PRVP, além das restrições comuns existentes no caso clássico, os clientes devem ser atendidos durante um determinado período de tempo e não apenas uma única vez. Este período de tempo, geralmente definido em dias, é chamado de horizonte de tempo, ou horizonte de planejamento. Cada cliente possui um conjunto de agendas de visitas, ou combinações de visitas, que especifica em quais dias do horizonte ele pode ser atendido. O objetivo é planejar a rota de todos os veículos dentro deste horizonte de modo que o custo total ao final do período seja o mínimo possível e uma agenda de cada cliente seja satisfeita.

1.1 O problema e sua importância

O PRVP tem grande aplicação prática no mundo real. Vários sistemas podem ser modelados utilizando o modelo periódico, como por exemplo coleta de lixo, Baptista et al. [2002], Angelelli & Speranza [2002], Higino et al. [2012], Bianchi-Aguiar et al. [2012], o recolhimento de leite por laticínios, do Valle & Nogueira [2010], da Silva et al. [2011], entrega de roupas em hospitais, Banerjee-Brodeur et al. [1998], distribuição de produtos para supermercados, Carter et al. [1996], Pacheco et al. [2012] e até mesmo distribuição de sangue, Hemmelmayr et al. [2009a].

Este problema pertence à área de Otimização, dentro de Pesquisa Operacional. Ele é classificado como NP-Difícil, assim como todas as variações dos problemas de roteamento de veículos Gary & Johnson [1979]. Problemas deste tipo apresentam grande complexidade, no sentido de não existirem algoritmos eficientes e completos que forneçam a solução ótima em tempo adequado. Dessa forma, muitas vezes, sua resolução recorre a métodos heurísticos para encontrar uma resposta próxima a ótima em tempo aceitável. Segundo Glover & Laguna [1997], as meta-heurísticas tem sido usadas com grande sucesso em uma vasta gama de problemas dentro da área de otimização, incluindo problemas de roteamento de veículos (Laporte [2009]).

Para aproximar o modelo do problema de roteamento periódico às situações reais, este trabalho considera que a frota disponível para atender os clientes é formada por veículos com características diferentes, chamada frota heterogênea. Nesse problema, além das decisões sobre roteamento, é necessário que se escolham os veículos mais adequados de modo que se obtenha o menor custo envolvido. Cada veículo pode possuir uma capacidade máxima diferente.

1.2 Objetivo

Este trabalho tem como objetivo geral propor métodos para a resolução do PRVP, que equilibrem a qualidade das soluções obtidas e o tempo necessário para obtê-las. Para análise da qualidade, os métodos desenvolvidos serão testados, calibrados, e seus resultados comparados entre si e também a métodos já existentes na literatura. Para estes testes, serão utilizadas tanto instâncias conhecidas na literatura como instâncias próprias, visto que o número total de instâncias disponíveis para este problema é muito limitado.

1.3 Organização do trabalho

O trabalho está assim dividido: no Capítulo 2, são apresentados os trabalhos relacionados e o atual estado da arte. No Capítulo 3 é apresentada a definição formal do problema. O Capítulo 4 descreve os métodos propostos para resolução do PRVP. No Capítulo 5, as novas instâncias criadas e suas características são apresentados. No Capítulo 6, são apresentados os resultados computacionais obtidos. Por fim, o Capítulo 7, encerra esse trabalho com as considerações finais e as sugestões de futuros trabalhos.

Capítulo 2

Problema de Roteamento de Veículos Periódico

O primeiro trabalho relacionado ao PRVP, que também o definiu formalmente é descrito no artigo Beltrami & Bodin [1974], em que os autores estudam a coleta de lixo em Nova Iorque. O problema da coleta de lixo também foi abordado por Russell & Igo [1979]. Ambos os trabalhos propõem heurísticas baseadas no algoritmo de Clarke & Wright [1964] e movimentos de otimização intra-rota. Depois disso, diversas novas abordagens foram testadas para resolução do PRVP. Na próxima seção será detalhado a evolução dos trabalhos para este problema, destacando as contribuições mais relevantes e em seguida, a definição matemática formal do problema será apresentada.

2.1 Trabalhos relacionados

Uma heurística de duas fases foi proposta por Christofides & Beasley [1984]. Na primeira fase, a heurística tenta encontrar uma combinação de agendas de visita e as rotas geradas a partir dessas agendas são otimizadas. Na segunda fase, alguns clientes tem suas agendas alteradas e consequentemente suas rotas, passando pelo processo de otimização novamente. A troca de agendas é computacionalmente cara, já que cada troca envolve resolver um novo PRV. Dessa forma, os autores separam o problema em dois outros problemas relaxados. No primeiro, busca-se minimizar a soma das distâncias dos clientes em determinados conjuntos. No segundo, há a resolução de problemas do caixeiro viajante para cada centro definido.

Ainda em 1984, Tan & Beasley [1984] propõem uma heurística de três fases, separando clientes em grupos, resolvendo um problema de programação linear inteira para definir suas combinações de visita e, por fim, resolvendo um PRV para cada

dia utilizando o algoritmo de Fisher & Jaikumar [1981]. Alguns anos depois, Russell & Gribbin [1991] propõem uma heurística de quatro fases, onde a primeira e a segunda são basicamente iguais às de Tan & Beasley [1984] e a terceira equivale ao procedimento de troca de combinações de Christofides & Beasley [1984], com a mudança de re-otimizar as rotas de cada dia. Por fim, a quarta fase é um problema de programação binária que altera as agendas de visitas de cada cliente para maximizar os ganhos de distância.

Chao et al. [1995] propõem uma heurística que divide o problema em uma fase de determinação de agendas de visita e, em seguida, montagem das rotas. Na fase de inserção de clientes nas rotas, a agenda de cada um deles pode mudar ou não. É ainda utilizado um processo de otimização das rotas removendo o cliente e reinserindo-o em todas as posições possíveis, além da utilização do método 2-opt. Esta foi a heurística que apresentou os melhores resultados até então para as 13 instâncias que já haviam sido utilizadas nos outros trabalhos, além de inserir 19 novas instâncias.

O começo da utilização de meta-heurísticas para o PRVP ocorreu em 1997 quando Cordeau et al. [1997] utilizaram uma busca tabu para resolução das instâncias conhecidas. Os autores ordenam os clientes em relação ao ângulo formado entre eles, o depósito, e um raio arbitrário. As agendas de visita de cada cliente são escolhidas aleatoriamente e as rotas são construídas, podendo haver rotas inviáveis no princípio. Um mecanismo de penalidade é aplicado sobre o valor da função objetivo. Com a solução inicial criada, inicia-se o processo da Busca Tabu. Durante este processo, os clientes são removidos de suas rotas e inseridos em outras. A inserção destes clientes na sua rota original é proibida (tabu) por um determinado número de iterações. Os autores executaram esta heurística para as 32 instâncias existentes e obtiveram melhor resultado em 24 delas e resultado igual ao de Chao et al. [1995] em 6. Além disso, 10 novas instâncias foram criadas para o problema considerando agora um limite máximo que pode ser percorrido por cada veículo, critério que não era considerado nas instâncias anteriores.

A segunda meta-heurística utilizada é um algoritmo genético proposto por Drummond et al. [2001]. Neste algoritmo, a solução é representada por uma string onde cada elemento é o número correspondente à agenda de visita escolhida pelo cliente daquela posição. Os operadores de *crossover* e mutação são clássicos, porém a população é dividida em subgrupos que evoluem separadamente. A iteração desses subgrupos acontece quando a taxa de renovação dos elementos de um grupo cai para menos de 5%. Neste cenário, os outros grupos enviam seus melhores indivíduos para este grupo. Os resultados desta heurística são semelhantes aos apresentados por Cordeau et al. [1997].

Ainda em 2001, Cordeau et al. [2001] propõem uma nova busca tabu, desta vez para PRVP com janelas de tempo e múltiplos depósitos. Neste novo trabalho, os au-

tores utilizaram praticamente os mesmos mecanismos de sua primeira heurística alterando, porém, a forma como os clientes são inseridos nas rotas. Adaptando um pouco o PRVP padrão, Angelelli & Speranza [2002] adicionam um conjunto de facilidades intermediárias para carga e descarga. Com essa modificação, considera-se que cada veículo só retorne ao depósito após os clientes associados a uma determinada facilidade terem sido atendidos, podendo ou não passar por facilidades intermediárias. O método utilizado para resolução deste problema foi baseado na busca tabu de Cordeau et al. [2001].

Alegre et al. [2007] propõem um algoritmo *scatter search* para resolução de um caso particular do PRVP. O algoritmo primeiramente constrói algumas soluções com um método de diversificação e, em seguida, elas são submetidas a um processo de busca local. Um conjunto de soluções de referência é montado a partir das soluções melhoradas pela busca local. Este processo continua e o conjunto de referência só aceita novas soluções se estas possuírem valor da função objetivo menor que a pior solução presente no conjunto de referência. Além disso, um método de combinação é aplicado às soluções gerando um novo conjunto de soluções que são posteriormente melhoradas pelo método de busca local. Este método foi testado para as 32 instâncias comuns do PRVP e conseguiu um resultado melhor que os métodos existentes em 17 delas, e igual em 4.

Hemmelmayr et al. [2009b] propõem uma heurística de busca com vizinhança variável, ou como é comumente conhecida, *Variable Neighborhood Search* (VNS) para o PRVP e para o problema do caixeiro viajante periódico, que corresponde ao PRVP comum, mas com apenas um veículo. Primeiro, uma solução inicial é gerada pelo método construtivo, muito semelhante ao método utilizado por Cordeau et al. [2001]. Após isso, o algoritmo inicia suas iterações até que o critério de parada seja atingido. Primeiro a solução passa pelo processo de perturbação onde uma das 15 estruturas de vizinhança é aplicada, em seguida a solução passa pelo processo de busca local. A busca local é aplicada apenas nas rotas modificadas pela perturbação e utiliza o movimento de otimização de rotas 3-opt sem inversão da sequência de clientes, também conhecido como 3-opt*. O critério de aceitação não avalia apenas o valor da função objetivo, pois isto poderia levar o algoritmo a um estado em que ele ficaria preso em um ótimo local. Dessa forma, os autores utilizam o conceito de temperatura da heurística *Simulated Annealing* (SA) para aceitar também soluções inferiores eventualmente. O VNS foi aplicado nas 32 instâncias conhecidas da literatura e apresentou resultado melhor em 8 instâncias e igual em 7 delas.

O PRVP com janelas de tempo foi abordado por Pirkwieser & Raidl [2010]. Neste trabalho, os autores propõem dois métodos para resolução do problema, um

VNS é um algoritmo evolutivo, ambos utilizando uma abordagem exata por geração de colunas para determinar algumas características da solução. De acordo com os testes dos autores, esta abordagem combinada das heurísticas com o método exato tem resultados significativamente melhores do que as heurísticas por si só. Os testes foram feitos em instâncias adaptadas pelos autores.

Vidal et al. [2012] propõem um algoritmo genético híbrido para três classes do problema de roteamento de veículos, incluindo o PRVP. O algoritmo genético inicia com um conjunto de indivíduos separados em dois grupos, viáveis e inviáveis. A cada iteração, os novos indivíduos, criados pela etapa de cruzamento envolvendo toda a população, passam por um processo de busca local, um processo de reparação e depois são incluídos no grupo correspondente. Quando a população ultrapassa o tamanho máximo permitido, um critério de sobrevivência é aplicado. Os processos de busca local e reparação incluem diversas análises e melhorias nas rotas com os movimentos clássicos do PRV como 2-opt e *swap*. A escolha dos indivíduos, tanto na etapa de sobrevivência quanto no cruzamento, segue um método guia que busca escolher sempre os mais indicados para cada caso, considerando também penalidades. Todo este processo foi chamado pelos autores de Heurística de Controle Adaptativo de Diversidade. Os testes executados sobre as instâncias conhecidas mostraram que o HGSADC (*The Hybrid Genetic Search with Adaptive Diversity Control Metaheuristic*) é extremamente robusto e eficiente, conseguindo resultado melhor em 20 de 42 instâncias e igual em 5, se tornando o método que apresentou os melhores resultados até o presente momento.

Pacheco et al. [2012] apresentam um estudo de caso real para uma empresa de panificação localizada no norte da Espanha. Os autores modelaram o sistema de distribuição da empresa como um PRVP e propuseram uma heurística de duas fases para resolução das instâncias criadas com base nos dados reais. Na primeira fase, é utilizada a meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedure*) para geração de um conjunto de soluções iniciais que são, na segunda fase, melhoradas por um processo de *Path-Relinking*. Para fazer com que as soluções dos algoritmos sejam o mais próximo possível da realidade, novas restrições foram adicionadas ao PRVP clássico. Os testes computacionais foram feitos utilizando as instâncias do autor e alguns dos algoritmos conhecidos da literatura. Nestes testes a heurística de Pacheco et al. [2012] foi superior em 16 das 20 instâncias geradas.

Higino et al. [2012] também fazem um estudo de caso propondo duas meta-heurísticas, um *Simulated Annealing* (SA) e um *Variable Neighborhood Search* (VNS) para serem aplicadas a coleta de lixo. Para o SA, foram usadas quatro estruturas de vizinhança para geração da solução inicial e o VNS possui 12 estruturas separadas em três grupos. Os testes foram executados na instância de Bianchi-Aguiar et al. [2012] que

representam o problema de coleta em Lima, Portugal, e obtiveram resultados melhores.

Cacchiani et al. [2014] propõem uma heurística baseada em cobertura de conjuntos, *Set Covering* (SC), programação linear inteira com relaxação e ILS. O problema relaxado utiliza geração de colunas e cada subproblema é resolvido pelo ILS. Este ILS utiliza duas estruturas de vizinhança na busca local e cinco estruturas na perturbação. O modelo é então executado até que um determinado critério de parada seja atingido, não necessariamente encontrando a solução ótima. Esta combinação de heurística e PLI foi testada nas instâncias conhecidas e nas instâncias introduzidas por Pacheco et al. [2012]. Este método conseguiu melhorar 4 das 32 instâncias e obteve um resultado igual em 13 perdendo em desempenho apenas para Vidal et al. [2012].

Recentemente Chen et al. [2016] propõem o uso de hyper-heurísticas para resolução do PRVP. Uma hyper-heurística é um método que busca automatizar a utilização de outras heurísticas, chamadas de heurísticas de baixo nível, do original *Low Level Heuristics* (LLH), muitas vezes utilizando um processo de aprendizado de máquinas. Neste trabalho, os autores utilizam um método construtivo, um conjunto de perturbações e diversos operadores de busca local para construir o conjunto de buscas que será utilizado no LLH. Uma comparação entre uma utilização aleatória das buscas, ou uma utilização baseada em procedimentos de aprendizagem foi feita. Nesta comparação, o método com aprendizagem se mostrou superior. Este algoritmo também foi comparado as instâncias conhecidas e obteve resultados competitivos ficando apenas 1% pior que os melhores valores conhecidos na literatura.

2.2 Definição do problema

O PRVP é uma generalização do PRV comum, no qual é adicionado um horizonte de planejamento. Todos os clientes possuem uma agenda de visita que especifica em quais dias deste horizonte eles devem ser visitados. O objetivo é atender a todos os clientes, nos dias requisitados, de modo que a distância percorrida pelos veículos seja a mínima possível ao final do horizonte. Cada cliente possui uma ou mais agendas de visita, sendo que exatamente uma deve ser atendida. Neste trabalho, abordamos o problema com um único depósito, assim, todos os veículos saem da mesma origem e ao final de suas rotas devem retornar a este ponto. A capacidade de cada veículo pode ser diferente.

Podemos definir o problema como: seja $U = \{u_0, u_1, \dots, u_n\}$ o conjunto de pontos onde u_0 corresponde ao depósito e os demais n pontos correspondem aos clientes, $G = (U, E)$ um grafo completo sendo U o conjunto de vértices, onde cada vértice

representa o depósito ou os clientes, e $E = \{(i, j) : i, j \in U, i \neq j\}$ o conjunto de arestas. Cada cliente i possui um conjunto de agendas de visitas R_i e uma demanda q_i , idêntica para todos os dias de sua agenda. Considere $V = \{v_1, v_2, \dots, v_w\}$ como o conjunto de veículos disponíveis no depósito. O problema consiste em determinar as rotas dos veículos para cada dia do horizonte de planejamento de tal forma que as demandas dos clientes sejam atendidas e os custos operacionais de transporte sejam minimizadas.

Algumas restrições devem ser satisfeitas: toda rota começa e termina no depósito; um veículo só pode ser utilizado uma vez para cada dia e sua capacidade deve ser respeitada; a distância máxima das rotas deve ser respeitada; todos os clientes devem ser atendidos de acordo com sua agenda de visita ou dias de disponibilidade e cada cliente só pode ser atendido por um único veículo por dia.

Como estamos considerando que a frota é heterogênea, adicionamos ao problema um custo fixo associado a cada veículo. Normalmente, este custo é proporcional à capacidade do veículo, ou seja, quanto maior a capacidade, maior o custo. Esta adição visa aproximar ainda mais a modelagem do problema a situações reais.

Com base na formulação apresentada por Higino et al. [2012], um modelo de Programação Linear Inteira (PLI) para o PRVP com frota de veículos heterogêneos é apresentado. O modelo utiliza os seguintes índices, parâmetros e variáveis de decisão.

Índices

- i, j : clientes
- v : veículos
- l : dias

Parâmetros

- n : número total de clientes.
- w : número total de veículos disponíveis.
- t : número total de dias do horizonte de planejamento
- d_{ij} : distância entre os clientes i e j .
- q_i : demanda do cliente i .
- f_v : custo fixo do veículo v .
- u_{ivl} : fluxo acumulado do veículo v no cliente i no dia l .

