UNIVERSIDADE FEDERAL FLUMINENSE

PABLO LUIZ ARAÚJO MUNHOZ

Um algoritmo baseado em Iterated Local Search para o Problema de Roteamento de Veículos Periódico

NITERÓI

2012

UNIVERSIDADE FEDERAL FLUMINENSE

PABLO LUIZ ARAÚJO MUNHOZ

Um algoritmo baseado em Iterated Local Search para o Problema de Roteamento de Veículos Periódico

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Algoritmos e Otimização.

Orientador:
Luiz Satoru Ochi

Co-orientador:
Marcone Jamilson Freitas Souza

NITERÓI

2012

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

M966 Munhoz, Pablo Luiz Araújo

Um algoritmo baseado em Iterated Local Search para o problema de roteamento de veículos periódico / Pablo Luiz Araújo Munhoz. – Niterói, RJ : [s.n.], 2012.

69 f.

Dissertação (Mestrado em Computação) - Universidade Federal Fluminense, 2012.

Orientador: Luiz Satoru Ochi, Marcone Jamilson Freitas Souza.

1. Algoritmo heurístico. 2. Problema de roteamento de veículo. 3. Otimização combinatória (Computação). 4. Metaheurística.

CDD 005.1

Um algoritmo baseado em *Iterated Local Search* para o Problema de Roteamento de Veículos Periódico

Pablo Luiz Araújo Munhoz

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre.

Aprovada por:

Prof. D.Sc. Luiz Satoru Ochi
IC-UFF (Presidente)

Prof. D.Sc. Marcone Jamilson Freitas Souza
DECOM-UFOP

Profa. D.Sc. Simone de Lima Martins
IC-UFF

Prof. D.Sc. Lucídio dos Anjos Formiga Cabral
DI-UFPB

Niterói, 01 de março de 2012.

Aos meus pais, pela confiança e apoio.

Agradecimentos

Aos meus pais, Jorge e Sueli, por sempre estarem presentes, me apoiando e me incentivando, não somente na vida acadêmica, mas em todos os momentos da minha vida.

Ao meu irmão Igor e ao meu primo Ian, sempre com palavras de confiança e preocupados com o andamento do meu trabalho.

À minha namorada Sarah, por ter suportado a distância e também a minha ausência durante esse período do mestrado, principalmente na finalização deste trabalho. Sempre com palavras de carinho e apoio nos momentos mais difíceis.

Aos OptHousers, Igor, Sabir, Blip, Marcos e Ed, pela convivência e por tudo que passamos juntos durante esses anos.

Ao Satoru, pelos ensinamentos, conselhos e pelas oportunidades durante o Mestrado.

Ao Marcone, por ser um exemplo de pesquisador e de pessoa.

Aos meus amigos de Ouro Preto e da Rep. Calamidade Pública.

Aos colegas do IC-UFF pela ideias e ajudas.

Resumo

O Problema de Roteamento de Veículos Periódico (PRVP) é uma variante do Problema de Roteamento de Veículos Clássico (PRV) em que rotas de veículos são construídas em múltiplos dias. O PRVP pertence à classe NP-Difícil, uma vez que pode ser reduzido ao PRV quando consideramos o período de planejamento das viagens de apenas um dia. Assim, em muitos problemas-teste, métodos exatos podem não conseguir resolver o problema em um tempo computacionalmente aceitável, motivando a utilização de heurístas e metaheurísticas para sua resolução. Para resolver o PRVP, um algoritmo baseado na metaheurística *Iterated Local Search*, e tendo os métodos *Variable Neighborhood Descent* e *Pattern Improvement* como procedimentos de busca local, foi desenvolvido. O algoritmo proposto foi testado em um conjunto de problemas-teste da literatura e mostrou-se competitivo na resolução do problema.

Palavras-chave: Problema de Roteamento de Veículos Periódico, *Iterated Local Search*, *Pattern Improvement*

Abstract

The Periodic Vehicle Routing Problem (PVRP) is a variant of the Classic Vehicle Routing Problem (VRP) in which vehicle routes are built on multiple days. The PVRP belongs to the class NP-Hard, since the PVRP can be reduced to VRP when considering the planning period of only one day trips. Thus, in many instances, exact methods can not resolve the problem in a computationally feasible time, encouraging the use of heuristics and metaheuristics for its resolution. To solve the PVRP, an algorithm that combines the metaheuristic Iterated Local Search, the heuristic Variable Neighborhood Descent and Pattern Improvement method, was developed. The proposed algorithm was tested on a set of instances from the literature and proved to be competitive in solving the problem.

Key-words: Periodic Vehicle Routing Problem, Iterated Local Search, Pattern Improvement

Palavras-chave

1. Problema de Roteamento de Veículos Periódico
2. *Iterated Local Search*
3. *Variable Neighborhood Descent*
4. *Pattern Improvement*

Glossário

GPU	:	*Graphics Processing Unit*
EAD	:	Estruturas Auxiliares de Dados
ILS	:	*Iterated Local Search*;
ILS-PVND	:	*Iterated Local Search with Periodic Variable Neighborhood Descent*;
LPV	:	Lista de Padrões de Visita;
PCV	:	Problema do Caixeiro Viajante
PCVP	:	Problema do Caixeiro Viajante Periódico
POP	:	Princípio da Otimalidade Próxima
PRV	:	Problema de Roteamento de Veículos;
PRVP	:	Problema de Roteamento de Veículos Periódico;
PRVPES	:	Problema de Roteamento de Veículos Periódico com Escolha de Serviço;
PRVPFI	:	Problema de Roteamento de Veículos Periódico com Facilidades Intermediárias;
PRVPJT	:	Problema de Roteamento de Veículos Periódico com Janelas de Tempo;
PRVPMD	:	Problema de Roteamento de Veículos Periódico com Múltiplos Depósitos;
PVRP	:	*Periodic Vehicle Routing Problem*;
PVND	:	*Periodic Variable Neighborhood Descent*
RVND	:	*Variable Neighborhood Descent with random neighborhood ordering*
VND	:	*Variable Neighborhood Descent*;
VNS	:	*Variable Neighborhood Search*;
VRP	:	*Vehicle Routing Problem*;

Sumário

Lista de Figuras **xiii**

Lista de Tabelas **xiv**

Lista de Algoritmos **15**

1 Introdução **1**

1.1 O Problema de Roteamento de Veículos Periódico 1

1.2 Motivação 2

1.3 Objetivos 3

1.3.1 Objetivo Geral 3

1.3.2 Objetivos Específicos 3

1.4 Estrutura do Trabalho 3

2 Revisão Bibliográfica **4**

2.1 Introdução 4

2.2 Formulações Matemáticas da Literatura 9

2.2.1 Formulação de Christofides e Beasley [?] 10

2.2.2 Formulação de Tan and Beasley [?] 12

3 Metodologia **14**

3.1 Representação de uma solução 14

3.2 Função de Avaliação 15

3.3 Geração de uma solução inicial 17

3.3.1 Baseada na Inserção Mais Barata por Padrões 17
3.4 Estruturas de Vizinhança 21
3.4.1 Intra-rotas 21
3.4.1.1 *Exchange* 22
3.4.1.2 *1Or-opt* 22
3.4.1.3 *2Or-opt* 23
3.4.1.4 *2-opt* 23
3.4.2 Inter-rotas 24
3.4.2.1 *Swap(1,1)* 25
3.4.2.2 *Swap(2,1)* 25
3.4.2.3 *Swap(2,2)* 25
3.4.2.4 *Shift(1,0)* 26
3.4.2.5 *Shift(2,0)* 27
3.4.2.6 *Cross* 27
3.5 Buscas Locais 28
3.5.1 Estruturas Auxiliares de Dados 28
3.5.2 *Periodic* VND 30
3.5.3 *Pattern Improvement* 32
3.5.4 Busca Local utilizada no POP 34
3.6 Algoritmo Proposto ILS-PVND 34
3.7 Pré-processamento 37

4 Resultados Computacionais 38

4.1 Ambiente de desenvolvimento 38
4.2 Problemas-teste utilizados 38
4.3 Comparações com a Literatura 40
4.4 Análise dos Resultados 47

5 Conclusões e trabalhos futuros 49

Lista de Figuras

3.1 Representação de uma solução para o PRV 15

3.2 Representação de uma solução para o PRVP 15

3.3 Solução Inicial s 22

3.4 Solução s', após aplicação do *Exchange* em s 22

3.5 Solução s', após aplicação do *10r-opt* em s 23

3.6 Solução s', após aplicação do *20r-opt* em s 23

3.7 Solução s', após aplicação do *2-opt* em s 24

3.8 Solução Inicial s 24

3.9 Solução s', após aplicação do *Swap(1,1)* em s 25

3.10 Solução s', após aplicação do *Swap(2,1)* em s 26

3.11 Solução s', após aplicação do *Swap(2,2)* em s 26

3.12 Solução s', após aplicação do *Shift(1,0)* em s 27

3.13 Solução s', após aplicação do *Shift(2,0)* em s 27

3.14 Solução s', após aplicação do *Cross* em s 28

3.15 Aplicação da regra delta 30

3.16 Transformação da representação do PRVP para o PRV 37

Lista de Tabelas

1.1 Exemplo de LPV 2

4.1 Características dos Problemas-teste 39

4.2 ILS-PVND × Literatura 41

4.3 ILS-PVND × IPH - Gulczynski et al. (2011) [**?**] 43

4.4 ILS-PVND × HGSADC - Vidal et al. (2011) [**?**] 44

4.5 ILSPVND × LCG - Baldacci et al. (2011) [**?**] 45

4.6 ILSPVND × ITS - Cordeau e Maischberger (2012) [**?**] 46

Lista de Algoritmos

1 GeraçãoSoluçãoInicial(N, D, K, δ) 18

2 InserçãoMaisBarataPorPadrões($c, s_0, taxaQ$) 20

3 Inserção($c, s_0, periodo, taxaQ$) 21

4 RVND(s) 31

5 PVND(s, statusDia[ ]) 32

6 PI(s, padraoCliente[ ]) 33

7 ILS-PVND($instancia, iterMax, iterMaxILS, \delta$) 36

Capítulo 1

Introdução

O Problema de Roteamento de Veículos (PRV), referenciado na literatura como *Vehicle Routing Problem* [**?**], é um dos problemas mais estudados na área de Otimização, tanto por sua complexidade combinatória, quanto por ter diversas aplicações em problemas reais.

O PRV tem como objetivo minimizar os custos de transporte relacionados ao atendimento de um conjunto de clientes a partir de uma frota de veículos homogêneos (que possuem a mesma capacidade) localizados em um depósito ou centro de distribuição. Neste modelo básico, um conjunto de rotas, com início e término no depósito, deve ser gerado de modo que cada cliente tenha sua demanda atendida por um único veículo em uma única visita, respeitando as restrições de capacidade do veículo.

Na literatura são encontradas diversas variantes do PRV, cada uma com uma nova característica ou restrição. O Problema de Roteamento de Veículos Periódico (PRVP) é um desses modelos, e é o foco de estudo deste trabalho. O PRVP foi proposto por [**?**], para um problema de coleta de lixo, e é uma generalização do PRV clássico, onde rotas de veículos são construídas em múltiplos dias. Durante cada dia, uma frota de veículos homogêneos viaja por meio de rotas que inicializam e terminam em um único depósito.

1.1 O Problema de Roteamento de Veículos Periódico

O PRVP, diferentemente do PRV Clássico, é um modelo de roteamento de veículos onde os clientes necessitam de visitas com uma certa frequência, seja para reabastecimento de estoque, ou coleta de produtos ou resíduos. Assim, há a necessidade de definir primeiramente quais clientes serão atendidos em cada dia, para assim realizar o roteamento desses clientes escolhidos, respeitando o número de veículos disponíveis por dia de planejamento,

e também a restrição de capacidade que cada veículo possui.

O PRVP pode ser definido em um grafo completo $G = (N, A)$, com custos não negativos dos arcos conhecidos $c_{ij}, \forall (i, j) \in A$; um período de planejamento de $|D|$ dias, indexado por d; um nó depósito, indexado por $i = 0$; um conjunto de nós clientes $N_c = N \setminus \{0\}$, onde cada nó $i \in N_c$ tem uma demanda total q_i para cada dia do planejamento e requer um número fixo de visitas f_i; e um conjunto de K veículos, cada um com capacidade Q.

Cada cliente i possui uma Lista de Padrões de Visita (LPV), que representa quais as combinações de dias do período são válidas para esse cliente, respeitando o número de visitas f_i. Caso o cliente esteja em dias que não respeitem a LPV, esta solução é considerada inviável. A Tabela 1.1 ilustra um exemplo de uma LPV, onde consideramos um planejamento de $|D| = 5$ dias (Segunda à Sexta).

Tabela 1.1: Exemplo de LPV

Cliente	$\boldsymbol{f_i}$	**Padrões**
1	3	{Seg, Ter, Sex}, {Seg, Qua, Sex}
2	1	{Seg}, {Ter}, {Qua}, {Qui}, {Sex}
3	2	{Ter, Qua}, {Qua, Qui}, {Qui, Sex}

Nessa Tabela são apresentados três clientes (1, 2 e 3) que possuem uma frequência de visita de 3, 1 e 2 dias, respectivamente. Podemos observar que o primeiro cliente possui dois possíveis Padrões de Visita, podendo ser visitado na {Seg, Ter, Sex} ou {Seg, Qua, Sex}. O segundo cliente é mais flexível e pode ser visitado em qualquer dia da semana, enquanto o terceiro cliente possui três opções de Padrões de Visita, podendo ser visitado {Ter, Qua}, {Qua, Qui} ou {Qui, Sex}. A LPV é um dado de entrada do problema e qualquer outra combinação de visitas de clientes que não esteja em um dos Padrões definidos por ela é considerada inviável.