- Q_v : capacidade do veículo v .
- T_v : distância máxima da rota do veículo v .
- L : conjunto de dias do horizonte.
- V : conjunto de veículos disponíveis.
- U : conjunto de locais $U = \{u_0, u_1, \dots, u_n\}$, onde u_0 corresponde ao depósito.
- U_c : conjunto de clientes $U_c = \{u_1, u_2, \dots, u_n\}$.
- R_i : conjunto de agendas possíveis para o cliente i .
- $a_{rli} = \begin{cases} 1 & \text{se o cliente } i \text{ deve ser visitado no dia } l \text{ na agenda } r \\ 0 & \text{caso contrário} \end{cases}$

Variáveis de decisão

- $x_{ijvl} = \begin{cases} 1 & \text{se o veículo } v \text{ no dia } l \text{ viaja de } i \text{ para } j \\ 0 & \text{caso contrário} \end{cases}$
- $y_{ir} = \begin{cases} 1 & \text{se a agenda } r \text{ foi escolhida para o cliente } i \\ 0 & \text{caso contrário} \end{cases}$
- $Y_v = \begin{cases} 1 & \text{se o veículo } v \text{ é utilizado} \\ 0 & \text{caso contrário} \end{cases}$

Modelo de PLI

$$\begin{aligned}
& \text{Minimizar } \sum_{v \in V} \sum_{l \in L} \sum_{i \in U} \sum_{j \in U} d_{ij} x_{ijvl} + \sum_{v \in V} f_v Y_v \\
& \text{Sujeto a: } (1) \sum_{r \in R_i} y_{ir} = 1 \quad \forall i \in U_c \\
& \quad (2) \sum_{j \in U, j \neq i} \sum_{v \in V} x_{ijvl} - \sum_{r \in R_i} a_{rli} y_{ir} = 0 \quad \forall i \in U_c; \forall l \in L \\
& \quad (3) \sum_{i \in U, i \neq h} x_{ihvl} - \sum_{j \in U, j \neq h} x_{hjvl} = 0 \quad \forall h \in U_c; \forall l \in L; \forall v \in V \\
& \quad (4) u_{ivl} - u_{jvl} + Q_v x_{ijvl} \leq Q_v - q_j \quad \forall i, j \in U_c; j \neq i; \forall l \in L; \forall v \in V \\
& \quad (5) q_i \leq u_{ivl} \leq Q_v \quad \forall i \in U_c; \forall l \in L; \forall v \in V \\
& \quad (6) \sum_{j \in U_c} x_{0jvl} \leq 1 \quad \forall l \in L; \forall v \in V \\
& \quad (7) \sum_{i \in U} \sum_{j \in U} x_{ijvl} \leq Y_v M \quad \forall v \in V; \forall l \in L \\
& \quad (8) \sum_{i \in U_c} x_{i0vl} = \sum_{j \in U_c} x_{0jvl} \quad \forall l \in L; \forall v \in V \\
& \quad (9) \sum_{i \in U} \sum_{j \in U, j \neq i} d_{ij} x_{ijvl} \leq T_v \quad \forall l \in L; \forall v \in V \\
& \quad (10) x_{ijvl}, y_{ir}, Y_v \in \{0, 1\}; \quad i, j \in U; l \in L; v \in V; r \in R_i
\end{aligned}$$

O objetivo é minimizar a distância total percorrida pelos veículos durante todo o período de planejamento e o custo fixo associado a utilização da frota. As restrições (1) asseguram que uma única agenda é atribuída para cada cliente. As restrições (2) garantem que os clientes são visitados apenas nos dias correspondentes em sua agenda. As restrições (3) garantem que toda vez que um veículo chega a um determinado cliente, em seguida ele deve deixar tal cliente. As restrições (4) e (5) garantem que não há subciclos na rota, utilizando o fluxo. As restrições (6) garantem que cada veículo seja utilizado no máximo uma vez por dia. As restrições (7) determinam quais veículos são usados para atender os clientes, sendo M uma constante suficientemente grande. Neste trabalho o valor adotado foi $M = 10^7$. As restrições (8) garantem que todo veículo que sai do depósito deve retornar ao mesmo. As restrições (9) garantem que a distância máxima de cada rota será respeitada. E por fim, as restrições (10) determinam o domínio das variáveis.

Capítulo 3

Métodos propostos

Para resolução do PRVP propomos três métodos diferentes. O primeiro método proposto é a **heurística ILS**, do inglês *Iterative Local Search*, comumente aplicada em problemas de otimização combinatória pela sua facilidade de implementação e eficiência. A busca local do ILS utiliza o **algoritmo RVND** (*Randomized Variable Neighborhood Descent*), que executa uma busca aleatória em vizinhanças pré definidas. O ILS, juntamente com o RVND, será chamado de ILS-RVND neste trabalho. O segundo método proposto é um **PSO**, do inglês *Particle Swarm Optimization*, que é um algoritmo populacional, assim como o algoritmo genético. E o último é a adaptação do algoritmo PS, do inglês *Proximity Search*, que utiliza o modelo matemático do problema aliado a uma heurística para encontrar soluções mais rapidamente.

Nas próximas seções, todos os três métodos serão detalhadamente explicados e divididos da seguinte maneira: primeiro a definição dos métodos construtivos e buscas locais, ambos utilizados pelas heurísticas ILS e PSO. Em seguida, o detalhamento específico de cada uma das heurísticas, e por fim a apresentação do PS.

3.1 Métodos construtivos

Tanto o ILS quanto o PSO necessitam de uma solução inicial, não necessariamente viável, para começar a execução do algoritmo. **O método construtivo é o responsável por criar esta solução inicial.** Implementamos três métodos construtivos distintos para o PRVP. A escolha de criar mais de um método se justifica devido ao grande impacto da solução inicial sobre a solução final, como pode ser visto na seção de calibração dos algoritmos. Os três métodos serão detalhados nas próximas subseções.

Métodos para solução inicial

3.1.1 Construtivo Cordeau

O primeiro método testado foi proposto por Cordeau et al. [1997] e será chamado neste trabalho de **ConstrutivoCordeau**. Neste método os clientes são ordenados em ordem crescente de acordo com o ângulo que eles formam com o depósito e um raio arbitrário, caso eles estejam representados por coordenadas euclidianas. Caso contrário, eles são ordenados por algum outro critério. Neste ponto, a cada cliente é atribuída uma combinação de visita aleatória escolhida dentre as suas combinações possíveis. Em seguida, a lista de clientes ordenados é dividida em uma posição j qualquer e os clientes são inseridos nas rotas dos dias em que eles precisam ser visitados. A ordem de inserção deve ser, do cliente j ao último cliente da lista e depois do primeiro cliente ao cliente da posição $j - 1$.

A inserção dos clientes nas rotas segue algumas pequenas regras. Os clientes são inseridos no primeiro veículo disponível enquanto estes não extrapolarem sua capacidade máxima. A partir do momento em que não é possível inserir mais clientes neste veículo, o próximo veículo será alocado e os clientes serão inseridos nele, novamente, enquanto sua capacidade máxima suportar. A exceção fica para o último veículo, que terá clientes incluídos em sua rota mesmo que isso ocasione um excesso de carga. Este processo se repete até que todos os clientes sejam alocados. A ordem dos veículos é simplesmente a ordem de leitura das instâncias.

Ordem de entrada Note que, potencialmente, a rota do último veículo será inviável, o que tornará a solução inviável como um todo. Para contornar este problema, uma penalidade é aplicada ao valor da função objetivo da solução de modo que, naturalmente, durante a execução dos algoritmos, a solução se torne viável. O sistema de penalidades será explicado mais a frente. O pseudocódigo deste método pode ser visto no Algoritmo 1.

3.1.2 Construtivo Mortati

O segundo método implementado consiste em uma adaptação do **ConstrutivoCordeau** e foi proposto por Mortati [2005]. Chamaremos este método de **ConstrutivoMortati**. Sua principal diferença em relação ao **ConstrutivoCordeau** está na forma de inserção dos clientes nas rotas. Neste método, se a inserção de um cliente na rota viola a restrição de capacidade do veículo, então ele deve ser inserido na rota de menor custo. Este processo se repete até que todos os clientes tenham sido inseridos em alguma rota para seus respectivos dias de visita.

Algorithm 1 ConstrutivoCordeau(*clientes*)

```

1: if (clientes estão representados por coordenadas euclidianas) then
2:   Ordená-los crescentemente em relação ao ângulo que eles formam com o depósito
   e um raio arbitrário
3: else
4:   Ordená-los segundo um critério
5: end if
6: Associar a cada cliente c uma combinação de visita aleatória pertencente ao seu
   conjunto de combinações
7: for all dia l in horizonte do
8:   Considerando a ordenação acima, escolha um cliente j aleatoriamente
9:    $k = 1$ ;
10:  Usando a sequência de clientes  $j, j + 1, \dots, n, 1, \dots, j - 1$ , executar os seguintes
   passos para cada cliente c que possua dia de visita l;
11:  if (A inserção do cliente c na rota k no dia l resultar na violação de capacidade)
   then
12:     $k = \min(k + 1, m)$  sendo m o número total de veículos disponíveis
13:  end if
14:  Inserir o cliente c na rota k no dia l
15: end for

```

3.1.3 Construtivo Aleatório

O método construtivo proposto, nomeado **ConstrutivoAleatorio**, é um método totalmente aleatório. Para todo cliente *i* do conjunto de clientes, é atribuída uma agenda de visitas aleatória pertencente ao seu conjunto de agendas disponíveis R_i . Em seguida, o cliente *i* é inserido nos dias correspondentes na agenda de visita escolhida em um veículo aleatório pertencente ao conjunto de veículos V . O pseudocódigo deste método pode ser visto no Algoritmo 2.

Algorithm 2 ConstrutivoAleatorio(*clientes*)

```

1: Inicializa a solução s com todas as rotas dos veículos vazias.
2: for (todo cliente i em  $U_c$ ) do
3:   Selecione uma agenda de visita r aleatória pertencente a  $R_i$ 
4:   for (todo dia l em  $L$ ) do
5:     if (cliente i precisa ser visitado no dia l na agenda r) then
6:       Selecione um veículo aleatório v pertencente ao conjunto de veículos de s
7:       Insira i na rota de v do dia l na última posição.
8:     end if
9:   end for
10: end for
11: Retorne s;

```

3.2 Penalidade

O método construtivo utilizado, bem como os eventuais movimentos aplicados sobre as rotas, podem fazer com que a solução se torne inviável, ou seja, a restrição de capacidade máxima dos veículos ou distância máxima percorrida em cada rota seja violada. Para evitar que a solução final retornada pelo algoritmo ILS possua alguma dessas inviabilidades, um método para penalizar as soluções inviáveis foi adotado. O método utilizado neste trabalho foi proposto por Cordeau et al. [2001]. A penalidade α é aplicada sobre o excesso de carga e a penalidade β é aplicada sobre o excesso de distância. Dessa forma, o cálculo final da função objetivo de uma solução pode ser definido como:

Função objetivo (Custo):

$$\sum_{v \in V} \sum_{l \in L} \sum_{i \in U} \sum_{j \in U} c_v d_{ij} x_{ijvl} + \sum_{v \in V} f_v Y_v + \alpha \times Q_{extra} + \beta \times D_{extra}$$

Onde Q_{extra} , o excesso de carga, é definido como:

$$Q_{extra} = \sum_{v \in V} \max\left[\left(\sum_{i \in U_c} \sum_{j \in U_c} x_{ijvl} q_j\right) - Q_v; 0\right] \quad \forall l \in L$$

E D_{extra} , o excesso de distância, como:

$$D_{extra} = \sum_{v \in V} \max\left[\left(\sum_{i \in U_c} \sum_{j \in U_c} x_{ijvl} d_{ij}\right) - T_v; 0\right] \quad \forall l \in L$$

A utilização destas penalidades conduz o algoritmo a soluções viáveis uma vez que o valor da função objetivo de uma solução inviável tenderá a ser maior do que o valor das soluções viáveis.

3.3 Buscas locais

Os procedimentos de buscas locais aplicados no ILS e no PSO utilizam movimentos que são comuns a problemas de roteamento de veículos. Os movimentos podem ser divididos em duas classes, os intra-rota, aqueles em que os clientes são movidos dentro de uma mesma rota, e os inter-rota, onde os clientes são movidos para rotas diferentes. Foram utilizados seis movimentos inter-rota e três movimentos intra-rota. Cada movimento é executado para todas as rotas de todos os dias da solução a qual é aplicada

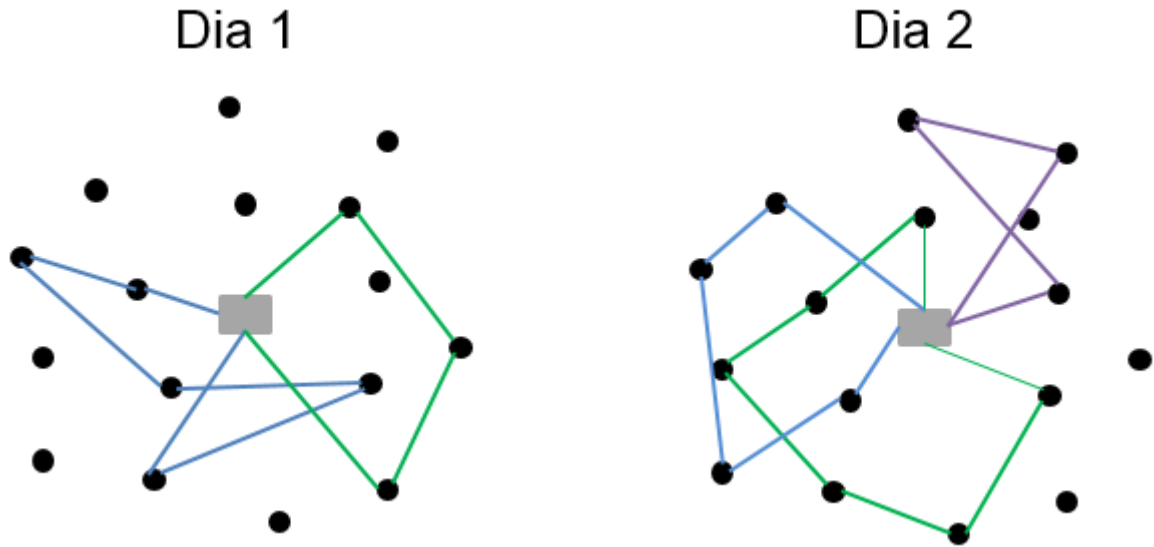


Figura 3.1. Configuração inicial da solução.

e o cliente só é movido se isto levar a uma redução na distância total percorrida do veículo. O detalhamento de cada movimento pode ser visto nas subseções seguintes. Para os exemplos gráficos a seguir, considere a Figura 3.1 como a configuração inicial da solução. Neste exemplo específico, 16 clientes precisam ser atendidos durante um período de dois dias.

3.3.1 Movimentos intra-rota

Os movimentos intra-rota utilizados foram Swap(1), 2-Opt e Or-Opt(1). O movimento Swap, como o nome sugere, consiste em trocar a posição de dois clientes. Selecione dois clientes i e j dentro da rota A. Coloque o cliente i na posição do cliente j e vice-versa. Este movimento pode ser visto na Figura 3.2.

O movimento 2-Opt seleciona dois arcos da rota para serem removidos e insere dois novos arcos no lugar, trocando a posição dos clientes envolvidos. Selecione um arco (i, j) e um arco (i', j') para serem substituídos pelos novos arcos (i, i') e (j, j') . Este movimento pode ser visto na Figura 3.3

O movimento Or-Opt remove u clientes da rota e em seguida, reinsere-os na posição de menor custo. Neste trabalho utilizamos $u = 1$, ou seja, um cliente será removido e inserido na posição de menor custo de sua rota. Selecione um cliente i pertencente a rota A e retire-o da rota. Em seguida, tente reinserir o cliente i em todas as posições da rota A mantendo-o na que apresentar menor distância total. Este

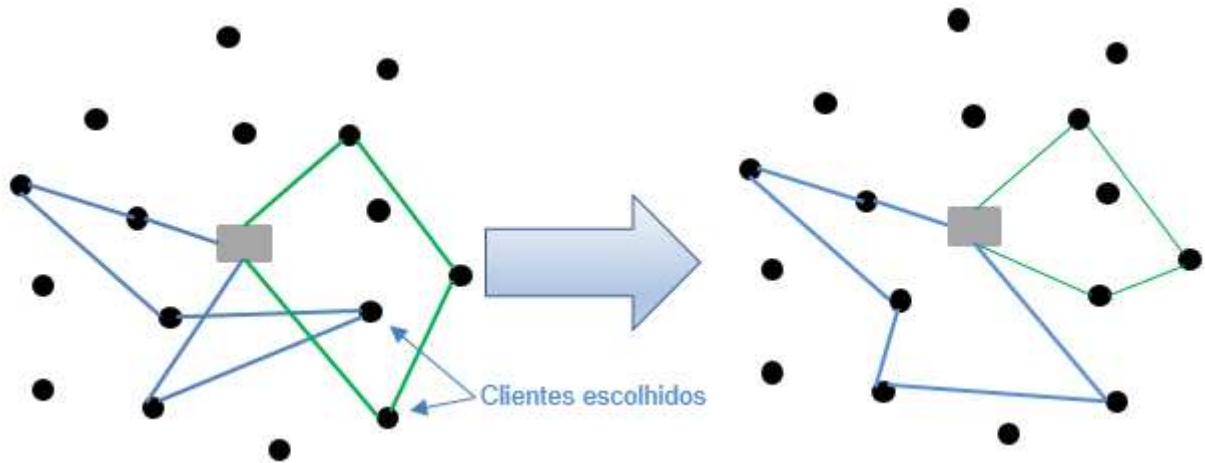


Figura 3.2. Exemplo de movimento Swap(1).

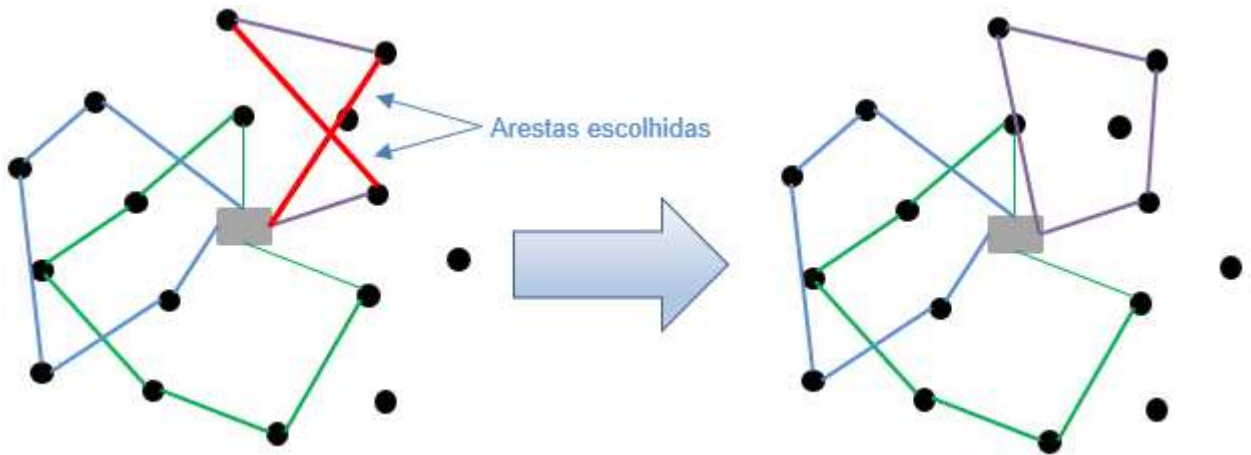


Figura 3.3. Exemplo de movimento 2-Opt.

movimento pode ser visto na Figura 3.4.

3.3.2 Movimentos inter-rota

Os movimentos inter-rota utilizados foram Swap, Shift, Cross e Radial. O movimento Swap funciona de forma semelhante ao apresentado na seção intra-rota, a diferença aqui é que u clientes de rotas distintas são selecionados para a troca. Neste trabalho utilizamos $u = 1$ (Swap(1,1)) e $u = 2$ (Swap(2,2)). Considerando o Swap(1,1), seja i um cliente pertencente a rota A e j um cliente pertencente a rota B, retire o cliente i da rota A e insira-o na rota B na posição de j . Em seguida, retire j da rota B e insira na rota A na posição de i . Este movimento pode ser visto na Figura 3.5

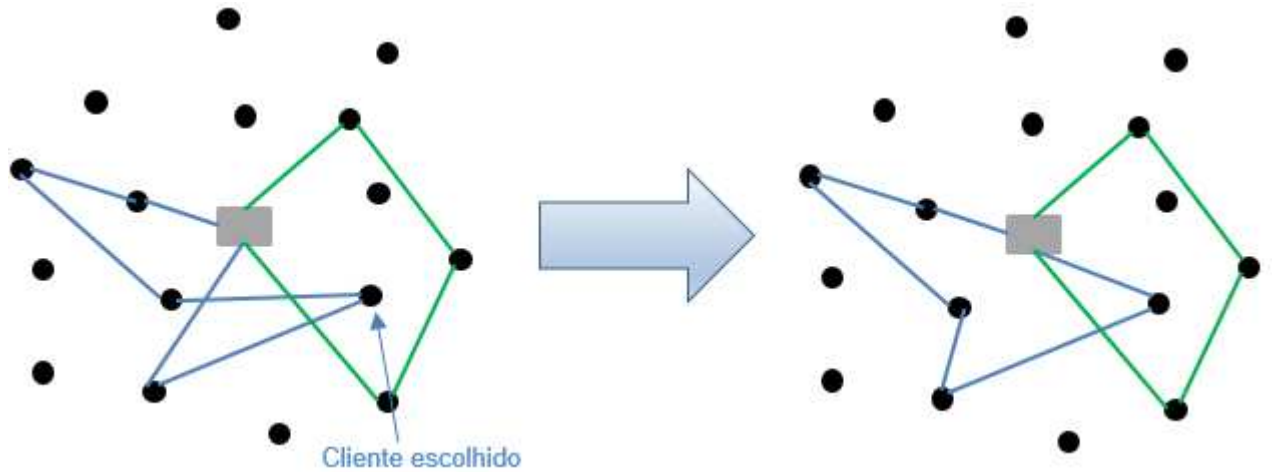


Figura 3.4. Exemplo de movimento Or-Opt(1).

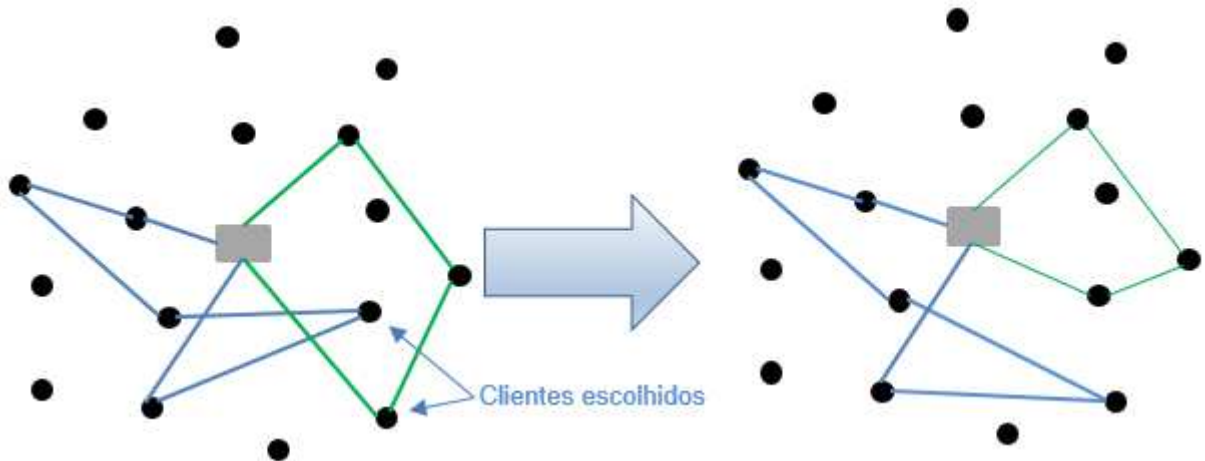


Figura 3.5. Exemplo de movimento Swap(1,1).

O movimento Shift consiste em retirar uma sequência de u clientes de uma rota e inserir em outra. Neste trabalho utilizamos $u = 1$ (Shift(1)) e $u = 2$ (Shift(2)). Considerando o Shift(1), selecione um cliente i pertencente a rota A e remova-o desta rota. Em seguida, insira o cliente i na rota B na posição de menor custo. Este movimento pode ser visto na Figura 3.6.

O movimento Cross, baseado no operador *cross exchange* de Hemmelmayr et al. [2009b], visa retirar cruzamentos entre as rotas. Um arco (i, j) pertencente a rota A e outro arco (i', j') pertencente a rota B são removidos, e dois novos arcos são inseridos, sendo eles (i', j) e (i, j') . Este movimento pode ser visto na Figura 3.7.

O movimento Radial consiste em adicionar clientes de outras rotas à rota atual

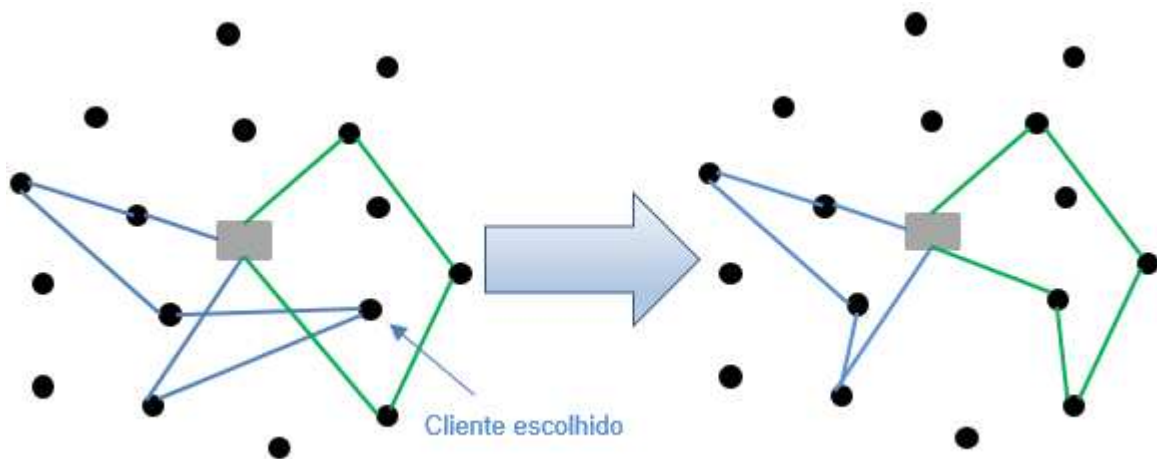


Figura 3.6. Exemplo de movimento Shift(1).

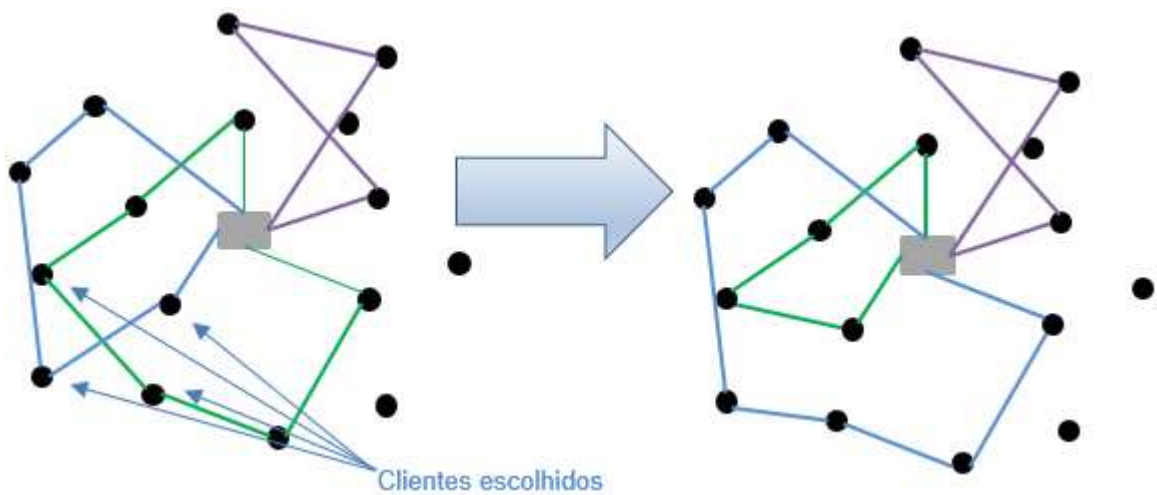


Figura 3.7. Exemplo de movimento Cross.

desde que estes estejam dentro de um raio máximo centrado no cliente atualmente analisado. Este movimento foi inspirado pela busca local utilizada por Gonçalves [2005]. Para todo cliente i da rota A , é feita uma verificação para determinar se existe algum outro cliente j de outra rota que está dentro de um determinado raio centrado em i . Caso exista, este cliente j será inserido na rota A na posição adjacente a i de menor custo. Este movimento pode ser visto na Figura 3.8.

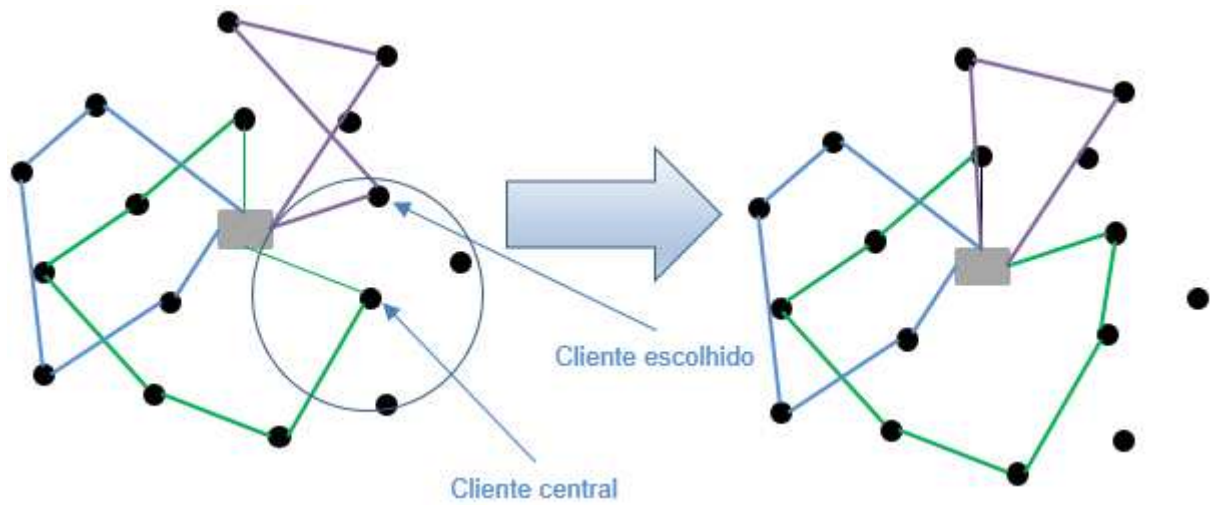


Figura 3.8. Exemplo de movimento Radial.

3.3.3 Otimiza agendas

Para tentar conduzir a solução a um estado de melhor qualidade, na qual todos os clientes sejam atendidos e os veículos percorram a menor distância possível, é necessário que as agendas de visitas dos clientes sejam verificadas. Até o momento, todos os movimentos apresentados modificam apenas as rotas já existentes sem que isso altere os dias de visita escolhidos pelos clientes. Este processo diminui o espaço de busca visto que outras opções de visita não são exploradas. Tentando contornar este problema criamos uma busca local chamada de `OtimizaAgendas()`.

A otimização de agendas consiste em se verificar para todo cliente $i \in U_c$, de maneira arbitrária, se existe alguma outra agenda de visita que levaria a um ganho no custo da solução. Assim, para todas as agendas disponíveis para o cliente i , é feita uma simulação de troca da agenda atual pela próxima na lista na solução corrente. Caso existam agendas melhores que a atual, aquela que obtiver o menor custo será a agenda escolhida. O pseudocódigo do método `OtimizaAgendas()` pode ser visto no Algoritmo 3.

Na linha 1 do Algoritmo 3, uma lista auxiliar é criada contendo todos os clientes existentes. Nas linhas 4 e 5, o valor da função objetivo da solução e a agenda de visita atuais do cliente i são armazenados. Nas linhas 6 a 14, é feita a simulação da alteração da agenda de visitas do cliente i e o melhor valor obtido é armazenado, juntamente com qual agenda conseguiu este valor. Na linha 16, a agenda de i será alterada em caso exista alguma agenda que melhore a solução atual. O processo de simulação possui basicamente duas etapas: remover o cliente das rotas nos dias aos quais ele não

Algorithm 3 OtimizaAgendas(Solução s)

```

1: Inicialize a lista  $C$  com todos os clientes existentes em  $U_c$ .
2: while ( $C \neq \emptyset$ ) do
3:   Selecione aleatoriamente um cliente  $i \in C$ .
4:   Inicialize o valor de  $bestFO$  com o  $Custo(s)$ .
5:   Inicialize  $bestR$  com a agenda atualmente escolhida para  $i$ .
6:   for (toda agenda  $r \in R_i$ ) do
7:     if ( $r$  não for a agenda atualmente escolhida para  $i$ ) then
8:       Faça a simulação da troca da agenda atual de  $i$  para  $r$  em  $s$  retornando a nova solução  $s'$ .

9:       if ( $Custo(s') < bestFO$ ) then
10:          $bestFO = Custo(s')$ ;
11:          $bestR = r$ ;
12:       end if
13:     end if
14:   end for
15:   if  $bestFO < Custo(s)$  then
16:     Troque a agenda atual de  $i$  por  $bestR$  em  $s$ .
17:   end if
18:   remova o cliente  $i$  de  $C$ .
19: end while
20: Retorne  $s$ ;

```

pertence mais e adicionar o cliente as novas rotas. A adição do cliente nas novas rotas é feita testando a inserção do novo cliente em todas as posições possíveis das rotas já existentes no novo dia e selecionado aquela em que o aumento no custo seja o mínimo. Caso a solução se torne inviável, a penalidade será aplicada.

Considere, por exemplo, o cliente i com suas respectivas agendas de visita $R_i = \{\{1, 2\}, \{2, 3\}\}$. Isso significa que na agenda 1 o cliente i deve ser visitado nos dias 1 e 2, e na agenda 2 o cliente i deve ser visitado nos dias 2 e 3. Suponha que na solução atual, a agenda escolhida para i seja a agenda 1. Quando o cliente i for escolhido durante o método de otimização, uma simulação será feita trocando a agenda atualmente escolhida, no caso a 1, por todas as outras disponíveis. Neste caso, i possui apenas mais uma agenda, então a agenda 1 será trocada pela 2. Isso implica que o cliente i será removido da rota em que ele estiver no dia 1, não sofrerá alteração no dia 2 e será inserido na rota de menor custo no dia 3. Se esta movimentação acarretar em uma redução no custo total da solução, a agenda atual do cliente i será trocada para a 2, caso contrário a 1 será mantida. Este processo é feito para todos os clientes em ordem aleatória.