1.2 Motivação

Além de o PRVP ocorrer em uma grande gama de aplicações reais, incluindo serviços de correio, manutenção de elevadores e coleta de resíduos em empresas [**?**], o PRVP pertence à classe de problemas NP-Difícil, visto que pode ser reduzido ao PRV se considerarmos o período de planejamento de somente $|D| = 1$ dia. Assim, há também um grande desafio na resolução de forma eficiente deste problema.

1.3 Objetivos

Nesta seção são apresentados os objetivos do trabalho.

1.3.1 Objetivo Geral

Este trabalho possui como objetivo geral desenvolver um algoritmo heurístico eficiente para a resolução do Problema de Roteamento de Veículos Periódico, bem como compará-lo com outros da literatura.

1.3.2 Objetivos Específicos

Os objetivos específicos a serem atingidos são os seguintes:

(a) Realizar uma revisão bibliográfica sobre o PRVP e métodos utilizados para sua resolução.

(b) Estudar heurísticas construtivas e de refinamento, além de metaheurísticas.

(c) Desenvolver um algoritmo baseado na metaheurística *Iterated Local Search*.

(d) Avaliar o desempenho do algoritmo desenvolvido frente a outros da literatura.

1.4 Estrutura do Trabalho

O presente trabalho está dividido em cinco capítulos, incluindo esta introdução, onde o Problema de Roteamento de Veículos Periódico é contextualizado. No Capítulo 2 é apresentada uma descrição dos trabalhos encontrados na literatura tanto do PRVP, suas variantes e aplicações reais, bem como modelos de programação matemática. No Capítulo 3 é apresentada a metodologia heurística desenvolvida neste trabalho, incluindo a representação de uma solução, métodos de construção de solução inicial, e também as heurísticas aplicadas ao problema. Por fim, apresenta-se o algoritmo proposto, denominado ILS-PVND. No Capítulo 4 são apresentados os resultados obtidos pelo algoritmo comparando-o com outros da literatura. Finalmente, no Capítulo 5, são apresentadas as conclusões e trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

2.1 Introdução

O Problema de Roteamento de Veículos Periódico (PRVP), conhecido na literatura como *Periodic Vehicle Routing Problem*, tem seu foco no atendimento dos clientes dentro de um período de planejamento pré-definido, onde cada cliente pode receber uma ou mais visitas. Escolhidos os padrões de visita para cada cliente, um Problema de Roteamento de Veículos Clássico deve ser resolvido para cada dia desse período de planejamento.

O PRVP possui importantes aplicações pois pode-se trabalhar com o atendimento de uma gama de clientes diversificados, cada um com sua necessidade de atendimento. Clientes podem necessitar de mais de uma visita por dia por trabalharem com uma pequena capacidade de estocagem, ou por possuírem demandas muito altas de produtos, necessitando que a reposição seja feita com maior frequência.

O PRVP foi proposto por Beltrami e Bodin (1974) [**?**], para um caso de coleta de lixo na cidade de Nova York, mas foi formalmente definido por Russell e Igo (1979) [**?**]. O PRVP tem como objetivo selecionar um padrão de visita para cada cliente e construir a partir dos padrões escolhidos para cada cliente, rotas de atendimento visando a minimização dos custos de transporte para todos os períodos.

Na literatura são encontrados muitos problemas reais modelados como um PRVP, o que mostra sua importância e aplicabilidade. Esses problemas reais incluem coleta de lixo e produtos reciclados ([**?**], [**?**], [**?**], [**?**]), distribuição de alimentos ([**?**]) e refrigerantes ([**?**]), entrega de óleo combustível e gás industrial ([**?**]), roteamento e planejamento de técnicos de manutenção de elevadores ([**?**]), planejamento de visitas de uma empresa prestadora de serviços ([**?**]), coleta de matérias-primas para um fabricante de autopeças

([?]), planejamento de extração de petróleo de poços terrestres ([?]), coleta de leite ([?]), entre outros.

Os trabalhos mais recentes na literatura para a resolução do PRVP utilizam métodos baseados em heurísticas e metaheurísticas. Os principais trabalhos serão apresentados brevemente a seguir.

Em [?] são propostas duas heurísticas de construção e uma de refinamento. Uma das heurísticas construtivas é baseada no Algoritmo de Economias de Clarke e Wright (*Clarke e Wright's Savings Algorithm* [?]), com modificações para que o método somente gere soluções viáveis. A segunda heurística de construção é feita em duas fases. Na primeira, *clusters* de rotas são criadas a partir de clientes que possuem um padrão de visita fixo, ou seja, $|LPV| = 1$. A partir desses *clusters* iniciais gerados na primeira fase, os demais clientes são inseridos na ordem decrescente da frequência de visitas (f_i). É proposta, então, uma heurística de refinamento que otimiza as rotas e também os períodos da solução. Esse procedimento de refinamento é baseado na heurística MTOUR para o PRV ([?]).

Em [?] é introduzida uma definição de distribuição dos custos, e uma heurística baseada em trocas é utilizada para minimizar os custos dessa distribuição para o problema. A distribuição dos custos é representada pela troca do PRV vinculado a cada dia do PRVP por um Problema da P-Mediana [?], ou por um Problema do Caixeiro Viajante (PCV).

Em [?] um modelo matemático é criado para o PRVP. Os autores estendem uma formulação proposta por [?] para o PRV Clássico utilizando o modelo para o problema de atribuir clientes a períodos. São propostas relaxações para esse modelo, e um algoritmo heurístico é utilizado para arredondar os valores não-inteiros gerados pelo modelo relaxado.

Em [?] é proposto um algoritmo de 4 fases. Uma fase de construção de solução inicial baseada em generalização por redes, duas fases heurísticas, uma levando em consideração a transformação do PRVP em vários PCVs, e outra explorando características do PRVP, alterando rotas entre períodos distintos. Em uma quarta fase, um modelo inteiro binário é utilizado buscando melhorar as soluções geradas pelas três fases antecessoras.

Os autores em [?] trabalharam com dois objetivos para o PRVP. Além do que comumente é utilizado, minimizar os custos de transporte envolvidos em todos os períodos, eles visam também a minimização da frota de veículos utilizada. Além das restrições de capacidade máxima de cada veículo, foram adicionadas restrições de duração máxima da rota, e alteradas as restrições de visitas de clientes por período. No modelo clássico do PRVP, cada cliente é vinculado a uma Lista de Padrões de Visita (LPV), enquanto no

modelo tratado por esses autores, são definidos um número mínimo e um número máximo de dias entre cada visita para o mesmo cliente.

Em [**?**], os autores utilizam a formulação de [**?**] para gerar soluções iniciais viáveis. Após esse passo, a solução encontrada é enviada para o procedimento heurístico *Record-to-Record* em uma fase de refinamento. Nesse artigo, os autores também descrevem métodos para melhorar a heurística utilizando relaxação de capacidade dos veículos e pós-processamento.

O PRVP é resolvido em [**?**] por uma abordagem baseada na metaheurística Busca Tabu. Esse método também é utilizado pelos autores para resolver o Problema de Roteamento de Veículos Periódico com Multiplos Depósitos (*Multi-depot Periodic Vehicle Routing Problem*) e o Problema do Caixeiro Viajante Periódico (PCVP)(*Periodic Traveling Salesman Problem*). Os movimentos utilizados consistem em mover um cliente de uma rota para outra dentro do mesmo período, ou atribuir um novo padrão de visita a um cliente. Tanto a inserção quanto a remoção dos clientes é feita utilizando o algoritmo GENI (inserção de menor custo) proposto por [**?**]. Um mecanismo de diversificação também é utilizado pelos autores, são adicionadas penalidades nas soluções que possuem movimentos que frequentemente foram utilizados, buscando, assim, uma melhor exploração do espaço de soluções.

Em [**?**] foi utilizado um algoritmo evolutivo paralelo baseado na metaheurística Algoritmos Genéticos. Para paralelizar o algoritmo usou-se o Modelo de Ilha, onde cada processo recebe uma subpopulação da população total. O mecanismo de reprodução usa *crossover* e mutações clássicos, e como critério de diversificação da população, realizaram-se trocas de soluções entre as subpopulações. Além dessas características, o algoritmo desenvolvido pelos autores ainda faz o uso de uma busca local aplicada a cada período que utiliza conceitos do Algoritmo de Economias de Clarke e Wright.

Os autores em [**?**] desenvolveram um algoritmo evolutivo de duas fases utilizando conceitos da metaheurística *Scatter Search* [**?**]. Para representar uma solução do problema, eles utilizam uma lista contendo o padrão p, $\forall p \in LPV$, escolhido para cada um dos clientes. Na primeira fase, é escolhida uma opção da LPV para cada cliente. Na segunda fase rotas são construídas para cada dia com base nessa escolha prévia. Como busca local são utilizadas as estruturas de vizinhança *Cross* e *Or-opt*.

Em [**?**] uma nova heurística baseada na metaheurística *Variable Neighborhood Search* (VNS) [**?**] foi proposta para resolver tanto o PRVP quanto o Problema do Caixeiro Viajante Periódico. Para a construção de uma solução inicial foi utilizado o Algoritmo de

Economias de Clarke e Wright. Como busca local utilizou-se o *3-Opt*, e investiu-se em várias perturbações, visando explorar melhor o espaço de soluções do problema. Para as perturbações foram utilizadas as estruturas de vizinhança *Or-1*, *Or-2*, *Or-3*, *Cross* e troca de Padrões de Visitas de clientes. Além disso, foi utilizado um critério para aceitação de soluções de piora baseado na metaheurística *Simulated Annealing* (SA).

Em [**?**], os autores, além de tratarem o PRVP Clássico, consideraram também duas variantes até então não exploradas na litetatura. A primeira visa melhorar as soluções enquanto restrige a quantidade de rompimentos causados nas rotas pelas novas atribuições de padrões aos clientes, denominada PRVP com Restrições de Reatribuição (*Period Vehicle Routing Problem with Reassignment Constraints* – PVRP-RC). A segunda visa melhorar as soluções mas mantendo um balanceamento na carga de trabalho dos motoristas, denominada PRVP com Restrições de Balanceamento (*Period Vehicle Routing Problem with Balance Constraints – PVRP-BC)*. Para tratar esses problemas, foi desenvolvida uma heurística híbrida que utiliza programação inteira e um Algoritmo *Record-to-Record* Aprimorado [**?**].

O trabalho desenvolvido em [**?**] foi o primeiro a utilizar métodos de buscas locais mais agressivos visando a melhoria dos resultados da heurística. Foi proposta uma adaptação da metaheurística Algoritmos Genéticos, utilizando buscas locais e um mecanismo de diversificação adaptativo da população. O método trabalha com duas subpopulações, uma de soluções viáveis e outra de inviáveis. A solução é representada como uma grande rota, sem delimitadores, ficando a cargo do algoritmo achar as rotas ótimas para cada solução. Essa representação é justificada pelos autores por facilitar a utilização de operadores *crossover* simples e eficientes. Após os cruzamentos, os novos indivíduos gerados são levados a um processo de busca local, definido pelos autores como uma fase de educação. Essa fase é voltada para otimizar cada PRV presente em cada período. Além de utilizar nove estruturas de vizinhança clássicas para o PRV, é definida uma nova estrutura de vizinhança específica para o PRVP, denominada *Pattern Improvement*.

Um algoritmo exato foi desenvolvido em [**?**]. Foi utilizado um *framework* matemático proposto por [**?**], que trabalha em 3 etapas: (i) computa uma solução dual, próxima à ótima, de uma formulação de Programação Linear Relaxada fortalecida por desigualdades válidas; (ii) utiliza essa solução dual para gerar um Problema Inteiro Reduzido contendo todas as soluções ótimas; (iii) resolve o Problema resultante utilizando um resolvedor matemático. Assim, foi feita uma formulação de Particionamento de Conjuntos (*Set Partitioning*) para o PRVP, e propostas três relaxações para essa formulação.

Em [**?**] foi proposta um abordagem paralela utilizando conceitos das metaheurística *Iterated Local Search* (ILS) e Busca Tabu, denominada *Parallel Iterated Tabu Search*. Após a geração de uma solução inicial utilizando o algoritmo GENI, o ILS realiza melhorias na solução baseando-se na alternância entre duas fases, uma de busca local e outra de perturbação da solução. Como busca local foi utilizada uma variação da heurística Busca Tabu proposta por [**?**], onde uma busca local intra-rota é acionada de tempos em tempos. Como mecanismo de perturbação, foi utilizada uma heurística de remoção de *clusters* de clientes, que reconstrói parte de uma rota. Esse algoritmo foi paralelizado, e para aproveitar características das soluções de cada processo, após a perturbação, um mecanismo de *crossover* entre soluções de processos diferentes é chamado com probabilidade de 10%.

Além do tratamento do modelo de PRVP Clássico, algumas variantes também são encontradas na literatura. Dentre essas variantes, podemos destacar quatro modelos. Um desses modelos é o Problema de Roteamento de Veículos Periódico com Múltiplos Depósitos (PRVPMD) (*Multi-depot Periodic Vehicle Routing Problem*), proposto por [**?**], onde mais uma escolha deve ser feita. Além de definir em quais dias os clientes serão atendidos, deve-se escolher também a partir de qual depósito cada veículo atenderá as demandas dos clientes para cada dia.

Uma variante muito similar ao PRVPMD é o Problema de Roteamento de Veículos com Facilidades Intermediárias (PRVPFI) (*Periodic Vehicle Routing Problem with Intermediate Facilities*), proposta por [**?**]. Nesta variante, não há múltiplos depósitos, porém é utilizada a ideia de "Pontos de Recarga", ou Facilidades Intermediárias, onde o veículo pode parar no meio do seu percurso para reabastecimento, permitindo, assim, que sua capacidade seja renovada e mais clientes sejam atendidos em uma mesma rota.

Uma outra variante é o Problema de Roteamento de Veículos Periódico com Janelas de Tempo (PRVPJT) (*Periodic Vehicle Routing Problem with Time Windows*), proposto por [**?**]. Nessa variante, além da frequência de visita, cada cliente possui um intervalo de atendimento. Caso o atendimento ao cliente ocorra fora desse intervalo, uma penalização é aplicada. O objetivo dessa variante é reduzir os custos de transporte obedecendo o intervalo de atendimento de todos os clientes.

Uma generalização do PRVP foi proposta por [**?**], definida como Problema de Roteamento de Veículos Periódico com Escolha de Serviço (PRVPES) (*Periodic Vehicle Routing Problem with Service Choice*). Nesta generalização, a frequência de visita aos clientes é uma decisão do problema. Assim, a dificuldade do problema é aumentada de duas formas: primeiro, há a complexidade de definir a frequência de visita de cada cliente, e em

segundo, pela exigência de capacidade do veículo ao visitar um cliente também se tornar uma decisão do modelo.

2.2 Formulações Matemáticas da Literatura

A resolução de forma totalmente exata para o PRVP foi pouco explorada na Literatura, e a maioria dos modelos propostos não tratam o problema diretamente, tratando primeiramente a definição de quais padrões de visita dos clientes serão utilizados, isto é, o Problema da Atribuição, para depois realizar o roteamento com os clientes já fixados em seus dias.

Os modelos recebem os seguintes dados de entrada:

- c_{ij}: custos de transporte entre os clientes i e j
- D: conjunto dos dias de visita do período de planejamento
- A: conjunto de arestas (i,j)
- K: conjunto dos veículos disponíveis
- N: conjunto dos clientes

Nos modelos, consideraremos também, uma *agenda* como uma coleção de dias dentro do período de planejamento em que nós são atendidos. Alocar um nó para uma *agenda* implica que o nó será atendido em todos os dias dessa *agenda*. Com isso, definimos um conjunto S com todas as possíveis *agendas*, e as indexamos por $s \in S$. Cada *agenda* em S é totalmente descrita pelo vetor binário a_{sd}, apresentado a seguir:

$$a_{sd} = \begin{cases} 1, \text{ Se o dia } d \in D \text{ pertence à } \textit{agenda } s \in S \\ 0, \text{ Caso contrário} \end{cases}$$

Além disso, no PRVP cada nó requer um número fixo de visitas durante o período de planejamento, definido por f_i. Por isso, para cada nó $i \in N_c$, uma *agenda* dentre o subconjunto não vazio de possíveis agendas candidatas $S_i \subseteq S$ deve ser escolhida. Esse subconjunto é definido como:

$$S_i = \{s \in S : \sum_{d \in D} a_{sd} = f_i\}$$

Nota-se que se $|S_i| = 0$ para qualquer $i \in N_c$, não há solução viável para o problema, pois nenhuma *agenda* consegue satisfazer o requisito de número de visitas do cliente i. Outra observação é que se $|S_i| = 1$, $\forall i \in N_c$, cada nó só possui uma única *agenda* que satisfaça ao requisito de número de visitas. Nesse caso, a alocação exata dos nós nas *agendas* é conhecida, e o problema pode ser decomposto em $|D|$ Problemas de Roteamento de Veículos Clássicos.

As variáveis de decisão que são comumente utilizadas em várias formulações são:

$$x_{ijk}^{d} = \begin{cases} 1, \text{ Se o veículo } k \in K \text{ percorre o arco } (i,j) \text{ no dia } d \in D \\ 0, \text{ Caso contrário} \end{cases} \tag{2.1}$$

$$y_{ik}^{s} = \begin{cases} 1, \text{ Se o veículo } k \in K \text{ visita o nó } i \in N_c \text{ na } \textit{agenda } s \in S \\ 0, \text{ Caso contrário} \end{cases} \tag{2.2}$$

Algumas formulações utilizam valores agredados das variáveis apresentadas acima:

$$\tilde{x}_{ik}^{d} = \sum_{j \in N} x_{ijk}^{d} = \begin{cases} 1, \text{ Se o veículo } k \in K \text{ visita o nó } i \in N_c \text{ no dia } d \in D \\ 0, \text{ Caso contrário} \end{cases} \tag{2.3}$$

$$z_{i}^{s} = \sum_{k \in K} y_{ik}^{s} = \begin{cases} 1, \text{ Se o nó } i \in N_c \text{ é visitado na } \textit{agenda } s \in S \\ 0, \text{ Caso contrário} \end{cases} \tag{2.4}$$

Em resumo, dois pontos de vista surgiram na definição do PRVP: em [**?**] os autores formularam o PRVP como um problema de roteamento com uma decisão de seleção de dias envolvida. Por outro lado, em [**?**] foi utilizada uma abordagem para resolver o PRVP como uma extensão do problema de atribuição que possui um componente de roteamento. Nesta Seção serão apresentados os modelos para tratar o PRVP desses dois pontos de vista.

2.2.1 Formulação de Christofides e Beasley [?]

Em [**?**], os autores apresentaram a primeira formulação matemática para o PRVP. Eles definiram o PRVP como um problema de designar um conjunto de rotas para cada dia de um período de planejamento de $|D|$ dias, atendendo a frequência de visita de cada um dos clientes. Para isso, utilizaram dois conjuntos de variáveis de decisão, um para alocar clientes à *agendas*, e outro para o roteamento de um dado veículo em um certo dia. São utilizadas três variáveis de decisão: x_{ijk}^{d}, definida na Eq. (2.1), z_{i}^{s}, definida na

Eq. (2.4), e uma variável de decisão binária v_i^d, que recebe o valor 1 caso o cliente $i \in N_c$ é visitado no dia $d \in D$, e o valor 0, caso contrário. Além disso, restrições adicionais foram adicionadas referentes à limitação do tamanho máximo da rota, seja em distância ou tempo, essa limitação é definida como L.

A formulação de Christofides e Beasley [**?**] para o PRVP é apresentada a seguir.

$$\min \sum_{d\in D}\sum_{(i,j)\in A}\sum_{k\in K} c_{ij}x_{ijk}^d \tag{2.5}$$

$$s.a.$$

$$\sum_{s\in S_i} z_i^s = 1 \qquad \forall i \in N_c \tag{2.6}$$

$$v_i^d = \sum_{s\in S_i} z_i^s a_{sd} \qquad \forall d \in D; i \in N_c \tag{2.7}$$

$$\sum_{k\in K} x_{ijk}^d \leq \frac{v_i^d + v_j^d}{2} \qquad \forall d \in D; i,j \in N_c (i \neq j) \tag{2.8}$$

$$\sum_{k\in K} x_{ijk}^d = \sum_{j\in N_c} x_{jik}^d \qquad \forall i \in N; k \in K; d \in D \tag{2.9}$$

$$\sum_{k\in K}\sum_{i\in N} x_{ijk}^d = \begin{cases} v_j^d, & \forall j \in N_c \\ |K|, & j = 0 \end{cases} \qquad \forall d \in D \tag{2.10}$$

$$\sum_{i,j\in R} x_{ijk}^k \leq |R| - 1 \qquad \forall R \subseteq N_c; k \in K; d \in D \tag{2.11}$$

$$\sum_{j\in N_c} x_{0jk}^d \leq 1 \qquad \forall k \in K; d \in D \tag{2.12}$$

$$\sum_{i\in N_c} q_i \sum_{j\in N} x_{ijk}^d \leq Q \qquad \forall k \in K; d \in D \tag{2.13}$$

$$\sum_{(i,j)\in A} c_{ij}x_{ijk}^d \leq L \qquad \forall k \in K; d \in D \tag{2.14}$$

$$z_i^s \in \{0,1\} \qquad \forall i \in N_c; s \in Si \tag{2.15}$$

$$x_{ijk}^d \in \{0,1\} \qquad \forall (i,j) \in A; k \in K; d \in D \tag{2.16}$$

A função objetivo (2.5) minimiza os custos de viagens entre os clientes. As restrições (2.6) garantem que uma *agenda* viável seja escolhida para cada nó, enquanto as restrições (2.7) definem v_i^d para os dias da *agenda* escolhida. As restrições (2.8) permitem somente a ligação de clientes que foram alocados para o dia $d \in D$. As restrições (2.9) são as de conservação de fluxo. Restrições (2.10) garantem que nós somente sejam incluídos em rotas para os dias dentro da sua *agenda* atribuída. As restrições (2.11) são as de

eliminação de subrotas. As restrições (2.12) garantem que um veículo não seja utilizado mais que uma vez por dia. As equações (2.13) e (2.14) são as restrições de capacidade do veículo e de tamanho máximo da rota, respectivamente. Finalmente, as restrições (2.15) e (2.16) definem a integralidade das variáveis.

2.2.2 Formulação de Tan and Beasley [?]

Para a resolução do PRVP, [?] definiram uma medida de custo θ_{ikd} que indica a distância ou o custo de visitar o cliente $i \in N_c$ com o veículo $k \in K$ no dia $d \in D$, considerando que esse cliente i faz parte de uma *agenda* $S_i \in S$. Nesta formulação, eles utilizam as variáveis de decisão $\tilde{x}_{ik}^d$, definida na Eq. (2.3), e z_i^s, definida na Eq. (2.4).

A formulação de Tan and Beasley [?] para o PRVP é apresentada a seguir.

$$\min \sum_{i \in N_c} \sum_{k \in K} \sum_{d \in D} \theta_{ikd} \tilde{x}_{ik}^d \tag{2.17}$$

$$s.a.$$

$$\sum_{s \in S_i} z_i^s = 1 \qquad \forall i \in N_c \tag{2.18}$$

$$\sum_{k \in K} \tilde{x}_{ik}^d = \sum_{s \in S_i} a_{sd} z_i^s \qquad \forall i \in N_c; d \in D \tag{2.19}$$

$$\sum_{i \in N_c} q_i \tilde{x}_{ik}^d \leq Q \qquad \forall k \in K; d \in D \tag{2.20}$$

$$\tilde{x}_{ik}^d \in \{0,1\} \qquad \forall i \in N_c; k \in K; d \in D \tag{2.21}$$

$$z_i^s \in \{0,1\} \qquad \forall i \in N_c; s \in S_i \tag{2.22}$$

A função objetivo (2.17) minimiza o custo do serviço especificado por θ_{ikd}. As restrições (2.18) atribuem cada nó a uma *agenda*. Restrições (2.19) garantem que os veículos são roteados no dia apropriado para visitar a *agenda* correspondente. As restrições (2.20) garantem que as atribuições de nós aos veículos não violem as restrições de capacidade. As restrições (2.21) e (2.22) definem as variáveis binárias de atribuição. Essa formulação apresentada descreve um problema que é claramente mais complexo do que uma atribuição de VRP em múltiplos dias, já que as restrições (2.18) e (2.19) não permitem que o problema seja decomposto em dias.

Observando essa dificuldade em resolver esse problema, Tan e Beasley [?] sugeriram que as atribuições dos nós aos veículos pudessem ser negligenciadas para reduzir o tamanho

do problema. Eles então decidiram trabalhar com uma formulação de duas fases, onde na primeira fase seria tomada a decisão de alocação dos nós aos dias; e, na segunda fase, o roteamento para cada dia seria resolvido. Eles propuseram uma medida de custo agregada, Θ_i^d , que representa a contribuição da distância de visita do nó $i \in N_c$, onde é considerada a *agenda* S_i, em *qualquer* rota que envolva o cliente i no dia $d \in D$. Os autores resolvem o problema da atribuição adicionando nós aos dias e assegurando que a demanda total atendida em cada dia não ultrapasse a capacidade total de todos os veículos para aquele dia ($Q \times |K|$). O objetivo do problema de atribuição é minimizar os custos de transporte entre os nós. A formulação modificada é apresentada a seguir.

$$\min \sum_{i \in N_c} \sum_{d \in D} \sum_{s \in S_i} \Theta_i^d a_{sd} z_i^s \tag{2.23}$$

$$s.a.$$

$$\sum_{s \in S_i} z_i^s = 1 \qquad \forall i \in N_c \tag{2.24}$$

$$\sum_{i \in N_c} \sum_{s \in S_i} q_i a_{sd} z_i^s \leq Q \times |K| \qquad \forall d \in D \tag{2.25}$$

$$z_i^s \in \{0, 1\} \qquad \forall i \in N_c; s \in S_i \tag{2.26}$$

A função objetivo (2.23) minimiza o custo de roteamento segundo a medida dada por Θ_i^d. As restrições (2.24) atribuem cada nó a uma *agenda*. As restrições (2.25) garantem que as atribuições não excedem a capacidade total de todos os veículos em qualquer dia. As restrições (2.26) definem as variáveis de atribuição como binárias. Após essa fase, são resolvidos $|D|$ PRV independentes para cada dia $d \in D$. Esse método depende fortemente de avaliar a contribuição da matriz Θ_i^d corretamente. Infelizmente, para avaliar com precisão essa matriz Θ_i^d é necessário resolver todos os PRVs para todas as combinações entre rotas e dias para cada nó $i \in N_c$.