3.4 Iterated Local search (ILS)

Iterated Local Search (ILS) é uma meta-heurística bastante utilizada na literatura para solucionar problemas de otimização NP-Difícil (Lourenço et al. [2003]). A ideia desta meta-heurística é utilizar um método de busca local que explora o espaço de soluções por meio de perturbações de soluções ótimas locais geradas durante a busca. De acordo com Tang & Wang [2006], o ILS se destaca por ser uma meta-heurística eficiente e simples.

Na implementação de um algoritmo ILS, quatro passos devem ser especificados. Primeiramente, o processo de construção da solução inicial se dá pelo método `CriaSolucao()`. Em seguida, o procedimento `AplicaBuscaLocal()` recebe a solução atual e retorna um ótimo local. O terceiro procedimento de perturbação denominado `PerturbaSolução()`, altera alguma característica da solução corrente para que, com isso, o processo de busca local consiga explorar um espaço de busca maior. Por fim, o critério de aceitação de novas soluções é aplicado.

Além dos quatro passos citados anteriormente, este trabalho adiciona mais dois procedimentos ao método ILS. O primeiro se trata do reinício da solução atual. Após um determinado número de iterações, se a solução atual não apresentar melhora no valor da função objetivo, ela será reiniciada. Ou seja, a solução atual será descartada e uma nova solução será gerada pelo método construtivo `CriaSolucao()`. O segundo considera a inclusão do procedimento de busca local `OtimizaAgendas()`. Este método será chamado sempre que a solução atual não apresentar melhora após o processo de busca local `AplicaBuscaLocal()`. O pseudocódigo do ILS implementado neste trabalho é apresentado no Algoritmo 4.

Na linha 2 do Algoritmo 4 a solução inicial s é gerada e atribuída a melhor solução conhecida s^* . A solução atual s é melhorada na linha 5 pelo procedimento de busca local, retornando assim a solução s' que é o ótimo local obtido a partir de s . Nas linhas 6 a 10, o critério de aceitação é avaliado. A solução atual s e a melhor solução encontrada pelo algoritmo s^* são atualizadas caso o valor da função objetivo de s' seja menor. Caso contrário, na linha 13, o processo de otimização de agendas de visita é chamado. Nas linhas 15 a 17, tem-se o reinício da solução atual. Na linha 19 a solução atual é perturbada. Por fim, na linha 23, a melhor solução encontrada após todas as iterações do método é retornada. Nas próximas seções os elementos principais do ILS estão detalhados.

O procedimento `CriaSolucao()` utiliza um dos três métodos construtivos descritos anteriormente (`ConstrutivoCordeau`, `ConstrutivoMortati`, `ConstrutivoAleatorio`). Na seção de calibração e testes será detalhado o uso

Algorithm 4 ILS()

```

1:  $iterSemMelhora = iterAtual = 0$ ;
2:  $s = CriaSolucao()$ ;
3:  $s* = s$ ;
4: while ( $iterAtual < maxIter$ ) do
5:    $s' = AplicaBuscaLocal(s)$ ;
6:   if ( $Custo(s') < Custo(s)$ ) then
7:      $s = s'$ ;
8:     if ( $Custo(s) < Custo(s*)$ ) then
9:        $s* = s$ ;
10:    end if
11:  else
12:     $iterSemMelhora = iterSemMelhora + 1$ ;
13:     $OtimizaAgendas(s)$ ;
14:  end if
15:  if ( $iterSemMelhora == maxIterSemMelhora$ ) then
16:     $s = Construtivo()$ ;
17:     $iterSemMelhora = 0$ ;
18:  else
19:     $s = PerturbaSolucao(s)$ ;
20:  end if
21:   $iterAtual = iterAtual + 1$ ;
22: end while
23: Retorne  $s*$ ;

```

de cada um destes. O procedimento `PerturbaSolução()` alterará a solução atual no intuito de conduzi-la a uma região diferente no espaço de busca. O processo de perturbação será descrito nas próximas seções.

O procedimento `AplicaBuscaLocal()` aplica os movimentos intra-rota e inter-rota. Dois tipos de busca local foram utilizadas no ILS. A primeira, chamada de `BuscaLocalEstática`, aplica os movimentos em uma ordem pré-definida. Esta ordem está descrita na seção de calibrações. A segunda busca local utiliza o método RVND para aplicação dos movimentos e está descrita na próxima seção.

3.4.1 ILS-RVND

No intuito de melhorar a qualidade das soluções encontradas pelo ILS, implementamos uma busca local baseada na heurística VND (*Variable Neighborhood Descent*) com sua característica aleatória, ou seja, a ordem de aplicação das vizinhanças não é definida, e sim determinada aleatoriamente a cada iteração. Esta variação do VND é denominada de RVND (*Randomized VND*). Neste trabalho, ao iniciar o RVND, uma estrutura de

vizinhança aleatória é escolhida dentre as possíveis no conjunto de movimentos Iter-Rota. Caso haja melhora na solução após a aplicação dos movimentos da vizinhança escolhida, a mesma estrutura será aplicada novamente na solução modificada. Caso não haja melhora, a estrutura escolhida será removida do conjunto e uma nova será selecionada aleatoriamente dentre as restantes. O RVND termina quando o conjunto de buscas inter-rota fica vazio.

O conjunto de vizinhanças utilizado para o RVND contém seis elementos. Utilizamos $\text{Shift}(u, u)$ e $\text{Swap}(u, u)$ com duas versões cada, uma considerando u igual a 1 e outra com u igual a 2. As duas restantes são as demais buscas inter-rota, Cross e Radial. A vizinhança gerada por cada um desses elementos consiste na utilização do movimento da busca local, aplicado exaustivamente na solução até que nenhuma melhora seja encontrada. Se após a utilização de alguma dessas vizinhanças a solução apresentar melhora, ela será submetida ao processo de buscas locais intra-rota. Neste processo, as buscas Swap(1), 2-Opt e Or-Opt serão utilizadas em todas as rotas da solução que sofreram alguma alteração. O pseudocódigo do RVND pode ser visto no Algoritmo 5.

Algorithm 5 RVND(Solução s)

```

1: Inicialize a lista de vizinhanças  $LV$  com as seis vizinhanças inter-rota
2: while ( $LV \neq \emptyset$ ) do
3:   Selecione aleatoriamente uma vizinhança  $k$  pertencente a  $LV$ 
4:   Encontre o melhor vizinho  $s'$  pertencente a vizinhança  $k(s)$ 
5:   if ( $\text{Custo}(s') < \text{Custo}(s)$ ) then
6:     Aplique os movimentos intra-rota nas rotas alteradas em  $s'$ 
7:      $s = s'$ ;
8:   else
9:     Remova a vizinhança  $k$  do conjunto  $LV$ 
10:  end if
11: end while
12: Retorne  $s$ ;

```

Na linha 1, o conjunto de vizinhanças LV é inicializado com as seis vizinhanças inter-rota: $\text{Shift}(1,1)$, $\text{Shift}(2,2)$, $\text{Swap}(1,1)$, $\text{Swap}(2,2)$, $\text{Cross}()$ e $\text{Radial}()$. Na linha 3, uma dessas vizinhanças é escolhida aleatoriamente para ser aplicada a solução s . Na linha 4, o melhor vizinho s' obtido após a aplicação da vizinhança escolhida na solução s é recuperado. Se o valor da função objetivo de s' é menor que o de s , então os movimentos intra-rota serão aplicados em s' . A ordem de aplicação dos movimentos intra-rota é a mesma utilizada na *BuscaLocalEstatica*. Se não houve melhora em s após a aplicação da vizinhança escolhida, esta é removida da lista LV na linha 9. O algoritmo termina quando o conjunto LV estiver vazio.

3.4.2 Perturbações

Uma fase essencial do ILS consiste na perturbação da solução atual. No caso do PRVP, uma estratégia eficiente de perturbação consiste em alterar as agendas de visita atribuídas aos clientes visando aumentar o espaço de busca explorado. Três métodos de perturbação foram implementados.

O primeiro método, chamado Perturbação1, consiste em se alterar a agenda de visita de uma determinada porcentagem de clientes para a próxima agenda na lista. Considere a lista de agendas de visita possíveis como uma lista circular. Esta porcentagem é calculada sobre o número de clientes que podem ter sua agenda alterada, ou seja, aqueles que possuem mais de uma agenda de visita disponível. A porcentagem de alteração foi de 10%, sendo o número mínimo de clientes alterado igual a 1, a menos que a instância não possua nenhum cliente com mais de uma opção de agenda de visitas.

O segundo, chamado Perturbação2, é muito semelhante à primeira. A diferença está na pré-validação da nova agenda do cliente antes de alterá-la, ou seja, a agenda do cliente escolhido somente será alterada caso ela não torne a solução atual inviável pela capacidade. Em outras palavras, é feito o cálculo da nova demanda total dos clientes, considerando a nova agenda escolhida para o cliente atual. Se a soma da demanda de todos os clientes de um determinado dia for superior a soma das capacidades máximas de todos os veículos disponíveis, esta agenda é descartada e um novo cliente é escolhido. Graças ao fator aleatório da escolha dos clientes e a própria configuração de determinadas instâncias, este método tem um limite máximo de tentativas por iteração igual a 50% do número total de clientes da instância.

O terceiro, denominado de Perturbação3, irá alterar um número x de clientes baseado no número k de iterações sem melhora da solução atual. Desse modo, x clientes, sendo x o mínimo entre k e o máximo entre 20% do número total de clientes ou o número total de clientes que possuem mais de uma agenda de visita, terão sua agenda atual de visita alterada. Em outras palavras, x pode ser definido como $x = \min(k; \max(0, 2 * n; |U_r|))$ sendo U_r o conjunto de todos os clientes em que $|R_i| > 1$.

3.5 Particle Swarm Optimization (PSO)

O PSO, ou Otimização por Nuvem de Partículas, é uma heurística populacional muito semelhante ao algoritmo genético. Esta heurística foi proposta por Eberhart e Kennedy [1995], e desde então tem sido aplicada a diversos problemas de otimização. Baseando-se no comportamento social de um bando de pássaros em revoada, com movimento

localmente aleatório mas globalmente determinado, os autores (Eberhart & Kennedy [1995] Kennedy [1997]) desenvolveram uma heurística que aproveita as melhores características deste cenário. No PSO, existe um conjunto de indivíduos chamado de nuvem de partículas. Cada partícula representa uma solução e a iteração entre elas busca encontrar melhores valores para a função objetivo do problema.

O fluxograma básico do PSO pode ser descrito da seguinte maneira: primeiro, um conjunto de partículas é inicializado. Cada partícula tem o valor de sua função objetivo determinado e a melhor delas recebe a nomenclatura de g_{best} . Em seguida, as partículas começam a se mover no espaço de busca. Este movimento é regido por um vetor velocidade que cada partícula possui. A cada iteração todas as partículas fazem um único movimento e o valor de g_{best} é atualizado. Após um determinado número de iterações, a partícula g_{best} é retornada e o algoritmo termina.

O vetor que determina para onde as partículas irão a cada iteração é chamado de vetor velocidade. Este vetor é influenciado por três vetores chamados de vetor de inércia, vetor de individualidade e vetor de sociabilidade. Cada um desses vetores contribui para a direção, sentido e valor de velocidade, como pode ser visto na Figura 3.9. O vetor de inércia determina o quanto a partícula deve tender a continuar se movendo na direção em que ela está. O vetor de individualidade contribui para que a partícula se mova para a melhor posição que ela já esteve, chamada de p_{best} . Por fim, o vetor de sociabilidade puxa a partícula para a melhor posição de toda a nuvem, ou seja, a posição encontrada por g_{best} .

Cada um destes vetores possui uma certa chance de ser considerado e a soma de todos eles resulta no vetor velocidade utilizado por cada partícula em cada iteração. Considerando P_b a melhor posição já encontrada pela partícula p , P_g a posição atual de g_{best} e P_p a posição atual da partícula p , temos que o vetor velocidade da partícula p na iteração k é definido pela seguinte equação:

$$V_p^k = W * V_p^{k-1} + c_1 * r_1 * (P_p - P_b) + c_2 * r_2 * (P_p - P_g). \quad (3.1)$$

Onde, W é o fator de inércia, c_1 é o fator de individualidade e c_2 o fator de sociabilidade. Os números r_1 e r_2 são valores entre 0 e 1 determinados aleatoriamente a cada iteração. Com o resultado desta equação, cada partícula irá se movimentar no espaço de busca na direção determinada. Os fatores W , c_1 e c_2 tem impacto direto sobre a qualidade das soluções encontradas e seus valores serão devidamente definidos no capítulo de calibração. O pseudocódigo do PSO pode ser visto no Algoritmo 6;

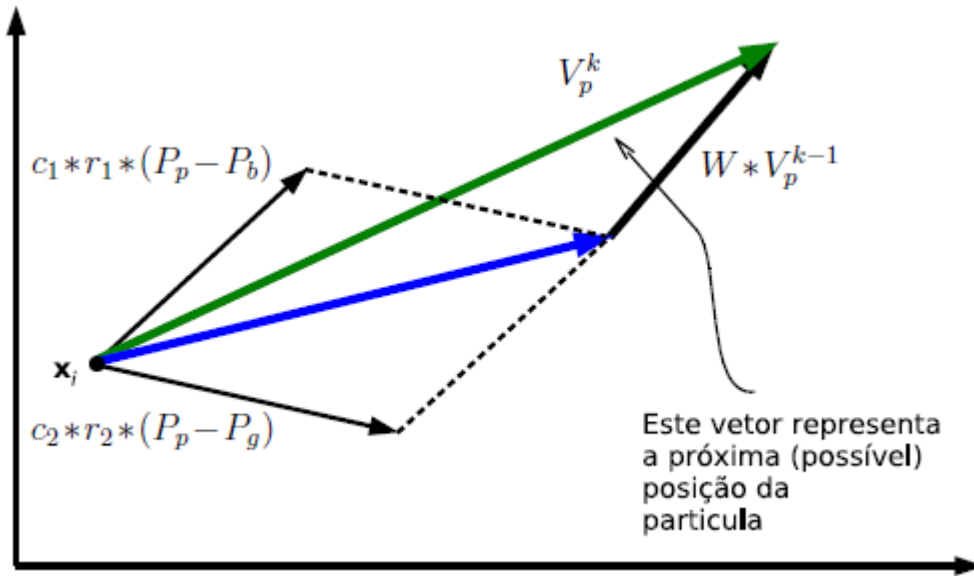


Figura 3.9. Cálculo do vetor velocidade da partícula i . Onde p_i é a melhor posição já encontrada pela partícula i , g é a posição da melhor partícula da nuvem e x_i é a posição atual da partícula i . Adaptado: de Lacerda [2007].

3.5.1 PSO Discreto

Devido à natureza do problema, o PSO apresentado, chamado de PSO Clássico, não pode ser diretamente aplicado. Para problemas de otimização combinatória como é o caso do PRVP, uma pequena alteração deve ser feita no algoritmo, mais especificamente na equação que rege a velocidade. Esta modificação faz com que o PSO seja caracterizado como PSO Discreto.

As modificações propostas para o PSO Discreto podem ser ditas de certa forma, elegantes, pois elas preservam toda a estrutura do algoritmo PSO original e, além disso, insere o PSO Discreto em uma nova classe de problemas. As mudanças sugeridas foram: os vetores de posição atual e melhor posição foram substituídos por valores discretos, e o vetor velocidade por probabilidades, onde um valor discreto em particular modificará uma determinada iteração.

No problema abordado neste trabalho, a solução é representada por um conjunto de strings, sendo que cada string armazena a rota de um determinado veículo. Dessa forma, cada solução possui $|L|*|V|$ strings, ou seja, para cada dia do problema existem $|V|$ strings que representam a rota de cada veículo v para o dia l . Além disso, existe também uma string que identifica qual agenda de visita r_i cada cliente está utilizando. Chamaremos esta string de X_p , a string que contém as combinações de visita de cada cliente $i \in n$ para a partícula p . É sobre esta string que a equação será aplicada.

Algorithm 6 PSO()

```

1: Inicializa cada partícula da população com o método construtivo atribuindo-as a
   lista  $P$ 
2: for (toda partícula  $p$  em  $P$ ) do
3:   BuscaLocal( $p$ )
4:   Salva a solução atual em  $p_{best}$ 
5: end for
6: Inicializa  $g_{best}$ 
7: while (Número máximo de iterações não for atingido) do
8:   for (toda partícula  $p$  em  $P$ ) do
9:     Movimenta a partícula  $p$  de acordo com a equação ou padrão estabelecido
10:    BuscaLocal( $p$ )
11:    if (A partícula  $p$  deve ser otimizada) then
12:      OtimizaAgendas( $p$ )
13:    end if
14:    Atualiza o valor de  $p_{best}$  se ele for melhor que o atual
15:  end for
16:  Atualiza o valor de  $g_{best}$ 
17: end while
18: Retorne  $g_{best}$ ;

```

No PSO proposto, a equação determina qual a chance de cada elemento de X ser modificado. A cada iteração, é avaliado o efeito da equação para cada posição de X . Há apenas três possíveis ações a serem tomadas: não alterar o valor atual, ou seja, o fator de inércia prevaleceu. Alterar o valor para o valor da melhor posição já encontrada pela partícula p , caso em que o fator de individualidade prevalece. E por fim, alterar para o valor da melhor partícula da nuvem, contribuição do fator social.