Capítulo 3

Metodologia

Neste trabalho é proposto um algoritmo heurístico, denominado ILS-PVND, para a resolução do PRVP. Este algoritmo é baseado na metaheurística *Iterated Local Search*, e nos métodos *Variable Neighborhood Descent*(VND) e *Pattern Improvement*. A combinação ILS+VND foi usada pois há aplicações bem sucedidas dessa metodologia em variantes do PRV, como por exemplo, em [**?**], [**?**] e [**?**]. Além disso, trata-se de uma metaheurística ainda pouco explorada em Problemas de Roteamento de Veículos. Neste capítulo são descritos os componentes do algoritmo ILS-PVND proposto.

3.1 Representação de uma solução

Uma solução para o PRV é representada por uma combinação de clientes, numerados de 1 a n, e separados em k partições, com k representando o número de veículos utilizados. Assim, em um problema com $n = 15$ clientes e $k = 3$ veículos, uma possível solução pode ser representada por: [[2 , 7 , 12 , 10 , 15 , 4] , [14 , 5 , 1 , 9] , [6 , 13 , 8 , 11 , 3]]. Em cada uma das partições, o primeiro elemento da lista representa o primeiro cliente a ser visitado pelo veículo a partir do depósito. A ordem de visitas dos demais clientes segue a ordem dos elementos da lista. Chegando ao último elemento, o veículo retorna ao depósito. Uma representação gráfica dessa solução pode ser vista na Figura 3.1.

Para representar uma solução do PRVP, adiciona-se mais uma dimensão, referente ao período de planejamento de $|D|$ dias, à representação do PRV. Adicionando-se então essa nova dimensão, os clientes são distribuídos entre os dias, de acordo com suas Listas de Padrões de Visita (LPV), e em cada dia tem-se um PRV.

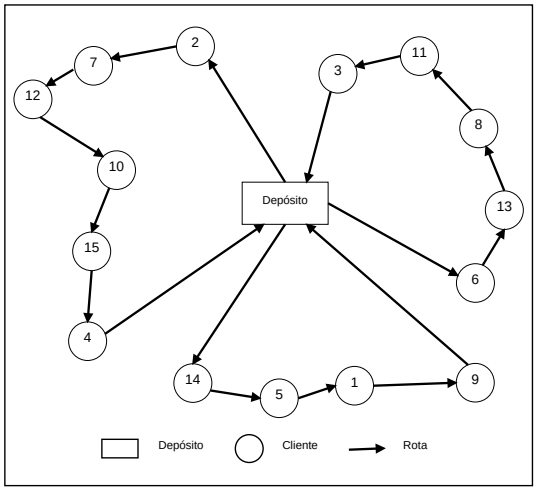

Figura 3.1: Representação de uma solução para o PRV

Assim, em um problema com $n = 16$ clientes, $k = 3$ veículos e $|D| = 3$ dias, uma possível solução pode ser representada por: [{ [8 , 1] , [9 , 6 , 2 , 3] , [5 , 7 , 4] } , { [8 , 10] , [15 , 11] , [12 , 9 , 4] } , { [8 , 16 , 6] , [14 , 7 , 13] }]. Cada conjunto de partições, unidos por {...} representa um dia do período de planejamento, e dentro de cada dia, temos uma representação de um PRV Clássico. Uma representação gráfica de uma solução com 3 dias pode ser vista na Figura 3.2.

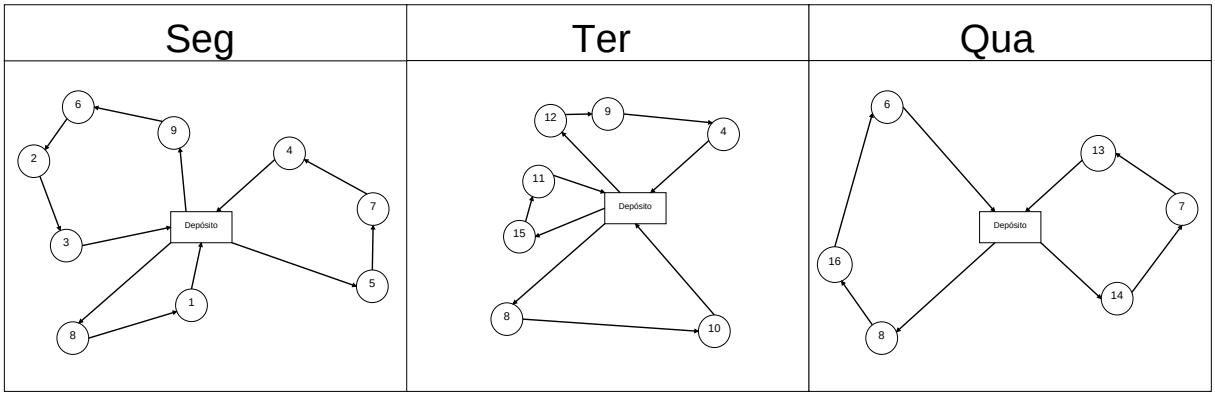

Figura 3.2: Representação de uma solução para o PRVP

3.2 Função de Avaliação

Uma solução s é avaliada por uma função $f(s)$ apresentada na Equação 3.1:

$$f(s) = \sum_{d \in D} \sum_{(i,j) \in A} c_{ij} x_{ijd} + f_{inv}(s) \tag{3.1}$$

em que:

- A: conjunto dos arcos (i, j), com $i, j \in N$
- D: conjunto de dias do período de planejamento
- c_{ij}: custos de transporte entre os clientes i e j
- x_{ijd}: indica se o arco $(i, j) \in A$ está sendo usado ($x_{i,j,d} = 1$) ou não ($x_{i,j,d} = 0$) na solução para o dia d

Como o arco $(i, j) \in A$ pode estar presente em mais de um dia, a variável de decisão x possui além dos índices desse arco, também o índice d referente ao dia que o arco está presente. Assim, caso o arco se repita na solução em dias diferentes, ele será contabilizado corretamente.

A parcela de inviabilidade $f_{inv}(s)$ foi calculada de acordo com a Equação 3.2:

$$f_{inv}(s) = w \times \sum_{d \in D} \sum_{k \in K} \max(0, Q'_{kd} - Q) \tag{3.2}$$

em que:

- D: conjunto de dias do período de planejamento
- K: conjunto de veículos disponíveis para cada dia
- Q'_{kd}: a capacidade utilizada do veículo k no dia d
- Q: a capacidade de transporte dos veículos
- w: peso dado a cada unidade de sobrecarga de capacidade

Dessa forma, para cada rota de cada dia, é calculado se houve um estouro da capacidade máxima do veículo. Caso houve esse estouro, um peso w é multiplicado por cada unidade que ultrapassou Q. Caso não tenha sido ultrapassada a capacidade máxima, não é atribuída nenhuma penalidade.

3.3 Geração de uma solução inicial

3.3.1 Baseada na Inserção Mais Barata por Padrões

A partir dos dados do problema, foi utilizado um método baseado na Heurística Construtiva de Inserção Mais Barata para gerar a solução inicial do problema abordado. Esse método é proposto nesse trabalho, e sua ideia é beneficiar os clientes que possuem uma frequência de visita (f_i) maior, pois são os que terão menos opções de inserção nos períodos. Esse método busca gerar soluções viáveis, mas caso não seja possível, uma relaxação da capacidade do veículo é utilizada.

Buscando a geração de soluções iniciais de melhor qualidade, utilizou-se também o Princípio da Otimalidade Próxima (*Proximate Optimality Principle*) [**?**] durante o processo construtivo. Esse princípio afirma que boas soluções em um nível estão próximas de boas soluções em um nível adjacente. Assim, a solução parcialmente gerada é submetida a uma busca local durante a construção. Esse procedimento de busca local é executado a cada $\delta\%$ do total de clientes serem inseridos. A busca local utilizada no POP é apresentada na Seção 3.5.4

O pseudocódigo do método é apresentado no Algoritmo 1, que recebe como parâmetro o número de clientes (N), o número de dias (D) e o número de veículos por dia(K). Nesse algoritmo, $f_{\max}$ representa a frequência máxima de visitas encontrada para todos os clientes, $taxaQ$ representa a porcentagem de capacidade total do veículo que pode ser utilizada na inserção (iniciada em 100%), e $inseriu$ garante com que um cliente sempre seja inserido na solução.

Inicialmente uma solução com rotas vazias é criada (linha 2 do Algoritmo 1) e uma lista L utilizada para armazenar os clientes para inserção é definida como vazia (linha 3 do Algoritmo 1). Após essa etapa de inicialização, os clientes são separados em grupos por frequência, em ordem decrescente. Os clientes pertencentes à cada grupo são embaralhados e adicionados à lista de clientes para inserção L (linhas 4-8 do Algoritmo 1). Ao embaralhar os clientes, garantimos que para cada geração inicial gerada a partir do mesmo problema-teste, novas soluções serão geradas. Após esse passo, a lista L conterá todos os clientes em ordem decrescente de frequência. Cada cliente pertencente a lista L será então adicionado sequencialmente a solução utilizando o Método de Inserção Mais Barata (linha 14 do Algoritmo 1), buscando não exceder a capacidade Q do veículo (linha 11 do Algoritmo 1). Caso não seja possível inserir o cliente, uma relaxação da capacidade do veículo é aplicada (linhas 15-17 do Algoritmo 1). Após o cliente ser inserido na solução

inicial ele é removido da lista L, e caso o critério de busca seja satisfeito, o Princípio da Otimalidade Próxima é aplicado (linhas 21-23 do Algoritmo 1). O processo se repete até que não haja mais nenhum cliente na lista. Esse método garante que todos os clientes sejam inseridos respeitando suas Listas de Padrões de Visita.

Algoritmo 1: GeraçãoSoluçãoInicial(N, D, K, δ)

```
1  início
2      Seja s_0 uma solução com |D| períodos e |K| rotas vazias
3      L ← ∅
4      para f = f_max até 1 faça
5          Grupo_f ← ClientesComFrequencia(f, N)
6          embaralha(Grupo_f)
7          insert(Grupo_f, L) // insere os cliente do Grupo_f em L
8      fim
9      enquanto L ≠ ∅ faça
10         c ← first(L) //o primeiro elemento de L é atribuído à c
11         taxaQ ← 1.0
12         repita
13             inseriu ← false
14             inseriu ← InserçãoMaisBarataPorPadrões(c, s_0, taxaQ)
15             se (¬ inseriu) então
16                 taxaQ = taxaQ + 0.1 //relaxação de Q
17             fim
18         até (inseriu)
19         remove(c, L) // Remove o cliente c da lista L
20         // Princípio da Otimalidade Próxima
21         se (δ% do total de clientes inseridos ) então
22             s_0 ← BuscaLocal(s_0)
23         fim
24     fim
25 fim
26 retorna s_0
```

O funcionamento do método *InserçãoMaisBarataPorPadrões* é detalhado no Algoritmo 2, que recebe como parâmetros o cliente c a ser inserido, a solução parcial s_0 e a porcentagem da capacidade Q do veículo que pode ser utilizada na inserção ($taxaQ$).

Inicialmente o cliente c ainda não foi inserido (linha 2 do Algoritmo 2), não há custo da melhor inserção (linha 3 do Algoritmo 2) e não temos as posições onde o cliente deve ser inserido em cada período (linha 4 do Algoritmo 2). A adição do cliente c à solução parcial é feita testando sua inserção em todos os seus dias definidos pelos padrões pertencentes à Lista de Padrões de Visita (LPV). Assim, para cada padrão pertencente à LPV (linha 5 do Algoritmo 2), são extraídos os possíveis dias de inserção que respeitam esse padrão e é calculado o custo de inserção do cliente c nesses dias (linhas 8-12 do Algoritmo 2). A inserção utilizada para cada período (linha 9 do Algoritmo 2) é descrita no Algoritmo 3. Caso não seja possível realizar a inserção do cliente pois a capacidade do veículo é violada em todas as rotas, um custo infinito é retornado. Verifica-se o custo do conjunto de inserções do padrão atual com o melhor custo obtido até então. Caso haja melhora, o melhor custo é atualizado e as posições das inserções que geraram esse melhor custo são armazenadas (linhas 13-16 do Algoritmo 2).

Após analisar todos os Padrões de Visita do cliente c, é verificado se ele pode ser inserido, ou seja, se a capacidade do veículo em algum dos dias não foi violada (linha 18 do Algoritmo 2). Caso as inserções respeitem a valor da capacidade do veículo utilizando a $taxaQ$, o cliente é inserido na solução parcial e o método retorna que foi possível inserir o cliente; caso contrário, o método retorna que não foi possível inserir o cliente utilizando a $taxaQ$ passada como parâmetro.

Algoritmo 2: InserçãoMaisBarataPorPadrões($c, s_0, taxaQ$)

```
1  início
2      inseriu ← false
3      custoBest ← ∞
4      posPeriodosBest ← ∅
5      para padrão ∈ LPV(c) faça
6          custoPadrão ← 0
7          posPeriodos ← ∅
8          para periodo ∈ padrão faça
9              [custo, pos] ← Inserção(c, s_0, periodo, taxaQ)
10             custoPadrão ← custoPadrão + custo
11             insert(pos, posPeriodos)
12         fim
13         se (custoPadrão < custoBest) então
14             custoBest ← custoPadrão
15             posPeriodosBest ← posPeriodos
16         fim
17     fim
18     se (custoBest < ∞) então
19         InsereCliente(c, s_0, posPeriodosBest)
20         inseriu ← true
21     fim
22 fim
23 retorna inseriu
```

Algoritmo 3: Inserção($c, s_0, periodo, taxaQ$)

```
1  início
2      custo ← 0
3      s_prv ← getPRV(s_0, periodo)
4      para cada rota r de s_prv faça
5          cost[r] ← o menor custo de inserção do cliente c para todas as posições em r
6          cap[r] ← capacidade atual do veículo utilizada para a rota r
7      fim
8      cost_min ← min{ cost[r] } ∀ r ∈ rotas de s_prv
9      r_min ← rota associada ao cost_min
10     //verificando se houve estouro de carga com a rota escolhida utilizando taxaQ
11     se cap[r_min] + q_c > Q × taxaQ então
12         custo ← ∞
13         pos ← ∅
14     senão
15         custo ← cost_min
16         pos ← arco de inserção associada ao cost_min
17         Adiciona o cliente c na posição pos da rota r_min
18     fim
19 fim
20 retorna [custo, pos]
```

3.4 Estruturas de Vizinhança

Para explorar o espaço de soluções, foram utilizadas 4 estruturas de vizinhança Intra-Rotas e 6 estruturas de vizinhança Inter-Rotas, todas pertencentes ao Problema de Roteamento de Veículos Clássico. Essas estruturas são descritas a seguir.

3.4.1 Intra-rotas

Nas Estruturas de Vizinhança Intra-Rotas os movimentos aplicados são sempre entre clientes que pertençam à mesma rota. Para ilustrar, tomaremos como solução inicial s a rota [1 , 2 , 3 , 4 , 5 , 6], representada pela Figura 3.3, e sobre ela aplicaremos movimentos de *Exchange*, *1Or-opt*, *2Or-opt* e *2-opt*. Como todas as soluções do espaço de busca são avaliadas, a complexidade computacional de cada uma das quatro estruturas

de vizinhança é dados por $\vartheta(\bar{n})$, onde $\bar{n}$ é o número de clientes da rota [**?**].

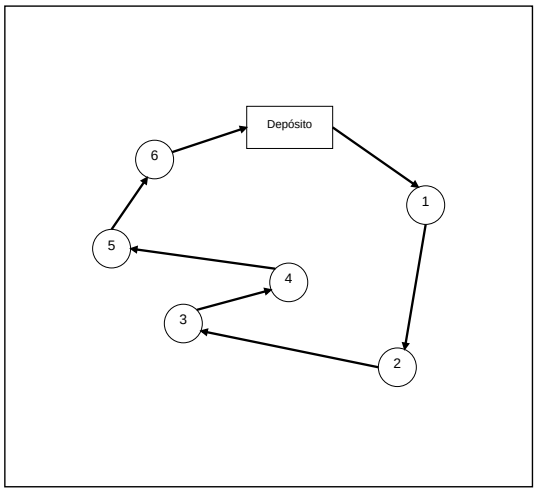

Figura 3.3: Solução Inicial s

3.4.1.1 *Exchange*

No movimento *Exchange* há uma permutação entre dois clientes i e j. No exemplo, os clientes $i = 3$ e $j = 4$ são trocados, obtendo assim uma nova solução $s' =$ [1 , 2 , 4 , 3 , 5 , 6], representada pela Figura 3.4.

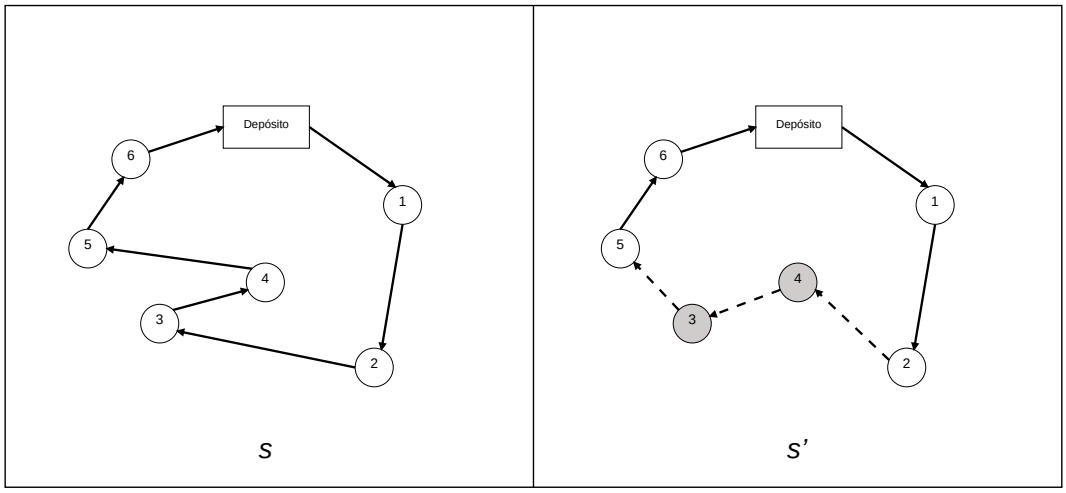

Figura 3.4: Solução s', após aplicação do *Exchange* em s

3.4.1.2 *1Or-opt*

No movimento *1Or-opt* há uma realocação de um cliente i para uma nova posição na rota. No exemplo, o cliente $i = 3$ é realocado entre os clientes 5 e 6, obtendo assim uma nova solução $s' =$ [1 , 2 , 4 , 5 , 3 , 6], representada pela Figura 3.5.

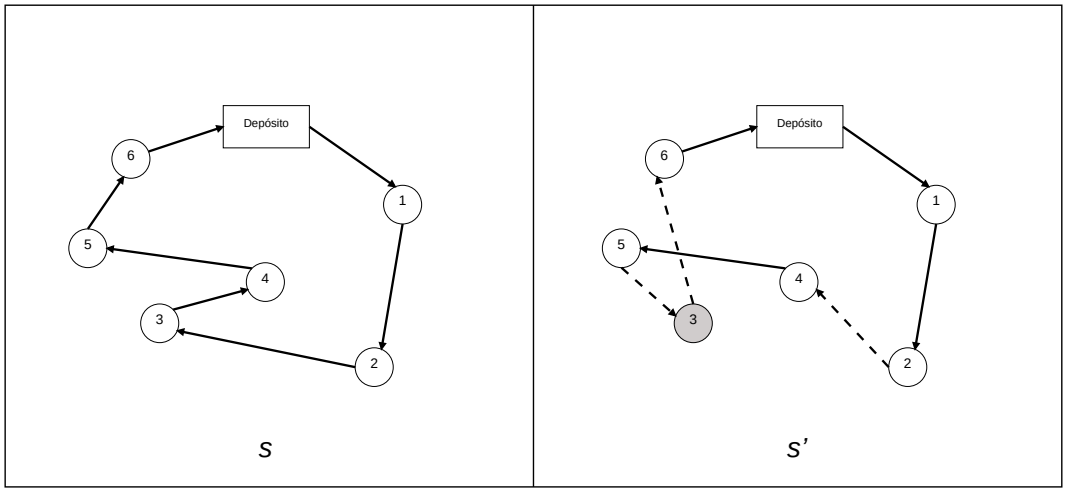

Figura 3.5: Solução s', após aplicação do *10r-opt* em s

3.4.1.3 *2Or-opt*

No movimento *2Or-opt* há uma realocação de dois clientes consecutivos i e j para uma nova posição na rota. No exemplo, os clientes $i = 2$ e $j = 3$ são realocados entre os clientes 4 e 5, obtendo assim uma nova solução $s' = [\ 1\ ,\ 4\ ,\ 2\ ,\ 3\ ,\ 5\ ,\ 6\ ]$, representada pela Figura 3.6.

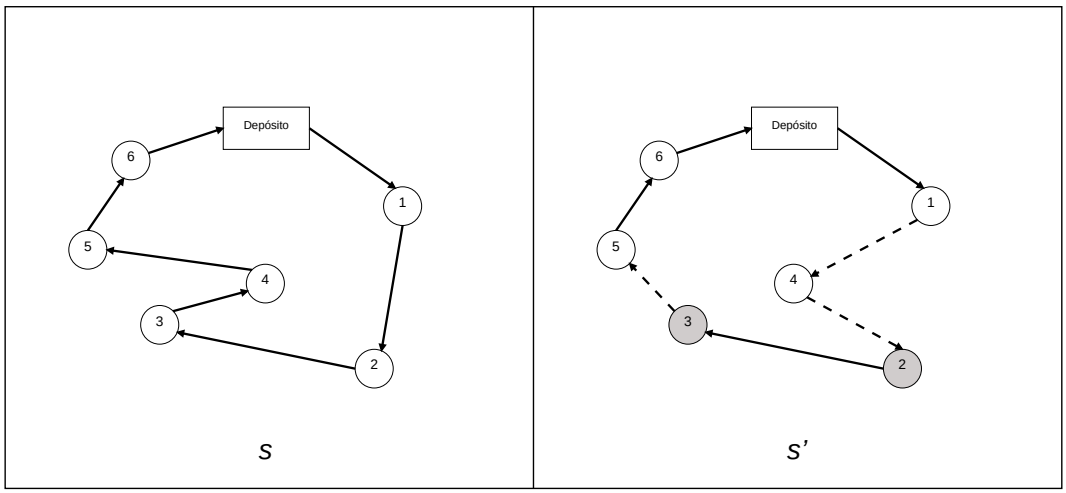

Figura 3.6: Solução s', após aplicação do *20r-opt* em s

3.4.1.4 *2-opt*

No movimento *2-opt*, dois arcos da solução, não adjacentes, são removidos e outros dois são inseridos formando uma nova solução onde há uma inversão na ordem de visitação dos clientes entre esses dois arcos. No exemplo, os arcos entre os clientes 3 e 4, e os cliente 5 e 6 são removidos e novamente ligados invertendo a ordem de visitação dos clientes 4 e 5.

Obtém-se, assim, uma nova solução $s' = [1,2,3,5,4,6]$, representada pela Figura 3.7.

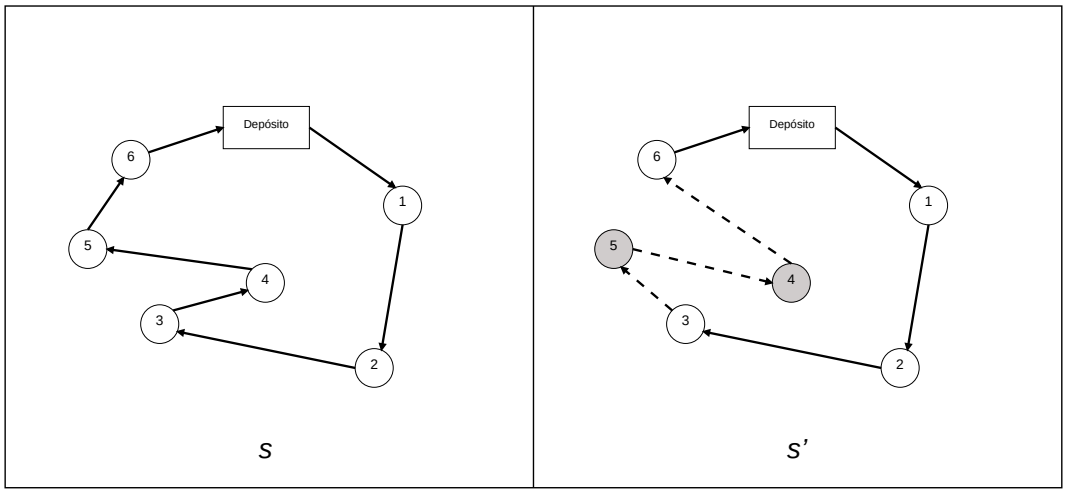

Figura 3.7: Solução s', após aplicação do *2-opt* em s

3.4.2 Inter-rotas

Nas Estruturas de Vizinhança Inter-Rotas os movimentos são aplicados entre clientes que pertencem à rotas diferentes. Para ilustrá-las, tomaremos como solução inicial s as seguintes rotas [[10 , 11 , 2 , 1 , 12 , 4] , [7 , 8 , 9 , 3 , 5 , 6]], apresentadas na Figura 3.8, a qual os movimentos de *Swap(1,1)*, *Swap(2,1)*, *Swap(2,2)*, *Shift(1,0)*, *Shift(2,0)* e *Cross* serão aplicados. Vale observar que ao se explorar todas as seis estruturas de vizinhança de forma exaustiva, obtemos uma complexidade computacional de $\vartheta(n^2)$, onde n representa o número de clientes nas rotas [**?**].