Para ficar mais claro, considere o seguinte exemplo. Em uma determinada execução do PSO, temos os valores de $W = 3$, $c_1 = 2$ e $c_2 = 5$. A princípio, isso significa que a partícula terá 30% de seus clientes inalterados, 20% assumirão o melhor valor já encontrado por ela mesma e 50% de assumirão o valor da melhor partícula do enxame. A soma de W , c_1 e c_2 não pode ser maior que 10, ou 100%. Além disso, ainda existem os valores de r_1 e r_2 que são determinados a cada iteração. Neste exemplo vamos supor que ambos assumiram o valor de 0,5. Para uma dada iteração k do algoritmo, temos as seguintes strings:

$$g_{best} = \{1, 5, 2, 5, 2, 6, 3, 2, 1, 3\}$$

$$p_{best} = \{2, 3, 4, 5, 1, 3, 3, 3, 2, 1\}$$

$$X_p = \{1, 4, 2, 1, 2, 1, 1, 4, 3, 4\}$$

Neste exemplo, a instância possui 10 clientes. Como cada posição na string significa a agenda de visita escolhida, isso significa que em g_{best} , por exemplo, o cliente

1 está utilizando a agenda 1, o cliente 2 a agenda 5 e assim por diante. Agora, é preciso determinar qual o trecho da string será afetado por cada elemento da equação, ou seja, qual cliente será influenciado por cada fator. Como $r1 = 0,5$ e $c1 = 2$, a multiplicação de $r1 * c1$ resulta em 1, o que significa 10% dos clientes sofrerão a ação do fator individualista (1 cliente, neste exemplo). Analogamente temos $c2 * r2 = 2,5$. Como não é possível movimentar uma fração dos clientes apenas a parte inteira é considerada, neste caso, 2 clientes. A escolha dos clientes é aleatória.

Supondo que os clientes escolhidos tenham sido o cliente 2 para $c1$ e os clientes 5 e 6 para $c2$, temos que a combinação final para a partícula p será:

$$X_p = \{1, \mathbf{3}, 2, 1, \mathbf{2}, \mathbf{6}, 1, 4, 3, 4\}$$

A agenda do cliente 2 foi alterada de 4 para 3, que é o valor atual em p_{best} . A agenda do cliente 5 não foi alterada pois ela era igual ao valor de g_{best} e o valor da agenda do cliente 6 foi alterado de 1 para 6. Os demais clientes continuam iguais, seguindo o fator de inércia.

Alterar a agenda de visita de um cliente significa alterar os dias em que o mesmo deve ser visitado. Esta alteração é feita da seguinte maneira. Para todos os dias em que o cliente precisava ser visitado mas não precisa mais, ele será removido da rota em que estava. Para todos os novos dias em que ele precisa ser visitado, ele será inserido na rota de menor custo. Para os demais dias, não há alteração.

3.5.2 Mutação

A fim de aumentar a variabilidade das partículas na nuvem, foi introduzido ao algoritmo um operador de mutação. Este operador atua sobre as strings das rotas e das agendas de visitação. A cada iteração, toda partícula tem uma determinada chance de sofrer mutação. A chance, ou porcentagem de mutação, foi determinada através de um processo de calibração.

A mutação é dividida em duas etapas. Primeiro, alguns clientes da partícula tem suas agendas de visita modificadas. Em seguida, outros clientes tem suas rotas alteradas. O número de clientes que possuem sua combinação alterada é $\delta\%$ do total, e o número de clientes que possuem sua rota alterada é $\gamma\%$ do total, podendo repetir o cliente nos dois procedimentos. A alteração da agenda de visita é a mesma utilizada na movimentação da partícula. A alteração na rota pode ser apenas alterar a posição do cliente dentro da rota em que ele está (Tipo 1) ou alterá-lo de rota (Tipo 2). Os valores de δ e γ são, respectivamente, 20% e 10%. O pseudocódigo do método de mutação pode ser visto no algoritmo 7.

Algorithm 7 Muta    (Part  cula p)

```

1: Selecione Inicialize o conjunto  $cli1$  com os  $\delta\%$  clientes de  $p$  selecionados aleatoria-
   mente
2: for (todo cliente  $i$  em  $cli1$ ) do
3:   if ( $|R_i| > 1$ ) then
4:     Selecione aleatoriamente uma agenda diferente para o cliente  $i$  pertencente a
        $R_i$ 
5:     Retire o cliente  $i$  das rotas dos dias em que ele n  o precisa mais ser visitado
6:     Insira o cliente  $i$  na posi     de menor custo das rotas dos novos dias em que
       ele precisa ser visitado
7:   end if
8: end for
9: Inicialize o conjunto  $cli2$  com os  $\gamma\%$  clientes de  $p$  selecionados aleatoriamente
10: for (todo cliente  $i$  em  $cli2$ ) do
11:   selecione aleatoriamente o tipo de altera     na rota
12:   Remova o cliente  $i$  das rotas em que ele atualmente est  
13:   if (Alter     selecionada    do tipo 1) then
14:     Insira o cliente  $i$  na mesma rota em que ele estava, mas em uma posi    
       aleat  ria
15:   end if
16:   if (Alter     selecionada    do tipo 2) then
17:     Insira o cliente  $i$  na rota de outro ve  culo qualquer diferente do original
18:   end if
19: end for

```

3.6 Proximity Search (PS)

O Proximity Search (PS), introduzido por Fischetti & Monaci [2014],    um m  todo de resolu     de problemas de programa     linear inteira mista (PLIM) do tipo bin  rio (0-1). O PS    uma modifica     na forma como o modelo matem  tico    executado em um *solver*, acrescentando uma estrat  gia heur  stica que busca melhores solu    es a cada itera    . Sua ideia principal    alterar a fun    o objetivo do problema para uma fun    o de proximidade e utilizar a solu    o obtida com esta nova fun    o como uma heur  stica de refinamento.

O m  todo utilizado para se resolver um problema de PLIM vai depender do solver a qual o problema foi submetido mas, geralmente, algum m  todo enumerativo como *branch-and-bound* ou *branch-and-cut* ser   aplicado. Estes m  todos possuem uma boa efic  cia para melhorar o limitante inferior (*lower bound*) durante a explora    o dos n  os da   rvore gerada, por  m n  o possuem a caracter  stica de gerarem boas solu    es no come  o da busca. Este problema    contornado utilizando o m  todo PS.

O PS inicia a busca com uma solu    o inicial vi  vel, chamada de x' , que pode ser

$$\begin{array}{rcccccccc}
x' & = & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
x & = & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
\hline
& & 1 & + & 1 & + & 1 & + & 0 & + & 0 & + & 0 & + & 1 & + & 1 & = & 5
\end{array}$$

Figura 3.10. Exemplo de distância de Hamming entre x e x' .

obtida por alguma heurística ou outro método mais simples. Em seguida, é adicionada uma nova restrição ao problema, vista abaixo:

$$f(x) \leq f(x') - \theta \quad (3.2)$$

Esta é uma restrição de corte que garante que apenas soluções melhores que x' serão aceitas. A variável θ é chamada de variável de corte e seu valor é $\theta > 0$. Dessa forma, o valor da nova função objetivo $f(x)$ deve obrigatoriamente ser menor que a anterior.

A função objetivo é modificada para que ela possa guiar, heurísticamente, o modelo a encontrar melhores soluções que x' . A nova função objetivo, chamada de função de proximidade, calcula a distância de x' a x e é representada por $\Delta(x, x')$. A distância de Hamming é utilizada pelo PS, e a equação pode ser vista a seguir:

$$\Delta(x, x') = \sum_{j \in J: x'_j = 0} x_j + \sum_{j \in J: x'_j = 1} (1 - x_j) \quad (3.3)$$

Um exemplo deste cálculo está detalhado na Figura 3.10. Como pode ser visto, a distância $\Delta(x, x')$ é igual a 5. Isso significa que existem 5 posições diferentes entre x e x' . Com a utilização da restrição de corte, a nova função objetivo do modelo irá buscar uma solução viável que seja melhor que $f(x') - \theta$ e o mais semelhante possível a x' .

O critério de parada do PS pode ser o tempo gasto para sua execução.

3.6.1 Implementação

Três versões diferentes do PS foram apresentadas no trabalho de Fischetti & Monaci [2014]. Segundo os autores, a primeira, onde não há alteração de x' na função $\Delta(x, x')$, apresenta consideráveis desvantagens e portanto, não vale a pena ser utilizada. Na

segunda, o valor de x' é atualizado sempre que uma nova solução encontrada satisfaça a função $f(x)$. Quando isto acontece o algoritmo é reiniciado com a nova solução incumbente. Na terceira opção, são aceitas tanto soluções de melhora, quanto soluções iguais a x' , porém com uma penalidade aplicada. A versão implementada neste trabalho é a segunda, com valor de $\theta = 1$.

A solução inicial utilizada no PS é proveniente da heurística ILS descrita neste trabalho, porém o critério de parada foi alterado para a primeira solução viável encontrada. Assim, o ILS inicia suas iterações com uma solução gerada pelo **ConstrutivoAleatorio** e assim que a solução atual atender aos critérios de distância e capacidade máxima, o algoritmo termina. O pseudocódigo do PS implementado pode ser visto no algoritmo 8.

Algorithm 8 ProximitySearch

```

1:  $\theta = 1$ 
2:  $x' =$  solução gerada pelo ILS com critério de parada alterado
3: while (Tempo limite não for atingido) do
4:   Adicionar ao modelo a restrição:
5:    $f(x) \leq f(x') - \theta$  Onde  $f(x)$  é a função de proximidade  $\Delta(x, x')$ 
6:   Inicie a execução do modelo
7:    $x =$  primeira solução viável encontrada pelo modelo
8:   if ( $x \neq \emptyset$ ) then
9:     Atualizar  $\Delta(x, x')$  fazendo  $x' = x$ 
10:  end if
11: end while

```

O modelo utilizado no PS é o mesmo apresentado na definição do problema no Capítulo 3. O tempo limite de execução do PS é o mesmo utilizado para o modelo e será detalhado na seção de testes.

Capítulo 4

Geração de instâncias e calibração dos algoritmos

Para extrair todo o potencial dos algoritmos criados, uma calibração fina é necessária. Após diversos testes com configurações diferentes, é possível determinar aquela com que provavelmente a heurística apresentará o melhor resultado médio. Nesta seção, o processo de calibração de ambos os algoritmos será detalhado. Os testes foram feitos isolando cada uma das etapas para sua análise individual. Todos os testes da calibração foram executados utilizando 32 instâncias da literatura (p01 - p32), sendo p01 a p10 introduzidas por Christofides & Beasley [1984], p11 a p23 por Chao et al. [1995] e as demais por Cordeau et al. [1997]. Os valores apresentados nos gráficos representam a diferença entre o valor encontrado pela solução e o melhor valor conhecido.

O GAP entre as soluções representa a diferença relativa entre os valores comparados.

$$GAP = \frac{f_{analizado} * 100}{f_{tabu}} \quad (4.1)$$

Onde $f_{analizado}$ representa o valor da função objetivo da instância sendo analisada no momento e f_{tabu} o valor da função objetivo obtida por Cordeau et al. [1997] para esta determinada instância. Utilizamos os valores obtidos por Cordeau em seu trabalho como base para as análises de calibração. Por exemplo, para uma instância fictícia qualquer temos $f_{analizado} = 563$ e $f_{literatura} = 539$ o GAP será igual a 104,45 o que significa que $f_{analizado}$ é aproximadamente 4,45% maior que $f_{literatura}$.

Além disso, é necessário um conjunto de instâncias que faça com que seja possível uma avaliação completa dos métodos utilizados para resolução do problema. Na

literatura existem apenas 42 instâncias focadas no PRVP, as 32 citadas anteriormente mais 10 (pr01-pr10) introduzidas por Cordeau et al. [2001]. Deste modo, foram criadas mais 252 instâncias de teste para o problema. O processo de criação destas instâncias é explicado passo a passo a frente.

4.1 Instâncias geradas

Neste trabalho propomos a criação de um novo *benchmark* com 198 instâncias de pequeno porte e 54 de grande porte descritas a seguir. Para as instâncias de pequeno porte o número n de clientes varia de 10 a 30 crescendo de 2 em 2, o que totaliza onze grupos de instâncias separadas pelo número de clientes. Cada grupo possui 18 instâncias. Para as instâncias maiores, o número n de clientes varia de 50 a 100 crescendo de 25 em 25, totalizando três grupos distintos. Para cada grupo, a localização dos clientes e o depósito (pontos) foram distribuídos no plano cartesiano de dimensão 100x100 de duas formas: na metade das instâncias os pontos foram dispostos de forma totalmente aleatória, incluindo o depósito. Na outra metade o depósito foi colocado na posição central do plano (50,50) e os clientes divididos em *clusters*, sendo cada *cluster* um quadrante do plano. O número de clientes em cada quadrante foi igualmente distribuído. Por exemplo, para as instâncias com 20 clientes serão escolhidos 5 para cada quadrante. Para as instâncias em que a divisão $n/4$ não seja exata, o número de clientes em cada quadrante será o mais próximo possível.

As demandas q_i de cada cliente foram atribuídas aleatoriamente dentro do intervalo de 50 a 100. A capacidade Q_v dos veículos foi dividida em duas categorias, uma restrita e uma com folga. Nas instâncias com capacidades restritas, a soma da capacidade dos veículos é a soma total das capacidades demandadas pelos clientes mais uma pequena sobra de 1% a 5%. Na classe de instâncias com folga nas capacidades, até dois veículos a mais foram adicionados fazendo com que a soma das capacidades dos veículos chegasse ao dobro das capacidades demandadas em alguns casos. Essa diferença visa observar o comportamento dos métodos de resolução para determinar se, no caso restrito, pelo menos uma solução será encontrada e, se no caso com folga, os veículos mais dispendiosos serão excluídos do roteamento.

Os custos fixos (f_v) são proporcionais a capacidade total do veículo, o que condiz com a realidade. Quanto maior a capacidade máxima Q_v do veículo, maior seu custo fixo e vice-versa. Dessa forma, o custo fixo foi calculado como uma porcentagem sobre a capacidade do veículo, variando de 100% a 150%, aleatoriamente. Por exemplo, para um veículo com $Q_v = 250$ e com a porcentagem sobre o custo de 120%, o seu custo

fixo será $f_v = 300$. A distância máxima (T_v) de todos os veículos de uma mesma instância é idêntica. O valor de T_v é calculado somando-se a distância necessária para percorrer todos os clientes, começando do primeiro, em seguida o segundo, depois o terceiro, e assim sucessivamente até o n -ésimo cliente. Em outras palavras, é a distância necessária para percorrer todos os clientes na ordem em que foram lidos da instância. Esta distância pode ser muito superior à distância final encontrada pelos métodos utilizados para resolução das instâncias, porém é suficiente para que exista pelo menos uma rota que cubra todos os clientes. Em situações reais, esta limitação poderia ser relacionada ao tempo máximo em que o veículo pode ficar em movimento, o que geralmente é associado às horas de trabalho do motorista.

O horizonte de planejamento L determinado para todas as instâncias foi de 2, 4 e 6 dias. O número de dias de visita de cada cliente foi determinado aleatoriamente entre 1 e L . As agendas de visitas dos clientes foram escolhidas aleatoriamente dentro das possibilidades existentes para o número de dias escolhido e o tamanho do horizonte de planejamento. Para ilustrar, um horizonte de quatro dias, com três dias de visita, as opções de visitas seriam $\{1,2,3\}$, $\{2,3,4\}$, $\{1,2,4\}$, $\{1,3,4\}$ e $\{2,3,4\}$.

4.2 Calibração do ILS

A calibração do ILS pode ser dividida em quatro etapas: influência do método construtivo, influência da perturbação, influência da busca local e alteração dos valores fixos (número máximo de iterações e número de repetições sem melhora).

O número de iterações testado para o ILS foi de 100 a 10000 repetições variando de 100 em 100. Na Figura 4.1 podemos observar o valor médio da função objetivo das instâncias para cada número de iterações. Observa-se que o custo computacional da execução do método foi inversamente proporcional à qualidade das soluções obtidas. Repare que a partir de 5000 iterações, a queda na curva não é significativa fazendo com que este seja o número de repetições escolhido. Este teste utilizou a `BuscaLocalEstatica` e 20 iterações sem melhora.

Para os métodos construtivos consideramos o desvio percentual (GAP) da função objetivo no início, logo após a primeira construção, e no término, ao final de todo o algoritmo, de cada procedimento. A comparação entre os valores iniciais pode ser vista na Figura 4.2. A instância p13 foi omitida deste gráfico por apresentar um valor muito superior as demais, prejudicando assim a avaliação das linhas. Os valores finais podem ser vistos na Figura 4.3.

Como é possível observar pelas figuras, os métodos `ConstrutivoCordeau`

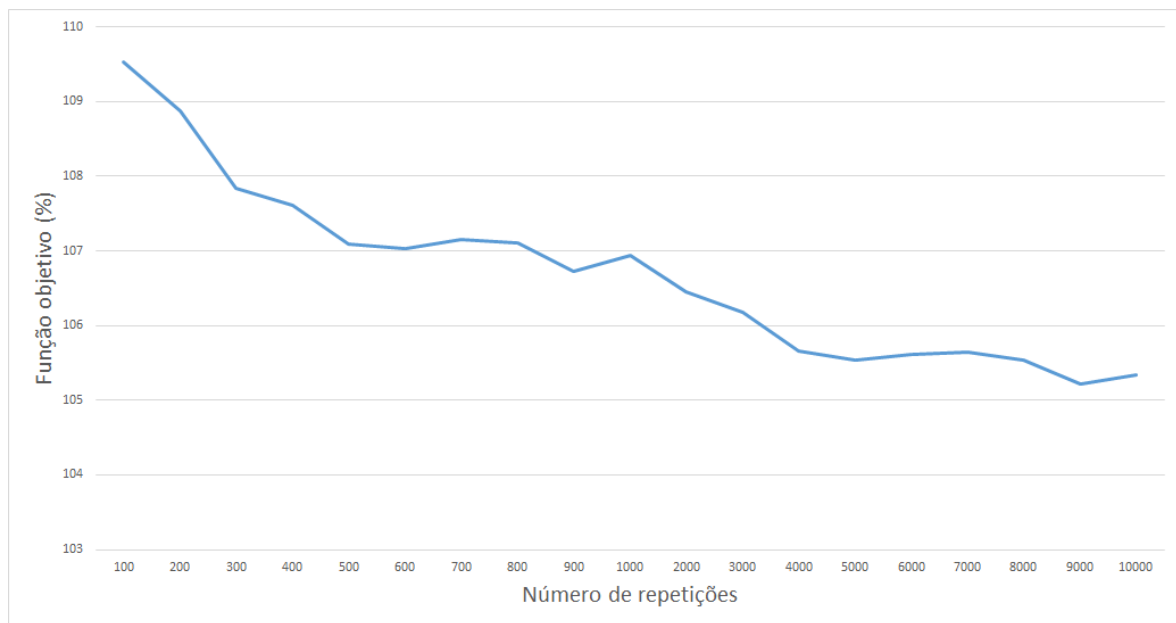


Figura 4.1. Valor médio da F.O para cada número de repetições do ILS.

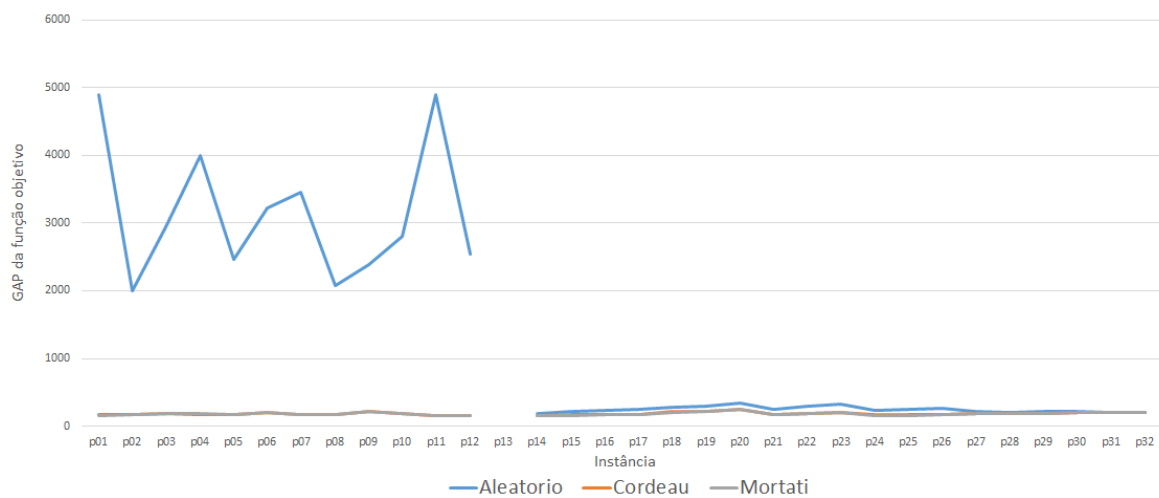


Figura 4.2. GAP entre o valor inicial dos métodos construtivos para o ILS.

e `ConstrutivoMortati` apresentam um valor inicial muito melhor que o `ConstrutivoAleatorio`, porém, ao final da execução do algoritmo, o `ConstrutivoAleatorio` tem uma ligeira vantagem em relação aos outros dois métodos.

As três perturbações também foram analisadas e seus resultados podem ser vistos na Figura 4.4. Observe que as perturbações 1 e 3 possuem um desempenho muito

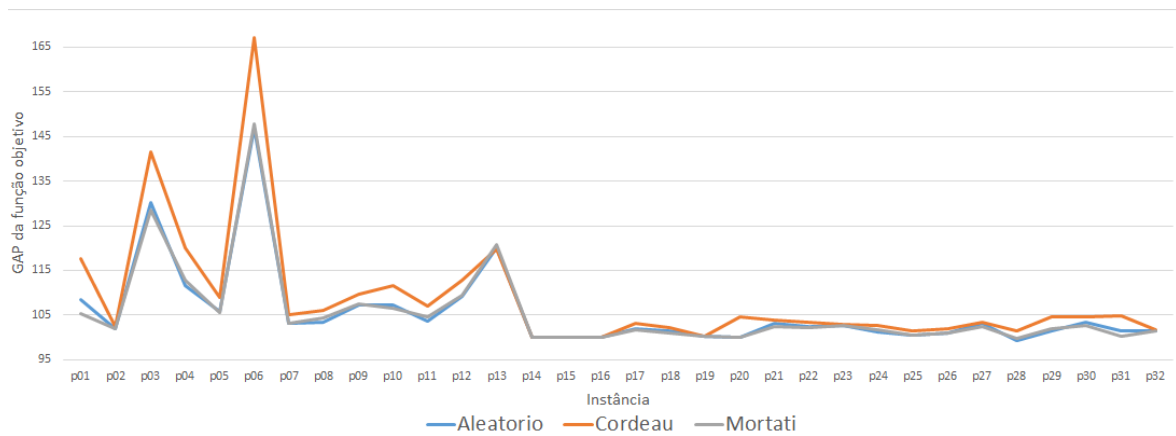


Figura 4.3. GAP entre o valor final dos métodos construtivos para o ILS.

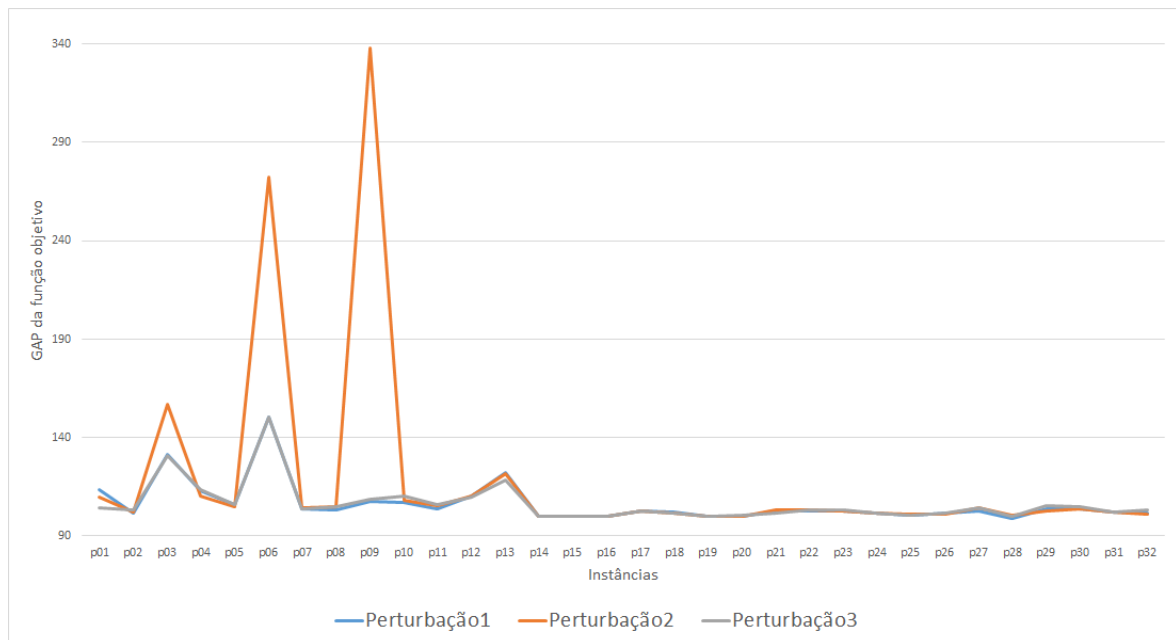


Figura 4.4. GAP entre o valor da função objetivo para cada tipo de perturbação.

semelhante, enquanto a Perturbação2 tem desempenho consideravelmente menor nas instâncias p03, p06 e p09. Estas instâncias possuem apenas um veículo disponível para o roteamento, dessa forma, a escolha de agendas válidas se torna mais difícil e a perturbação não consegue efetuar muitos movimentos úteis, o que prejudica a qualidade final da solução. Como critério de desempate, utilizamos o menor valor médio das soluções, priorizando, assim, a Perturbação1.

Para a *BuscaLocalEstatica*, foi analisado o impacto da ordem dos movimen-

tos sobre o valor final da função objetivo. Considerando as 3 buscas intra-rota e as 4 inter-rota o total de possibilidades de combinação seria $7!$, porém os testes iniciais mostraram que a execução das buscas intra-rota após a execução das inter-rota apresenta melhor resultado. Isso fez com que o número de combinações caísse para $3! + 4!$. Destas, as 10 combinações com provável melhor resultado, baseados na melhor média obtida em algumas execuções utilizando as instâncias da literatura, foram testadas, e a melhor ordem obtida foi: Inter-rota Shift(1) \rightarrow Swap(1,1) \rightarrow Radial \rightarrow Cross, seguida dos movimentos intra-rota Swap(1) \rightarrow Or-Opt(1) \rightarrow 2-Opt. Esta ordem é a mesma utilizada pelo PSO.

4.3 Calibração do PSO

A calibração do PSO pode ser dividida em três etapas: influência do método construtivo, alteração do número de partículas e repetições, porcentagem da mutação e definição dos valores utilizados na equação mestre. Salvo para o teste específico de cada etapa, a configuração padrão utilizada para execução do PSO foi: **ConstrutivoAleatorio**, **BuscaLocalEstatica**, 30 partículas, 60 repetições, probabilidade de mutação de 5%, $W=1$, $c_1=2$, $c_2=4$.

A análise dos métodos construtivos para o PSO segue a mesma linha utilizada no ILS. Como pode-se observar na Figura 4.5, o **ConstrutivoMortati** tem um pequeno ganho em relação aos outros dois para as primeiras 12 instâncias, enquanto o **ConstrutivoAleatorio** apresenta melhores resultados para as últimas. De modo geral, os três métodos apresentam resultados muito semelhantes na primeira iteração do PSO, sendo que o menor GAP foi obtido pelo **ConstrutivoMortati** com valor de 150,3. Já na Figura 4.6, os métodos apresentaram praticamente o mesmo desempenho, sendo que o **ConstrutivoCordeau** foi o que apresentou os valores mínimos, conseguindo uma média de GAP de 100,95. Analisando as duas situações, é difícil escolher qual método será utilizado, sendo que cada um deles obteve o melhor desempenho em um nicho, porém, como o algoritmo será executado com um grande número de iterações, o melhor ganho a longo prazo é mais viável, sendo assim o método escolhido é o **ConstrutivoCordeau**.

Os valores de W , c_1 e c_2 tem impacto direto sobre a qualidade da solução final, pois são eles que determinam a direção para a qual a partícula irá se mover. Dada tamanha importância todas as combinações possíveis, cuja soma dos três fatores seja no máximo 10, foram testadas. Estes resultados podem ser vistos no gráfico da Figura 4.7.

De modo geral, à medida que o valor de W aumenta, a qualidade das soluções diminui. Isso mostra que o excesso de liberdade das partículas é prejudicial ao desen-

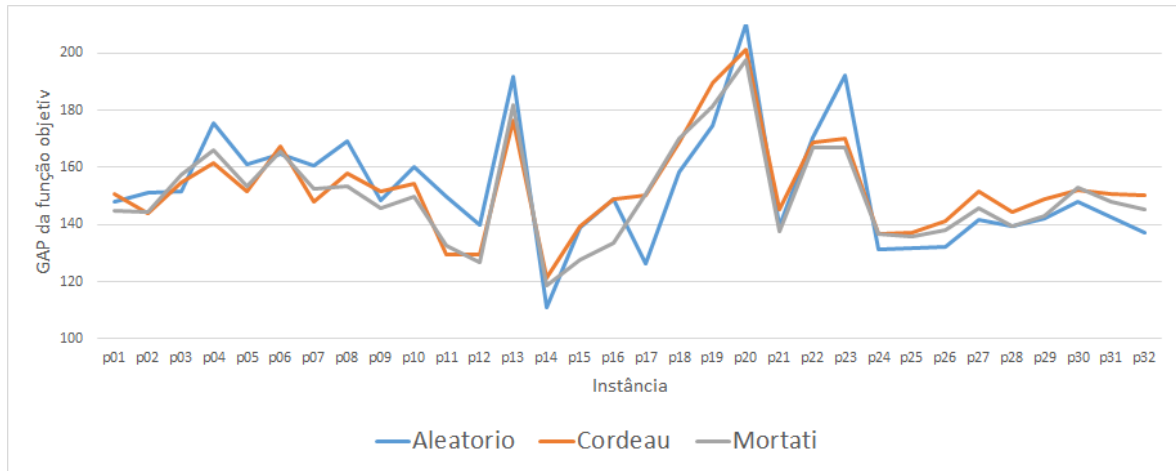


Figura 4.5. GAP entre o valor inicial dos métodos construtivos para o PSO.

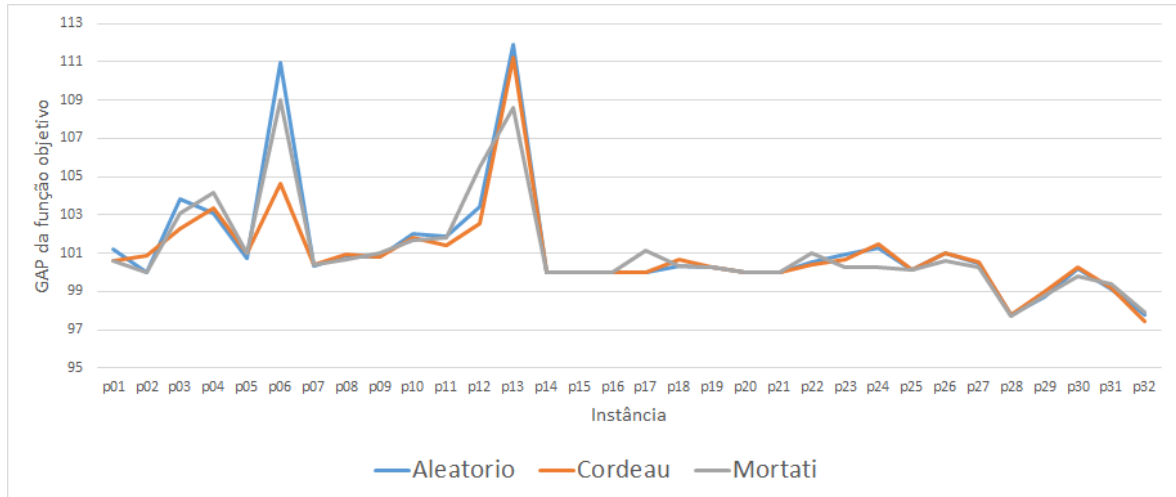


Figura 4.6. GAP entre o valor final dos métodos construtivos para o PSO.

volvimento do enxame. Quanto aos fatores c_1 e c_2 não é interessante que nenhum dos dois se distancie um do outro, sendo que os melhores resultados foram obtidos quando eles possuíam valores próximos. Por fim, a melhor combinação foi $W=1$, $c_1=2$, $c_2=4$, sendo esta a escolhida.

O número de repetições foi avaliado a partir de 1 até o máximo de 5000. O resultado pode ser visto na Figura 4.10. No gráfico da figura, o número de repetições começa em 100, pois valores anteriores a este apresentavam uma queda muito acentuada, prejudicando a visualização na figura. Como era de se esperar, quanto maior o número de repetições, melhor a qualidade das soluções encontradas, porém o tempo gasto pelo algoritmo também aumenta. Para 100 repetições, o tempo médio gasto por instância é

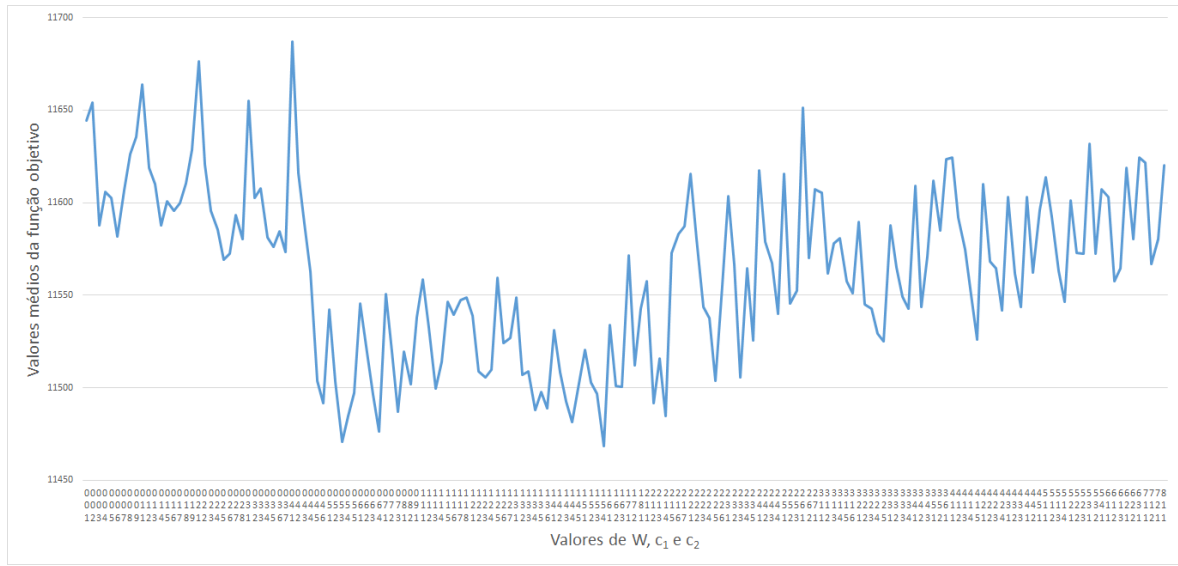


Figura 4.7. Valor médio da F.O para cada combinação de W , c_1 e c_2 .