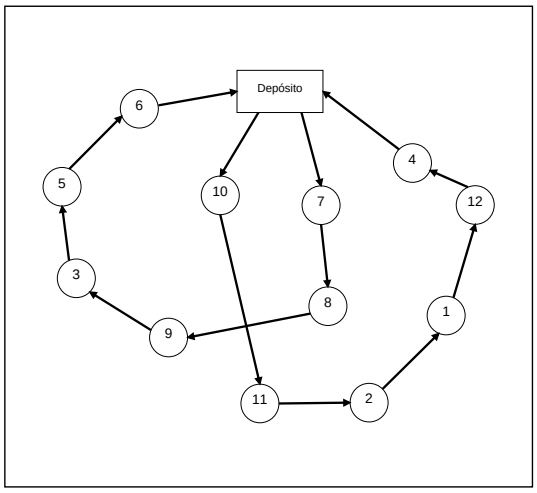

Figura 3.8: Solução Inicial s

3.4.2.1 *Swap(1,1)*

No movimento *Swap(1,1)* há uma permutação entre um cliente i da rota r_1 e um cliente i' da rota r_2. No exemplo, o cliente $i = 8$ e o cliente $i' = 11$ são trocados, obtendo assim uma nova solução $s' = [\ [\ 10\ ,\ 8\ ,\ 2\ ,\ 1\ ,\ 12\ ,\ 4\ ]\ ,\ [\ 7\ ,\ 11\ ,\ 9\ ,\ 3\ ,\ 5\ ,\ 6\ ]\ ]$, apresentada na Figura 3.9.

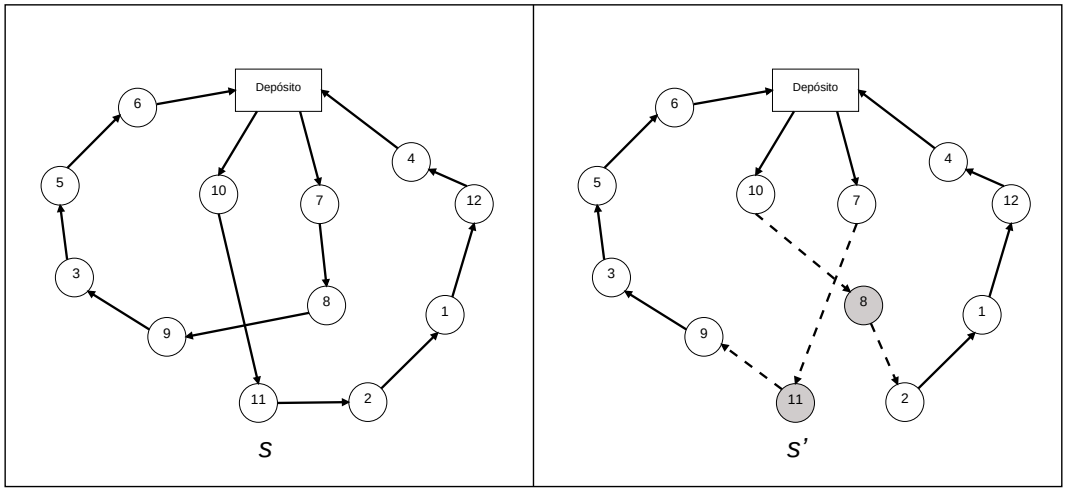

Figura 3.9: Solução s', após aplicação do *Swap(1,1)* em s

3.4.2.2 *Swap(2,1)*

No movimento *Swap(2,1)* há uma permutação entre dois clientes adjacentes i e j da rota r_1 e um cliente i' da rota r_2. A permutação é testada trocando tanto o arco (i,j) quanto o arco (j,i) com o cliente i'. No exemplo, os clientes $i = 7$ e $j = 8$ são trocados com o cliente $i' = 10$, obtendo assim uma nova solução $s' = [\ [\ 7\ ,\ 8\ ,\ 11\ ,\ 2\ ,\ 1\ ,\ 12\ ,\ 4\ ]\ ,\ [\ 10\ ,\ 9\ ,\ 3\ ,\ 5\ ,\ 6\ ]\ ]$, apresentada na Figura 3.10.

3.4.2.3 *Swap(2,2)*

No movimento *Swap(2,2)* há uma permutação entre dois clientes adjacentes i e j da rota r_1 e dois clientes adjacentes i' e j' da rota r_2. A permutação é testada fazendo as quatro combinações possíveis entre os arcos (i,j) e (i',j'). No exemplo, os clientes $i = 7$ e $j = 8$ são trocados com os clientes $i' = 10$ e $j' = 11$, obtendo assim uma nova solução $s' = [\ [\ 7\ ,\ 8\ ,\ 2\ ,\ 1\ ,\ 12\ ,\ 4\ ]\ ,\ [\ 10\ ,\ 11\ ,\ 9\ ,\ 3\ ,\ 5\ ,\ 6\ ]\ ]$, apresentada na Figura 3.11.

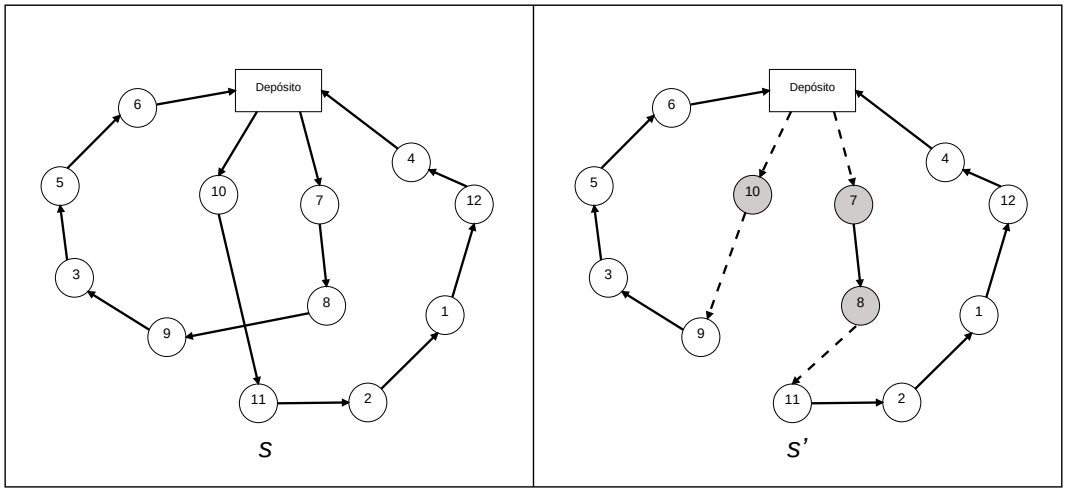

Figura 3.10: Solução s', após aplicação do *Swap(2,1)* em s

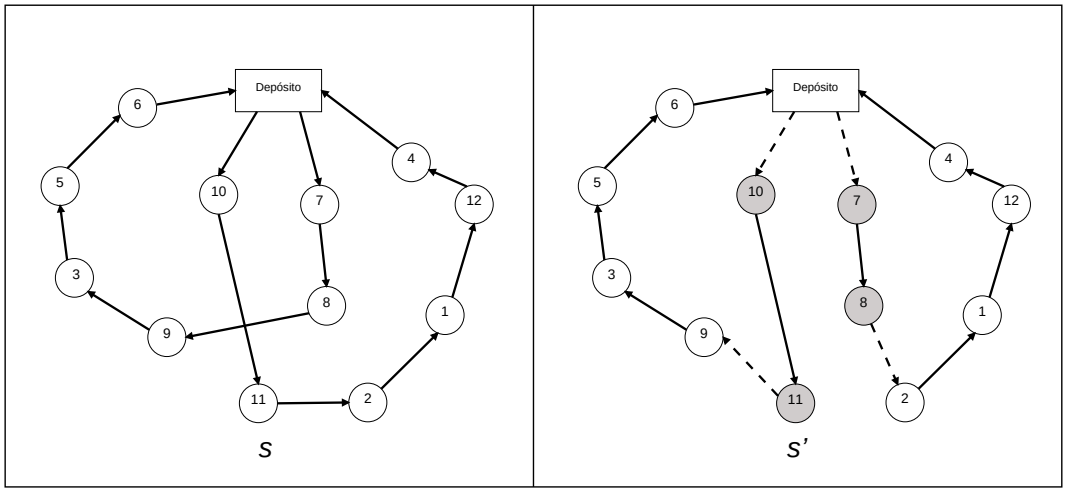

Figura 3.11: Solução s', após aplicação do *Swap(2,2)* em s

3.4.2.4 *Shift(1,0)*

No movimento *Shift(1,0)* há uma realocação de um cliente i da rota r_1 para uma nova posição na rota r_2. No exemplo, o cliente $i = 9$ foi realocado de sua rota para a posição entre os clientes 10 e 11 da outra rota, obtendo assim uma nova solução $s' =$ [[10 , 9 , 11, 2 , 1 , 12 , 4] , [7 , 8 , 3 , 5 , 6]], apresentada na Figura 3.12.

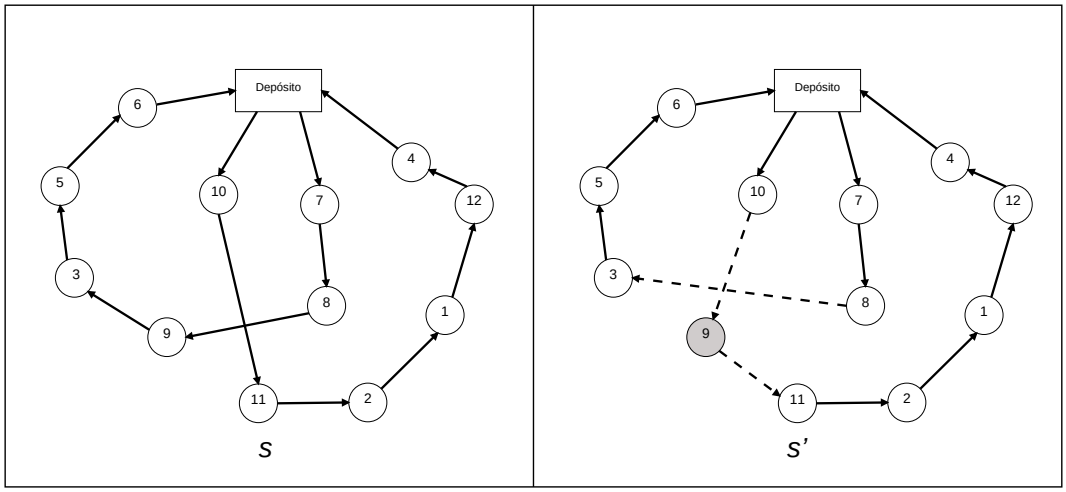

Figura 3.12: Solução s', após aplicação do *Shift(1,0)* em s

3.4.2.5 *Shift(2,0)*

No movimento *Shift(2,0)* há uma realocação de dois clientes adjacentes i e j da rota r_1 para uma nova posição na rota r_2. Assim como no movimento *Swap(2,1)* a realocação é testado com os arcos (i,j) e (j,i). No exemplo, os clientes $i = 10$ e $j = 2$ foram realocados de sua rota para a posição entre os clientes 8 e 9 da outra rota, obtendo assim uma nova solução $s' =$ [[10 , 1 , 12 , 4] , [7 , 8 , 2 , 11 , 9 , 3 , 5 , 6]], apresentada na Figura 3.13. Observa-se que nessa nova solução gerada, o arco utilizado foi (j,i).

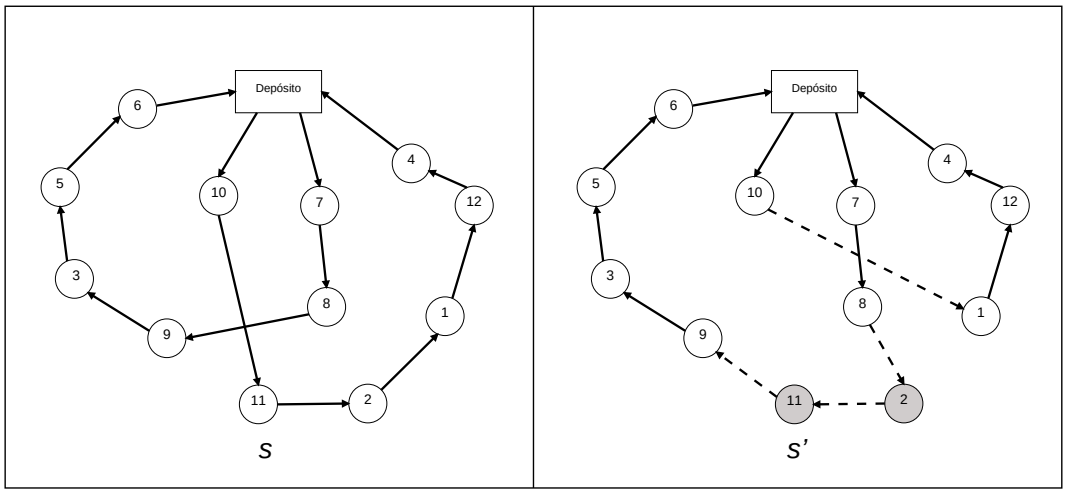

Figura 3.13: Solução s', após aplicação do *Shift(2,0)* em s

3.4.2.6 *Cross*

No movimento *Cross* são escolhidos dois arcos, $(i,\ j)$ pertencente à rota r_1, e outro $(i',\ j')$ pertencente à rota r_2. Esses arcos são removidos e dois novos arcos $(i,\ j')$ e $(i',\ j)$ são

adicionados. No exemplo, os arcos (9, 3) e (10, 11) são removidos e os arcos (9, 11) e (10, 3) são adicionados formando uma nova solução $s' = [\ [\ 7\ ,\ 8\ ,\ 9\ ,\ 11\ ,\ 2\ ,\ 1\ ,\ 12\ ,\ 4\ ]\ ,\ [\ 10\ ,\ 3\ ,\ 5\ ,\ 6\ ]\ ]$, representada pela Figura 3.14.