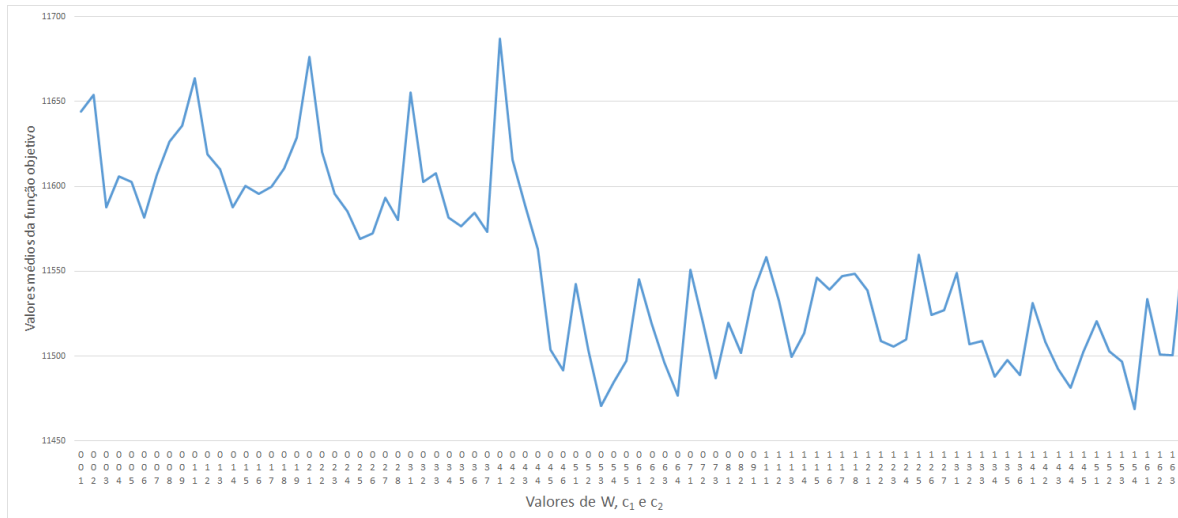


Figura 4.8. Imagem ampliada da primeira metade do gráfico da Figura 4.7 para melhor visualização.

menor que 1 segundo, enquanto que para 5000 repetições, algumas instâncias chegam a 15 segundos. Para obter o melhor ganho em relação ao custo benefício, o número de repetições escolhido foi de 1700, visto que a partir deste ponto, a curva se torna menos acentuada e a queda no valor da F.O é cada vez menos significativa.

O número de partículas foi testado variando de 20 a 500, aumentando de 10 em 10. Os resultados médios deste teste podem ser vistos na Figura 4.11. O gráfico

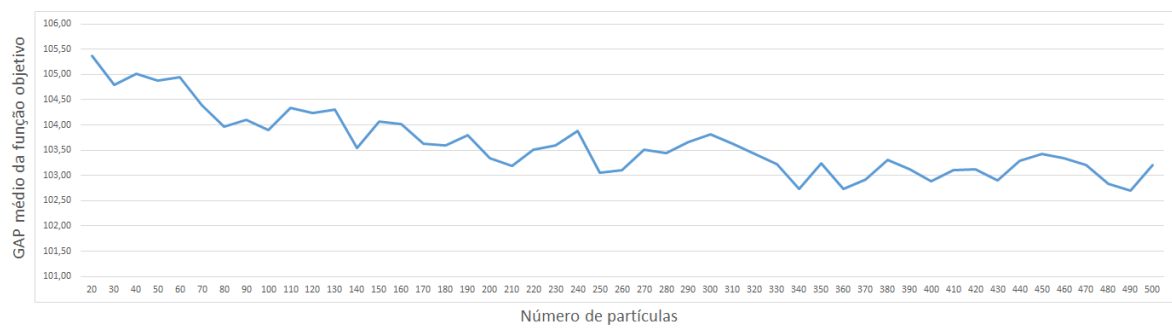


Figura 4.11. Valor médio da F.O para cada número de partículas do PSO.

Capítulo 5

Testes computacionais

Os experimentos computacionais realizados com os algoritmos propostos são relatados neste capítulo. O objetivo destes testes é determinar a qualidade dos métodos propostos frente às instâncias da literatura e às novas instâncias criadas. Os parâmetros utilizados por cada uma das heurísticas foram detalhados no capítulo de calibrações. Nas próximas seções, os resultados dos experimentos serão exibidos e discutidos.

A comparação do desempenho dos algoritmos ILS-RVND, PSO e do modelo executado no CPLEX será medida calculando o RPD (*Relative Percentage Deviation*) dos resultados, com relação à melhor solução conhecida. O RPD é determinado da seguinte maneira:

$$RPD = \frac{f_{atual} - f_{melhor}}{f_{melhor}} \times 100 \quad (5.1)$$

Em que f_{atual} é o valor da solução obtida pelo algoritmo, para uma determinada instância e f_{melhor} é o melhor resultado conhecido obtido dentre todos os algoritmos, para esta determinada instância. Cada instância foi executada uma vez pelo *solver*, tanto para o modelo puro quanto para o PS e dez vezes pelas heurísticas. A análise considera o melhor resultado das dez execuções.

5.1 Ambiente de testes

Os testes foram executados em um computador com processador Intel Core i7 4790K com 32 GB de memória RAM e sistema operacional Windows 7 (x64). Todos os métodos propostos, incluindo o modelo, foram implementados utilizando a linguagem

Tabela 5.1. Médias dos RPDs (por grupo de instâncias de 10 a 30 clientes) dos algoritmos comparados para o melhor resultado encontrado.

N Clientes	CPLEX	ILS-RVND	PSO
10	3,38	0,00	8,65
12	4,26	0,06	11,08
14	2,62	0,02	12,01
16	2,31	0,00	8,64
18	7,08	0,00	9,89
20	8,22	0,00	12,19
22	25,87	0,01	10,28
24	13,41	0,00	7,30
26	25,95	0,02	7,77
28	29,91	0,00	9,44
30	22,92	0,00	7,73
Média	13,27	0,01	9,54

C++. Apenas o modelo utilizou multiprocessamento na sua execução, através do *solver* CPLEX versão 12.6.

5.2 Avaliação das heurísticas nas instâncias geradas

O primeiro conjunto de testes analisado será em relação às novas instâncias que foram geradas. Este novo *benchmark* de instâncias foi dividido em dois grupos, como dito anteriormente, as de pequeno porte, com número de clientes variando de 10 a 30 e custo fixo associado aos veículos, e as de grande porte, com número de clientes variando de 50 a 100 e sem custo fixo. Para o primeiro grupo, as de pequeno porte, executamos o modelo e os algoritmos ILS-RVND e PSO. Para o segundo grupo, as de grande porte, incluímos a execução do algoritmo Proximity Search. Os resultados dos testes estão divididos nas seções que seguem.

5.2.1 Resultados para instâncias de pequeno porte

Para as instâncias de pequeno porte, foram executados os testes com os algoritmos ILS-RVND, PSO e o modelo matemático utilizando o *solver* CPLEX que será chamado apenas de "CPLEX" nas tabelas e figuras seguintes. O resultado compilado destes testes pode ser visto na Tabela 5.1 e na Figura 5.1.

Agrupamos os resultados por número de clientes para melhor disposição visual. Cada grupo de clientes contém a média do valor obtido por cada um dos algoritmos

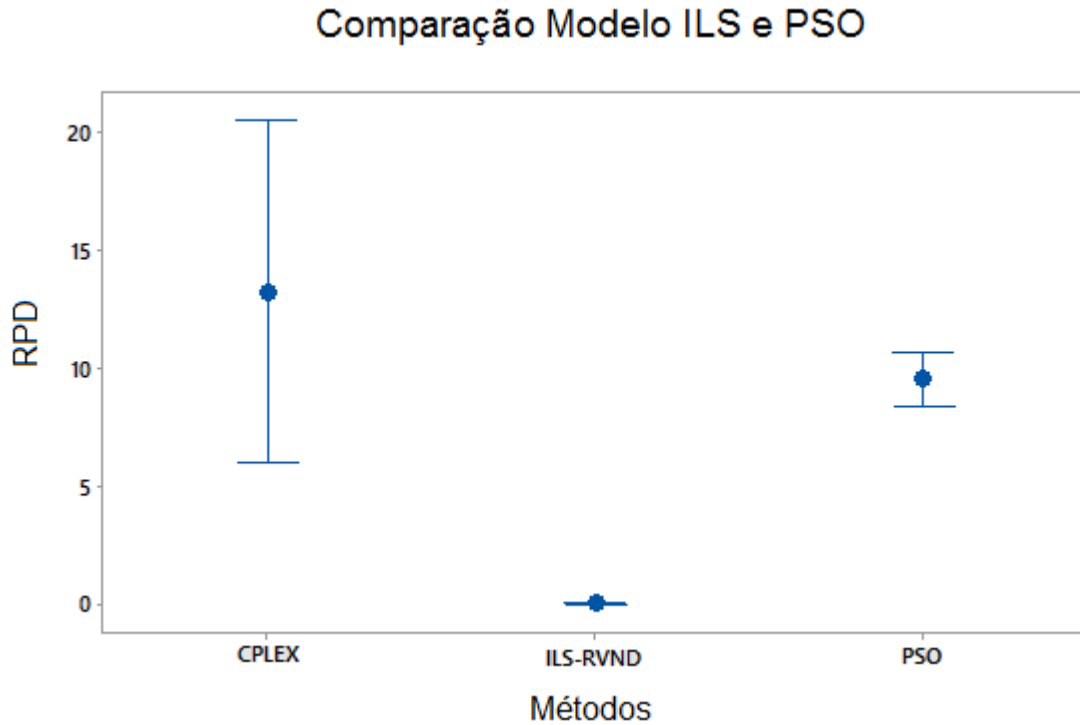


Figura 5.1. Gráfico de Médias e intervalos HSD de Tukey com nível de confiança de 95% para a comparação dos resultados encontrados pelos três métodos.

para todas as instâncias com o respectivo número de clientes. A tabela 5.1 e a figura 5.1 avaliam o desempenho do modelo do PRVP, CPLEX, e dos algoritmos ILS-RVND e PSO.

Como pode-se observar, o algoritmo que obteve melhor desempenho para esse grupo de instâncias foi o ILS-RVND. Das 198 instâncias analisadas ele conseguiu o melhor resultado em 191 delas. O PSO obteve o mesmo resultado em 28, enquanto o modelo se sobressaiu em 40. O ILS-RVND obtém resultados mais consistentes, com um menor desvio, seguido pelo PSO como pode ser observado pelo comprimento da linha no gráfico.

5.2.2 Resultados para instâncias de grande porte:

Comparação entre Proximity Search e CPLEX

Nesta seção, o Proximity Search e o CPLEX serão comparados e seus desempenhos serão avaliados. Para os dois algoritmos, o tempo limite de execução foi de uma hora e ambos foram executados utilizando o *solver* CPLEX 12.6. Para avaliar os resultados, adotamos duas métricas de comparação: denominadas como RPI (*Relative Percentage Improve*) e *primal integral*.

O RPI analisa a melhora da solução final obtida pelo método em relação a solução inicial fornecida. Em outras palavras, esta métrica avalia a porcentagem de melhoria provocada pelo método em relação a solução inicial. O RPI pode ser definido como:

$$RPI(\%) = \frac{f_i - f_{metodo}}{f_i} \times 100 \quad (5.2)$$

Onde f_i representa o valor da função objetivo da solução inicial utilizada e f_{metodo} o valor da função objetivo da solução final retornada pelo método.

O *primal integral* é uma métrica proposta por Achterberg et al. [2012] e Berthold [2013], e tem como objetivo avaliar o trade-off entre o esforço computacional gasto e a qualidade da solução encontrada. Considerando z_{opt} como a solução ótima ou a melhor solução conhecida para uma determinada instância do problema, e $z(t)$ como o valor da melhor solução encontrada até o instante de tempo t , o *primal gap function* $p(t)$, cujo valor deve estar no intervalo $[0,1]$, pode ser calculado da seguinte forma:

$$p(t) = \begin{cases} 1 & \text{Se nenhuma solução foi encontrada até o instante de tempo } t \\ \frac{z(t) - z_{opt}}{z_{opt}} & \text{caso contrário} \end{cases} \quad (5.3)$$

Após isso, o *primal integral* é calculado de $t = 0$ até t_{max} e seu valor final é calculado pela seguinte equação:

$$P(t_{max}) = \int_0^{t_{max}} p(t) \times \Delta t \quad (5.4)$$

Esta integral vai definir a área formada por $p(t) \times \Delta t$. O valor de Δt representa o intervalo de tempo gasto entre a última melhoria e a melhoria atual da solução corrente.

Com o *primal integral*, podemos medir a qualidade de diferentes métodos em relação a rapidez com que as melhorias são obtidas. Utilizando esta métrica, podemos determinar qual é o melhor método a partir do valor de $P(t)$ dentro de um intervalo de $t = 0$ até t_{max} . Quanto menor é o valor assumido por $P(t)$ melhor é o desempenho do método avaliado.

Como dito anteriormente, o método construtivo utilizado para criar a solução inicial do Proximity Search é o ILS limitado à primeira solução viável. A mesma solução encontrada por este método é também utilizada como solução de partida para

Tabela 5.2. Média de RPI por grupo de instâncias.

N Clientes	CPLEX	PS
50	6,42	14,95
75	1,87	15,47
100	0,72	14,66
Média	3,00	15,02

a execução do modelo.

Com as métricas definidas, podemos agora analisar os resultados. Na Tabela 5.2 são apresentadas as médias de RPI(%) do Modelo e do PS para cada grupo de instâncias. As instâncias foram separadas por número de clientes. Como pode ser observado, o PS obteve a maior média geral, 15,02%, alcançando uma melhora mais significativa que o modelo em relação à solução inicial dentro do limite de uma hora.

De forma complementar, foi feita uma pequena análise sobre o desempenho do PS e do modelo em relação ao tempo. A cada intervalo de tempo pré-definido é analisado o valor da função objetivo dos métodos. Quatro instâncias foram separadas como amostra para esta análise detalhada, que podem ser vistas na Figura 5.2 e dão uma ideia geral do comportamento dos algoritmos no tempo. Note que o PS consegue melhorar rapidamente o valor da função objetivo da solução inicial, e se estabiliza próximo ao final do tempo de execução. De modo geral, esse comportamento se reflete nos resultados finais encontrados. Das 54 instâncias, o PS foi superior ao modelo em todas, sendo que o modelo não conseguiu melhorar o valor da solução inicial em 21 delas e o PS em apenas duas.

Considerando também o tempo gasto para encontrar a solução final, utilizamos a métrica *primal integral* na comparação dos métodos. Para determinados períodos de tempo, definidos em 10, 50, 100, 200, 400, 800, 1300, 2000, 3000 e 3600 segundos, foi calculado o valor da integral. Quanto menor é este valor em cada intervalo de tempo, melhor é o desempenho do método avaliado. A Tabela 5.3 apresenta a média geométrica do *primal integral* das instâncias para cada intervalo.

Como podemos observar na tabela, mesmo que em intervalos específicos de tempo o PS tenha um valor ligeiramente maior que o modelo, na grande maioria seu valor é menor, o que significa que o PS consegue melhorar a solução mais rapidamente no tempo do que o modelo, comprovando assim sua eficiência.

**Figura 5.2.** Evolução dos métodos no tempo.**Tabela 5.3.** Média geométrica do primal integral por intervalo de tempo.

Tempo(s)	CPLEX	PS
10	0,48	0,49
50	2,23	1,84
100	2,57	2,24
200	5,00	2,68
400	9,50	4,15
800	11,60	6,37
1300	20,85	6,90
2000	24,75	25,49
3000	53,32	26,16
3600	392,83	69,66

5.2.3 Resultados para instâncias de grande porte:

Comparação entre todos os métodos

Nesta seção, faremos a comparação de todos os métodos propostos neste trabalho para as instâncias de grande porte. Na Tabela 5.4, podemos ver os resultados médios agrupados por número de clientes e na Figura 5.3, o gráfico referente a esta tabela.

Observando a figura, percebe-se que o desempenho geral dos métodos propostos é

Tabela 5.4. Médias dos RPDs (por grupo de instâncias de 50 a 100 clientes) dos algoritmos comparados para o melhor resultado encontrado.