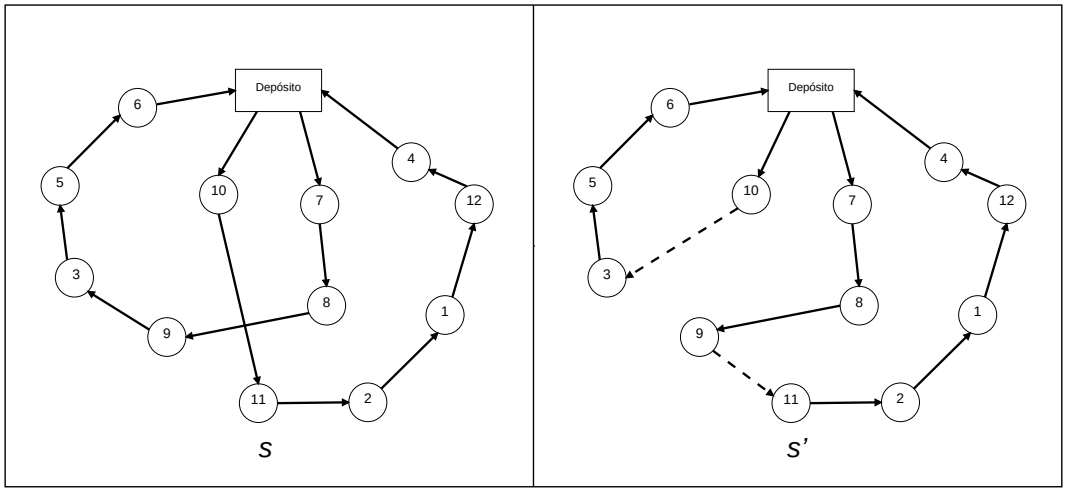

Figura 3.14: Solução s', após aplicação do *Cross* em s

3.5 Buscas Locais

Para realizar as buscas locais para o PRVP, foram utilizas duas heurísticas. A primeira, denominada *Periodic* VND, explora as características do Problema de Roteamento de Veículos Clássico presentes no PRVP, e a outra, denominada *Pattern Improvement*, explora a característica de periodicidade do PRVP. Essas heurísticas são apresentadas nas nas Seções 3.5.2 e 3.5.3, respectivamente.

3.5.1 Estruturas Auxiliares de Dados

Visando auxiliar as heurísticas de Busca Local, três Estruturas Auxiliares de Dados (EAD) foram utilizadas: $carga[\ ][\ ]$, $padraoCliente[\ ]$ e $statusDia[\ ]$. Cada uma dessas estruturas armazena em si informações úteis sobre as rotas da solução corrente do Algoritmo.

- $carga[\ ][\ ]$: esta estrutura guarda o valor da carga utilizada de cada veículo k para cada dia d ($carga[d][k]$). Essa informação é utilizada para que um movimento inviável, que exceda a capacidade Q do veículo, seja descartado antes de ser aplicado. Assim movimentos que gerariam soluções inviáveis são evitados, não sendo aplicados desnecessariamente.

- *padraoCliente*[ ]: esta estrutura guarda em qual Padrão de Visita o cliente se encontra. Assim, quando há troca de padrões de visita de um cliente, e este precisa ser removido, não é necessário que toda a solução seja percorrida à procura dele, bastando acessar os dias em que ele realmente se encontra.

- *statusDia*[ ]: esta estrutura guarda se houve alguma modificação na solução no dia d. Assim, caso $statusDia[d] = 1$, alguma mudança ocorreu no dia d e 0, caso contrário. Esta estrutura faz que somente dias que sofreram algum tipo de alteração passem por Buscas Locais, evitando que dias que não sofreram alterações tentem ser explorados exaustivamente sem que nenhuma melhora possa ser conseguida.

Além dessas Estruturas Auxiliares de Dados, em todas as Estruturas de Vizinhança, exceto a *Cross*, é aplicada uma regra delta para cálculo do novo valor da Função Objetivo após a aplicação de um movimento. Essa regra delta evita que a solução tenha que ser totalmente avaliada ($O(n^3)$ para o PRVP), avaliando somente as alterações locais realizadas nas rotas e clientes envolvidos nesse movimento em $O(1)$. Como exemplo da aplicação dessa regra de recálculo, iremos utilizar uma aplicação de um movimento Swap(1,1) em uma solução do PRV Clássico, Figura 3.15.

Na parte (a) da Figura 3.15, é definida a solução inicial s', que consideraremos com um valor de função de avaliação $f(s') = \Theta$. Na parte (b) é definido em quais clientes o movimento será aplicado, nesse exemplo são os clientes 8 e 10. Na parte (c) o movimento foi aplicado e a nova solução s" é gerada. Normalmente, ao se aplicar um movimento, tem-se um novo valor da função objetivo $f(s") = \Theta'$. Para se calcular esse novo valor para o PRV, computa-se toda a distância percorrida por cada veículo em cada uma das rotas, levando a uma reavaliação completa de toda a solução, mesmo em rotas em que o movimento não foi aplicado. Assim há um desperdício computacional ao calcularmos todas as distâncias de todas as rotas, quanto na verdade somente parte das rotas foram modificadas. A ideia dessa regra é calcular o valor Δ somente das distâncias que foram modificadas.

No exemplo apresentado na Figura 3.15, na parte (b) os arcos tracejados serão os que deixarão de existir com o movimento {(0,10),(10,11),(7,8),(8,9)} , e na parte (c) os arcos pontilhados são os novos arcos criados {(7,10),(10,11),(0,8),(8,9)}. Portanto o restante da solução se mantém, alterando-se somente esses arcos marcados. O valor alterado nas distância é calculado com o $\Delta = -arcos\_removidos + arcos\_adicionados$, onde $arcos\_removidos = c_{0,10} + c_{10,11} + c_{7,8} + c_{8,9}$ e $arcos\_adicionados = c_{7,10} + c_{10,11} + c_{0,8} + c_{8,9}$. Dessa forma, o recálculo da função de avaliação $f(s")$ apresenta complexidade

de $O(1)$ pois é feita através de uma soma, $f(s") = \Theta + \Delta$.

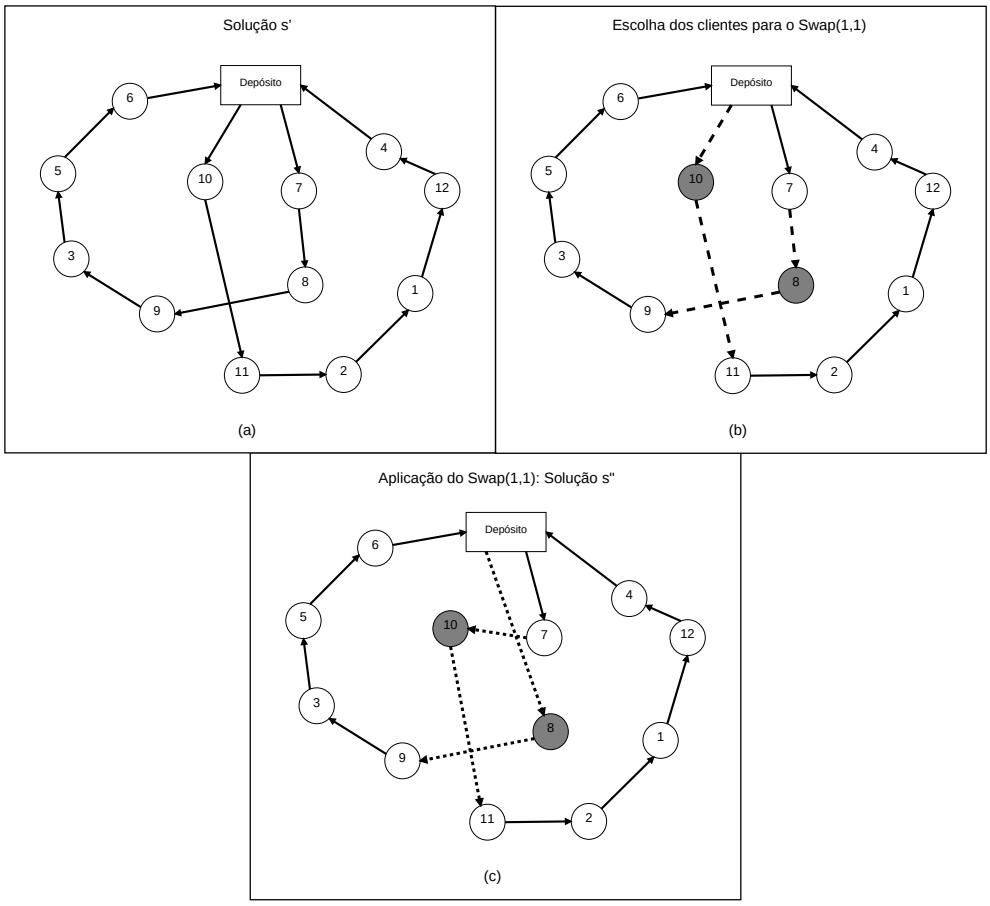

Figura 3.15: Aplicação da regra delta

3.5.2 *Periodic* VND

A heurística *Variable Neighborhood Descent* (VND) [**?**], é um método de refinamento que consiste em explorar o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada [**?**].

As trocas de estruturas de vizinhança são classicamente feitas seguindo uma ordem pré-estabelecida. No presente trabalho optou-se por utilizar uma ordem aleatória de exploração, que além de simplificar o algoritmo, pois como são utilizadas 10 estruturas de vizinhança para o problema, há 10! permutações possíveis e não seria viável estabelecer a ordem mais adequada. A utilização dessa técnica obteve bons resultados nos problemas tratados por [**?**], [**?**] e [**?**] e é denominada *Variable Neighborhood Descent with random neighborhood ordering* (RVND). Seu pseudo-código é apresentado no Algoritmo 4 abaixo.

Algoritmo 4: RVND(s)

```
1  início
2      NL ← Lista com todas as Estruturas de Vizinhança do PRV
3      Embaralha(NL)
4      r ← |NL| //número de Estruturas de Vizinhança
5      k ← 1 //controle de qual Estrutura de Vizinhança está sendo aplicada
6      enquanto k < r faça
7          Encontra o melhor vizinho s' ∈ N^(k) de s
8          se f(s') < f(s) então
9              s ← s' //atualizando a melhor solução
10             k ← 1 //retornado à primeira Estrutura de Vizinhança
11         senão
12             k ← k + 1
13         fim
14     fim
15 fim
16 retorna s
```

Como podemos considerar cada dia do PRVP como um PRV Clássico, a heurística RVND é utilizada, em cada um desses dias, com todas as Estruturas de Vizinhança do PRV descritas na Seção 3.4 aplicadas diretamente ao PRVP sem que adaptações tenham sido feitas. Essa estratégia de executar o RVND em cada dia foi denominado *Periodic VND* (PVND). A exploração do espaço de soluções é feita de forma exaustiva, onde todos os movimentos são aplicados e somente o melhor deles é efetivado na solução corrente. Essa forma de exploração é definida na Literatura como *Best Improvement*.

O *PVND* faz uso da EAD $statusDia[\,]$, aplicando o RVND somente nos dias em que alguma mudança foi feita. A pseudocódigo do PVND é apresentado no Algoritmo 5.

Algoritmo 5: PVND(s, statusDia[ ])

```
1  início
2      para d = 1 até |D| faça
3          se (statusDia[d] = 1) então
4              s_prv ← getPRV(s, d) //obtendo o PRV referente ao dia d
5              s'_prv ← RVND(s_prv)
6              setPRV(s, d, s'_prv) //atualizando a solução corrente para o dia d
7              statusDia[d] ← 0 //dia d não possui mais modificações
8          fim
9      fim
10 fim
11 retorna s
```

3.5.3 *Pattern Improvement*

A heurística *Pattern Improvement* (PI) foi proposta por [**?**] e visa explorar a característica de periodicidade do PRVP. Para entendermos o seu funcionamento, tomaremos $\bar{p}$ o padrão de visita do cliente i na solução corrente. O método PI é executado sobre todos os clientes, tomados em ordem aleatória, e computa, para cada cliente i e padrão $p \in LPV_i$, $\Psi(i,p) = \sum_{l \in p} \psi(i,l)$, o custo mínimo que satisfaz aos requisitos de visita do cliente i de acordo com o padrão de visita p. Definimos $\psi(i,l)$ como o custo da inserção mais barata do cliente i no dia l. Caso o padrão de visita exista de tal modo que $\Psi(i,p) < \Psi(i,\bar{p})$, então todas as visitas do cliente i são removidas da solução corrente, e as novas visitas são adicionadas na melhor posição encontrada pela busca nos dias $l \in p$. A busca termina quando todos os clientes forem considerados e nenhuma mudança ocorrer na solução corrente.

É importante notar que, em alguns casos, o padrão $\bar{p}$ do cliente i é mantido, mas uma melhor inserção do cliente i dentro dos dias do padrão é feita. Nesse caso, podemos considerar o PI como um $Shift(1,0)$ periódico. Observamos assim que o *Pattern Improvement* trabalha de duas formas, tanto alterando os dias de visitação dos clientes, como também realizando uma busca local dentro de cada PRV da solução. O pseudocódigo do *Pattern Improvement* é apresentado no Algoritmo 6.