N clientes	CPLEX	PS	ILS-RVND	PSO
50	147,12	58,23	49,49	42,72
75	136,98	42,97	62,67	22,95
100	108,08	31,93	94,72	14,21
Média	130,72	44,38	68,96	26,63

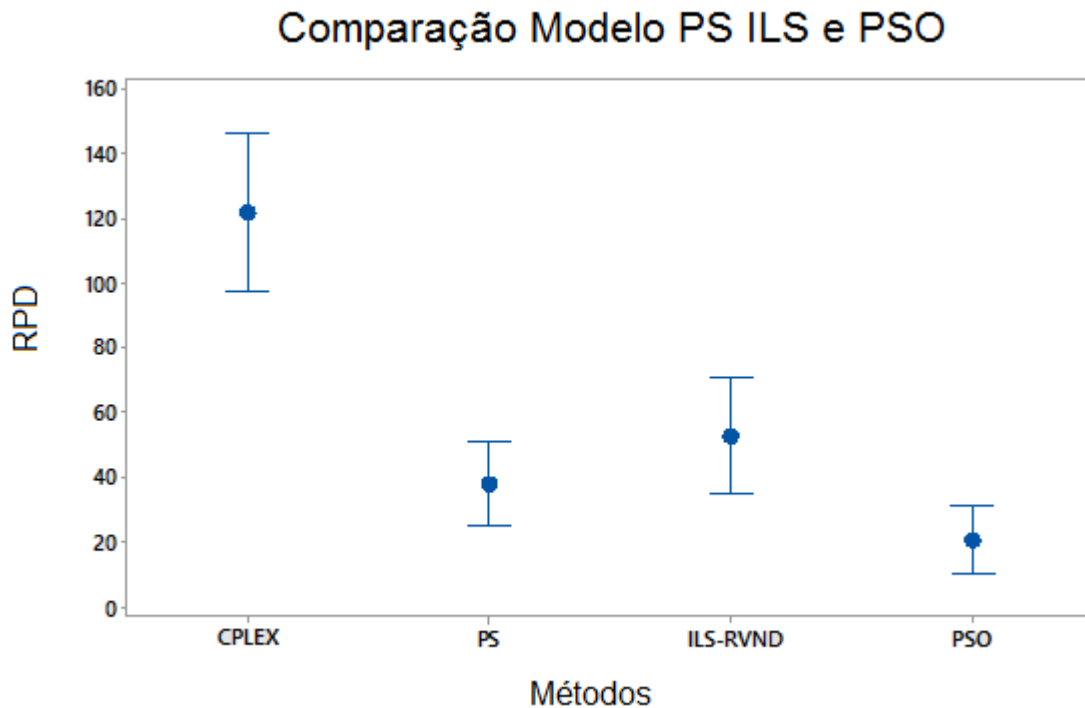


Figura 5.3. Gráfico de Médias e intervalos HSD de Tukey com nível de confiança de 95% para a comparação dos resultados encontrados pelos três métodos.

muito similar. Isso ocorre devido ao fato de que, com o aumento do número de clientes, as possibilidades de criação das rotas também crescem, o que aumenta as chances de se encontrarem boas combinações de rotas e agendas de visita. Ao mesmo tempo, essa grande variedade de opções é prejudicial ao modelo, uma vez que isto apenas aumenta o número de variáveis e, conseqüentemente, o tempo de processamento necessário para se encontrar soluções de mesma qualidade comparadas às heurísticas.

Para este segundo grupo de instâncias, o PSO se mostrou ligeiramente melhor que os demais métodos. Mesmo que o ILS-RVND tenha conseguido melhor resultado em 17 instâncias contra 14 do PSO, os valores da função objetivo encontrados pelo PSO

mantêm uma média inferior, consolidando-o como método mais estável para instâncias de maior porte.

5.3 Avaliação das heurísticas em instâncias da literatura

Utilizamos também o grupo de instâncias da literatura para comparação das heurísticas. Estas instâncias foram criadas ao longo dos anos nos trabalhos referentes a roteamento e roteamento periódico. Neste conjunto existem 32 instâncias que não possuem limite de distância para os veículos. Além disso, nesta seção, analisamos o resultado com mais 10 instâncias, pr01 a pr10, introduzidas por Cordeau et al. [2001]. Estas possuem limite de distância máxima para os veículos. O número de clientes dentre as 42 instâncias varia de 20 a 417, formando assim um grupo misto.

O modelo e o PS não encontraram solução, ou não conseguiram melhorar a solução inicial da heurística passada como parâmetro, com a limitação de tempo estabelecida. Dessa forma, eles serão desconsiderados para análise das instâncias da literatura.

A Tabela 5.5 apresenta o valor da função objetivo final da execução dos algoritmos propostos em comparação aos dois melhores algoritmos existentes na literatura atualmente, o híbrido de Set Covering e heurísticas de Cacchiani et al. [2014], chamado de SC-heur, e o algoritmo genético de Vidal et al. [2012], chamado de HGSADC. Os valores em negrito são o valor mínimo de cada instância.

Como pode ser visto na tabela, a qualidade das soluções encontradas pelos métodos da literatura é muito alta, com alguns deles sendo a solução ótima da instância. O ILS-RVND conseguiu encontrar o mesmo valor em 3 instâncias enquanto o PSO foi melhor em 4. Mesmo que as heurísticas não tenham conseguido encontrar nenhum resultado melhor que os atuais existentes, a diferença entre os valores da função objetivo é muito pequena. Na média, o ILS-RVND possui um valor 5,46% maior e o PSO apenas 3,33% maior.

Para as instâncias com limite de distância máxima para as rotas, o algoritmo de comparação utilizado foi a busca tabu proposta por Cordeau et al. [2001], já que a comparação com o HGSADC e SC-Heur não foi possível. Isso acontece pois nas análises do HGSADC os autores não disponibilizaram os resultados detalhados destas instâncias. Já o SC-Heur considerou estas instâncias para uma análise do Problema do Caixeiro Viajante Periódico, e não para o PRVP comum. Os resultados encontrados pelo PSO, ILS-RVND e pela Busca Tabu podem ser vistos na Tabela 5.6.

Tabela 5.5. Comparação do resultado da função objetivo entre as heurísticas.

Instância	HGSADC	SC-Heur	ILS-RVND	PSO
p01	524,61	524,61	546,77	546,31
p02	1322,87	1326,16	1349,63	1334,11
p03	524,61	524,61	653,59	609,84
p04	836,59	837,04	909,27	888,51
p05	2033,72	2048,29	2121,98	2070,09
p06	842,48	841,96	1166,06	1027,35
p07	827,02	834,14	849,83	831,31
p08	2022,85	2042,02	2120,22	2055,93
p09	826,94	847,03	877,96	838,03
p10	1605,22	1664,9	1702,97	1651,28
p11	775,84	790,77	815,14	802,32
p12	1195,29	1228,14	1259,4	1259,16
p13	3599,86	3549,3	4191,87	3959,99
p14	954,81	954,81	954,81	954,81
p15	1862,63	1862,63	1862,63	1862,63
p16	2875,24	2875,24	2875,24	2875,24
p17	1597,75	1597,75	1632,12	1626,93
p18	3131,09	3149,09	3194,77	3196,5
p19	4834,5	4839,87	4846,49	4846,49
p20	8367,4	8367,4	8513,71	8367,4
p21	2170,61	2170,61	2236,58	2232,46
p22	4194,23	4232,5	4391,06	4336,49
p23	6434,1	6573,33	6790,8	6750,49
p24	3687,46	3687,46	3737,83	3734,42
p25	3777,15	3777,15	3788,21	3784,79
p26	3795,32	3795,32	3842,33	3834,01
p27	21885,7	21963,23	22614,03	22216,41
p28	22272,6	22271,91	22861,61	22484,33
p29	22564,05	22564,8	23654,17	22735,74
p30	74534,38	75193,09	77998,57	76137,23
p31	76686,65	76496,67	79152,66	78694,85
p32	78168,82	78065,24	80264,06	80190,04

Tabela 5.6. Comparação do resultado da função objetivo entre as heurísticas para as instâncias com limite de distância.

Instância	Tabu	ILS-RVND	PSO
pr01	2209,02	2183,64	2193,31
pr02	3799,28	3734,68	3763,79
pr03	5218,13	5248,60	5210,70
pr04	6012,79	6125,92	6138,78
pr05	6769,80	6995,93	7012,18
pr06	8422,64	8425,81	8481,54
pr07	4997,41	4908,73	4966,99
pr08	7094,52	7351,21	7350,48
pr09	10370,45	10671,20	10729,10
pr10	13370,04	13679,00	13321,00

Como pode ser visto na tabela, o PSO conseguiu um melhor resultado em duas instâncias e o ILS-RVND em outras 3. Este resultado demonstra que, com o acréscimo da restrição do limite de distância, as heurísticas propostas mantêm um alto nível de eficiência. Além disso, esta restrição aproxima as instâncias dos problemas encontrado no mundo real. Desde que se tenha a velocidade média dos veículos, é possível converter esta distância em tempo e fazer com que o limite seja o tempo máximo que o motorista pode trabalhar.

Mesmo para as instâncias em que as heurísticas propostas não conseguiram encontrar um resultado melhor, a distância entre o valor da função objetivo é muito pequena. Nas instâncias com restrição de distância, o ILS-RVND obteve um GAP médio de apenas 101,0049, e máximo de 103,6181 para a instância pr08, ou seja, na média, o ILS obteve resultados apenas 1% maiores que os encontrados na literatura. Considerando o PSO, o GAP médio foi de 101,0678, equivalente a 1,06%, e o máximo de 103,6079 também para pr08.

Capítulo 6

Conclusões

Neste trabalho foram propostos três métodos para resolução do Problema de Roteamento de Veículos Periódico. O primeiro consiste em uma implementação do método Proximity Search aplicado às instâncias de grande porte do PRVP. Este método se mostrou eficiente especialmente se comparado a resolução do modelo puro. Neste caso, o PS consome menos tempo que o modelo para atingir o mesmo resultado. Além disso, não é necessário que a solução inicial inserida no PS seja de alta qualidade, basta que ela seja viável e mediana, para que este método consiga melhorá-la eficientemente.

As duas heurísticas propostas, ILS-RVND e PSO exibiram bons resultados tanto para as instâncias geradas quanto para as instâncias comuns da literatura. O método PSO conseguiu ser superior ao ILS-RVND em termos gerais e ambos se mostraram competitivos aos métodos mais recentes propostos por outros autores. Mesmo que os resultados não tenham sido completamente melhores eles não estão distantes. O PSO, por exemplo, pode ser útil em situações reais pois ele não possui apenas uma única solução, mas toda uma população de soluções que pode ser utilizada. Se por algum motivo um determinado trecho de uma rota específica não pode ser utilizado, basta escolher outra solução do enxame que não utilize este trecho sem que seja necessário reconfigurar e executar o algoritmo todo novamente.

O tempo de execução das heurísticas também é extremamente curto, o que é ideal para aplicações reais que requerem resposta rápida às mudanças, facilitando a tomada de decisão pelos gestores.

6.1 Trabalhos futuros

Existe um grande espaço para melhoras nos algoritmos propostos. Novos testes considerando mais de um parâmetro por vez no PSO, por exemplo, podem encontrar uma

combinação que leve a melhores resultados. Tanto no PSO quanto no ILS é possível executar uma calibração mais apurada utilizando o auxílio de testes estatísticos ou algoritmos como o Firefly de Yang [2010].

Os movimentos utilizados na busca local tem ação direta sobre a qualidade da solução, assim uma mudança ou adição de novos operadores sobre as rotas também possui chances de encontrar soluções melhores.

Ambas as heurísticas utilizam a penalidade nas soluções inviáveis. Neste trabalho uma penalidade fixa foi utilizada, porém, uma penalidade variável pode levar os métodos a explorar locais do espaço de busca inicialmente não muito atrativos mas que podem levar a resultados satisfatórios.

Mesmo as instâncias maiores ainda possuem um número de clientes pequeno se comparado a determinadas frotas reais. É possível executar as heurísticas para instâncias ainda maiores e avaliar sua viabilidade.

Referências Bibliográficas

- Achterberg, T.; Berthold, T. & Hendel, G. (2012). Rounding and propagation heuristics for mixed integer programming. Em *Operations Research Proceedings 2011*, pp. 71-76. Springer.
- Alegre, J.; Laguna, M. & Pacheco, J. (2007). Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts. *European Journal of Operational Research*, 179(3):736--746.
- Angelelli, E. & Speranza, M. G. (2002). The application of a vehicle routing model to a waste collection problem: two case studies. Em *Quantitative Approaches to Distribution Logistics and Supply Chain Management*, pp. 269--286. Springer.
- Baldacci, R. & Mingozzi, A. (2009). A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, 120(2):347--380.
- Banerjea-Brodeur, M.; Cordeau, J.-F.; Laporte, G. & Lasry, A. (1998). Scheduling linen deliveries in a large hospital. *Journal of the Operational Research Society*, 49(8):777--780.
- Baptista, S.; Oliveira, R. C. & Zúquete, E. (2002). A period vehicle routing case study. *European Journal of Operational Research*, 139(2):220--229.
- Beltrami, E. J. & Bodin, L. D. (1974). Networks and vehicle routing for municipal waste collection. *Networks*, 4(1):65--94.
- Berthold, T. (2013). Measuring the impact of primal heuristics. *Operations Research Letters*, 41(6):611--614.
- Bianchi-Aguiar, T.; Carravilla, M. A. & Oliveira, J. F. (2012). Municipal waste collection in Ponte de Lima, Portugal—A vehicle routing application. *OR insight*, 25(4):185-198.

- Cacchiani, V.; Hemmelmayr, V. & Tricoire, F. (2014). A set-covering based heuristic algorithm for the periodic vehicle routing problem. *Discrete Applied Mathematics*, 163:53--64.
- Carter, M. W.; Farvolden, J.; Laporte, G. & Xu, J. (1996). Solving an integrated logistics problem arising in grocery distribution. 34(4):290--306.
- Chao, I.; Golden, B. L.; Wasil, E. et al. (1995). An improved heuristic for the period vehicle routing problem. *Networks*, 26(1):25--44.
- Chen, Y.; Mourdjis, P.; Polack, F.; Cowling, P. & Remde, S. (2016). Evaluating hyperheuristics and local search operators for periodic routing problems. Em *European Conference on Evolutionary Computation in Combinatorial Optimization*, pp. 104--120. Springer.
- Christofides, N. & Beasley, J. E. (1984). The period routing problem. *Networks*, 14(2):237--256.
- Clarke, G. & Wright, J. V. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568--581.
- Cordeau, J.-F.; Gendreau, M. & Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, (30):105--119.
- Cordeau, J.-F.; Laporte, G.; Mercier, A. et al. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, 52(8):928--936.
- da Silva, I. C. V.; Reis, R. P. & Gomes, M. J. N. (2011). Custos e otimização de rotas no transporte de leite a latão e a granel: um estudo de caso. *Organizações Rurais & Agroindustriais*, 2(1).
- Dantzig, G. B. & Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80--91.
- de Lacerda, E. G. (2007). A otimização nuvem de partículas (particle swarm).
- do Valle, W. A. & Nogueira, T. H. (2010). Análise e redução dos custos logísticos da coleta de leite a granel pela utilização de um modelo de roteamento. *XXX Enegep*.
- Drummond, L. M.; Ochi, L. S. & Vianna, D. S. (2001). An asynchronous parallel metaheuristic for the period vehicle routing problem. *Future generation computer systems*, 17(4):379--386.

- Eberhart, R. C. & Kennedy, J. (1995). A new optimizer using particle swarm theory. Em *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pp. 39--43. New York, NY.
- Fischetti, M. & Monaci, M. (2014). Proximity search for 0-1 mixed-integer convex programming. *Journal of Heuristics*, 20(6):709--731.
- Fisher, M. L. & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109--124.
- Gary, M. R. & Johnson, D. S. (1979). Computers and intractability: A guide to the theory of np-completeness.
- Glover, F. & Laguna, M. (1997). Tabu search, 1997. *Kluwer Academic Publishers*.
- Gonçalves, L. B. (2005). Heurísticas grasp para um problema de roteamento periódico de veículos. *Acesso em: [www. ic. uff. br/PosGraduacao/Dissertacoes/274. pdf](http://www.ic.uff.br/PosGraduacao/Dissertacoes/274.pdf)*. *Access*, 4(29):2012.
- Hemmelmayr, V.; Doerner, K. F.; Hartl, R. F. & Savelsbergh, M. W. (2009a). Delivery strategies for blood products supplies. *OR spectrum*, 31(4):707--725.
- Hemmelmayr, V. C.; Doerner, K. F. & Hartl, R. F. (2009b). A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791--802.
- Higino, W.; Bezerra, H. M.; Araújo, E. J.; Poldi, K. C. & Chaves, A. A. (2012). Metaheurísticas simulated annealing e pesquisa em vizinhança variável aplicadas ao problema de roteamento periódico de veículos para coleta de lixo. *XVI CLAIO/XLIV SBPO, Rio de Janeiro*.
- Kennedy, J. (1997). The particle swarm: social adaptation of knowledge. Em *Evolutionary Computation, 1997., IEEE International Conference on*, pp. 303--308. IEEE.
- Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43(4):408--416.
- Lourenço, H. R.; Martin, O. C. & Stützle, T. (2003). *Iterated local search*. Springer.
- Mortati, C. F. (2005). *Busca Tabu aplicada ao problema de roteamento periódico de veículos*. PhD thesis, Universidade Estadual de Campinas.

- Pacheco, J.; Alvarez, A.; García, I. & Angel-Bello, F. (2012). Optimizing vehicle routes in a bakery company allowing flexibility in delivery dates. *Journal of the Operational Research Society*, 63(5):569--581.
- Pirkwieser, S. & Raidl, G. R. (2010). Matheuristics for the periodic vehicle routing problem with time windows. *Proceedings of matheuristics*, pp. 28--30.
- Russell, R. A. & Gribbin, D. (1991). A multiphase approach to the period routing problem. *Networks*, 21(7):747--765.
- Russell, R. A. & Igo, W. (1979). An assignment routing problem. *Networks*, 9:1--17.
- Tan, C. & Beasley, J. E. (1984). A heuristic algorithm for the period vehicle routing problem. *Omega*, 12(5):497--504.
- Tang, L. & Wang, X. (2006). Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem. *The International Journal of Advanced Manufacturing Technology*, 29(11-12):1246--1258.
- Toth, P. & Vigo, D. (2002). The vehicle routing problem, society for industrial and applied mathematics. *SIAM Monographs on Discrete Mathematics and Applications*.
- Vidal, T.; Crainic, T. G.; Gendreau, M.; Lahrichi, N. & Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611--624.
- Yang, X.-S. (2010). *Nature-inspired metaheuristic algorithms*. Luniver press.

Apêndice A

Publicação

A seguir está relacionado o artigo aceito e apresentado em conferência, elaborado a partir do trabalho descrito nesta dissertação:

Título: An application of ILS heuristic to Periodic Vehicle Routing Problem with heterogeneous fleet and fixed costs.

Autores: Robert C. Abreu, José Elias C. Arroyo.

Evento: Computing Conference (CLEI), 2015 Latin American

Local: Arequipa, Peru

Período: 19 a 23 de outubro de 2015