Algoritmo 6: PI(s, padraoCliente[ ])

```
1  início
2      alterado ← false
3      repita
4          s' ← s // Fazendo uma cópia da solução corrente
5          L ← {1, 2, 3, ..., n} //Criando uma lista com todos os clientes
6          embaralha(L)
7          alterado ← false
8          // Enquanto não verificar todos os clientes
9          enquanto L ≠ ∅ faça
10             c ← first(L) //o primeiro cliente de L é atribuído à c
11             removeCliente(c, s', padraoCliente[c])
12             taxaQ ← 1.0
13             repita
14                 inseriu ← false
15                 inseriu ← InserçãoMaisBarataPorPadrões(c, s', taxaQ)
16                 se (¬ inseriu) então
17                     taxaQ = taxaQ + 0.1 //relaxação de Q
18                 fim
19             até (inseriu)
20             remove(c, L) // Remove o cliente c da lista L
21             //Verificando se a inserção modificou a solução corrente
22             se (s' ≠ s) então
23                 alterado ← true
24             fim
25         fim
26         s ← s' //Atualizando a solução corrente
27     até (alterado = false)
28     Atualiza o statusDia[ ] de acordo com as mudanças nos dias
29 fim
30 retorna s
```

3.5.4 Busca Local utilizada no POP

A busca local utilizada no Princípio da Otimalidade Próxima é uma versão reduzida do PVND aplicado ao PRVP. Nessa versão reduzida, apenas algumas Estruturas de Vizinhança foram escolhidas para tornar o método computacionalmente mais rápido. A escolha das Estruturas de Vizinhança utilizadas foi feita de modo empírico, e dentre as combinações testadas, a que obteve melhores resultados na melhoria da solução inicial gerada foi a combinação das estruturas de vizinhança *1Or-opt*, *Exchange*, *Swap(1,1)* e *Shift(1,0)*.

3.6 Algoritmo Proposto ILS-PVND

O *Iterated Local Search* – ILS [**?**], é uma metaheurística que procura focar a busca não no espaço completo de soluções, mas num pequeno subespaço definido por soluções que são ótimas locais de um determinado mecanismo de otimização. De acordo com [**?**], o sucesso do ILS é centrado no conjunto de amostragem de ótimos locais, juntamente com a escolha da Busca Local, das Perturbações e do Critério de Aceitação.

O algoritmo proposto, denominado ILS-PVND, é um algoritmo *Multistart* que baseia-se na metaheurística ILS e recebe como parâmetro a instância que irá executar, o número de iterações *Multistart* ($iterMax$), o número de iterações sem melhora do ILS ($iterMaxILS$), e δ que é utilizado como critério de execução do Princípio da Otimalidade Próxima dentro do gerador de solução inicial. Seu pseudocódigo é apresentando no Algoritmo 7.

O método de geração de Solução Inicial utilizado é o baseado na Inserção Mais Barata por Padrões, apresentado na Seção 3.3.1, que por gerar boas soluções e ter um caráter aleatório, faz com a sua utilização seja adequada para a metodologia *Multistart* adotada (linha 6 do Algoritmo 7).

A Busca Local utilizada é uma combinação entre o *Pattern Improvement* e o PVND. Uma solução é passada para essa busca local que primeiramente executa o PI na solução corrente. Após essa execução, a solução gerada pelo PI é passada para o PVND para uma nova etapa de otimização. O processo se repete até que não haja mais melhora na solução corrente (linhas 14-18 do Algoritmo 7).

A etapa de perturbação da solução foi implementada utilizando a aplicação de um movimento aleatório de uma das seguintes Estruturas de Vizinhança: *Swap(1,1)*, *Shift(1,1)*,

Shift(2,1) e *ShiftLPV* (linha 12 do Algoritmo 7). Essa combinação foi escolhida de forma empírica, onde combinações de várias estruturas de vizinhança foram testadas, e essa escolhida foi a que se destacou entre as demais.

A estrutura de vizinhança *Swap(1,1)* foi detalhada na Seção 3.4.2.1. Na estrutura de vizinhança *Shift(1,1)*, dois clientes e duas posições são escolhidos, i e p_1 pertencentes à rota r_1 e i' e p_2 pertencentes à rota r_2. O cliente i é removido da rota r_1 e inserido na rota r_2 na posição p_2 e o cliente j é removido da rota r_2 e inserido na rota r_1 na posição p_1. Já na estrutura de vizinhança *Shift(2,1)*, três clientes e duas posições são escolhidos, dois clientes adjacentes i e j e a posição p_1 pertencentes à rota r_1, e um cliente i' e uma posição p_2 pertencentes à rota r_2. Os clientes i e j são removidos da rota r_1 e inseridos na posição p_2 da rota r_2, e o cliente i' é removido da rota r_2 e inserido na posição p_1 da rota r_1. A última estrutura de vizinhança aplicada como perturbação é a *ShiftLPV*, que consiste na troca do Padrão de Visita de um cliente escolhido aleatoriamente.

Caso haja melhora no valor da função objetivo, a melhor solução encontrada é armazenada e é reiniciado o contador de iterações sem melhora do ILS ($iterILS$). Observa-se que somente soluções de melhora foram consideradas pelo Critério de Aceitação adotado na execução do ILS (linhas 20-25 do Algoritmo 7). Após cada execução completa do ILS, é feito o teste que armazena a melhor solução encontrada em todas as iterações do ILS-PVND (linhas 27-30 do Algoritmo 7).

Algoritmo 7: ILS-PVND($instancia, iterMax, iterMaxILS, \delta$)

```
1  início
2      prob ← CarregaProblemaTeste(instancia)
3      s_best ← ∅
4      f_best ← ∞
5      para iter = 1 até iterMax faça
6          s_0 ← GeraçãoSoluçãoInicial(prob.N, prob.D, prob.K, δ)
7          statusDia[d] ← 1 //todos os dias d ∈ D podem receber Busca Local
8          padraoCliente ← extraiPadrões(s_0)
9          s* ← PVND(s_0, statusDia)
10         iterILS ← 0
11         enquanto iterILS < iterMaxILS faça
12             s' ← Perturbação(s*)
13             //Busca Local
14             enquanto houver melhora faça
15                 s" ← PI(s', padraoCliente)
16                 s" ← PVND(s", statusDia)
17                 s' ← s"
18             fim
19             //Critério de Aceitação
20             se (f(s') < f(s*)) então
21                 s* ← s'
22                 iterILS ← 0
23             senão
24                 iterILS ← iterILS + 1
25             fim
26         fim
27         se (f(s*) < f_best) então
28             s_best ← s*
29             f_best ← f(s*)
30         fim
31     fim
32 fim
33 retorna s_best
```

3.7 Pré-processamento

Um das maiores dificuldades encontradas no PRVP é exatamente sua característica principal, a periodicidade, ou seja, ter que definir, dentre os Padrões de Visita de cada cliente, qual ou quais os melhores dias para atendê-lo. Porém, podem ocorrer casos onde, apesar de se trabalhar com múltiplos dias, todos os clientes requerem somente uma visita, e podem ser visitados em qualquer dia do período de planejamento.

Considerando esses casos, podemos transformar o PRVP em um PRV Clássico, considerando o número de veículos para o PRV Clássico como o número de veículos disponíveis por dia para o PRVP vezes o número de dias do período de planejamento ($K_{PRV} = K_{PRVP} \times |D|$), e atendendo a todos os clientes utilizando esses K_{PRV} veículos. Para transformar a solução encontrada para o PRV em uma solução do PRVP, basta formarmos $|D|$ conjuntos de K_{PRVP} rotas consecutivas, onde cada um desses conjuntos representará um dia da solução do PRVP. Assim, é possível reduzir a complexidade de alguns casos para o PRVP. Um exemplo é apresentada no Figura 3.16, onde temos $|D| = 2$, $K_{PRVP} = 3$, logo $K_{PRV} = 3 \times 2 = 6$.

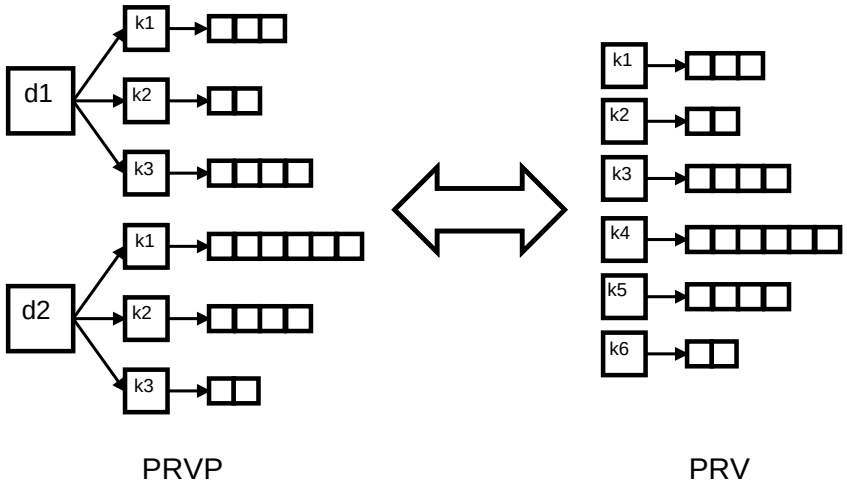

Figura 3.16: Transformação da representação do PRVP para o PRV

Capítulo 4

Resultados Computacionais

4.1 Ambiente de desenvolvimento

O algoritmo desenvolvido no presente trabalho, ILS-PVND, foi desenvolvido utilizando a linguagem C++ e o *framework* OptFrame v1.3[1] [**?**]. Os testes foram executados em um computador com processador Intel Core i7 3,07 GHz, 8GB de memória RAM, sistema operacional Ubuntu 10.04, kernel 2.6.32-36-generic e compilador gcc v4.4.3. Apesar de o computador oferecer recursos *multi-core*, esses não foram utilizados.

4.2 Problemas-teste utilizados

Para testar o algoritmo foi utilizado um conjunto de 32 problemas-teste que não possuem restrição de duração da rota. Desse conjunto, dez problemas-teste (p01-p10) foram criados por [**?**] para o PRV e adaptados por [**?**] para o PRVP. Um problema-teste (p11) foi proposto por [**?**]. Dois problemas-teste (p12 e p13) foram propostos por [**?**] e dezenove (p14-p32) por [**?**]. Eles possuem de 50 a 417 clientes, com períodos de 2 a 10 dias. As principais características dos problemas-teste são apresentados na Tabela 4.1, na qual n representa o número de clientes, K o número de veículos, d o número de dias, Q a capacidade de cada veículo, e f_i quantos clientes existem por frequência de visita i, ou seja, quantos clientes precisam ser visitados i dias.

[1] http://sourceforge.net/projects/optframe/

Tabela 4.1: Características dos Problemas-teste

Problema-teste	n	K	d	Q	Número de visitas					
					f_1	f_2	f_3	f_4	f_5	f_6
p01	50	3	2	160	50					
p02	50	3	5	160	17	26			7	
p03	50	1	5	160	50					
p04	75	2	5	140	75					
p05	75	6	5	140	30	34			11	
p06	75	1	10	140	75					
p07	100	4	2	200	100					
p08	100	5	5	200	40	46			14	
p09	100	1	8	200	100					
p10	100	4	5	200	40	46	14			
p11	139	4	5	235	103	22	12	1	1	
p12	163	3	5	140	148	8	7			
p13	417	9	7	2000	377	40				
p14	20	2	4	20	8	8		4		
p15	38	2	4	30	16	16		6		
p16	56	2	4	40	24	24		8		
p17	40	4	4	20	16	16		8		
p18	76	4	4	30	32	32		12		
p19	112	4	4	40	48	48		16		
p20	184	4	4	60	80	80		24		
p21	60	6	4	20	24	24		12		
p22	114	6	4	30	48	48		18		
p23	168	6	4	40	72	72		24		
p24	51	3	6	20	36	9				6
p25	51	3	6	20	36	9				6
p26	51	3	6	20	36	9				6
p27	102	6	6	20	72	18				12
p28	102	6	6	20	72	18				12
p29	102	6	6	20	72	18				12
p30	153	9	6	20	108	27				18
p31	153	9	6	20	108	27				18
p32	153	9	6	20	108	27				18

4.3 Comparações com a Literatura

O ILS-PVND foi testado no conjunto de 32 problemas-teste apresentados na Seção 4.2, sendo executados 10 vezes para cada um deles, com limite de tempo de 1 hora (3600 segundos). Os parâmetros do algoritmo foram calibrados empiricamente, sendo seus valores apresentados a seguir. Na geração de Solução Inicial apresentada na Seção 3.3.1, a Busca Local foi acionada a cada $\delta = 20\%$ dos clientes inseridos na solução. O número de iterações *Multistart* ($iterMax$) foi fixado em 10, e o número de iterações sem melhora do ILS foi calculado como: $(n \times K \times d) \times 0,013 + 196$, onde n, K e d representam o número de clientes, número de veículos e número de dias, respectivamente.

Na Tabela 4.2 são apresentados os melhores resultados encontrados na literatura para cada um dos problemas-teste analisados e também os resultados obtidos pelo ILS-PVND. Na primeira coluna indica-se o nome do problema-teste. A segunda coluna, Literatura, é feita uma compilação dos melhores resultados disponíveis na literatura. Na terceira e quarta colunas são apresentados os melhores resultados obtidos pelo ILS-PVND, e o tempo gasto, em segundos, para a obtenção desse resultado. Na quinta coluna é calculado o GAP (Eq. 4.1) da melhor solução encontrada pelo ILS-PVND e o melhor resultado do Alg_X, onde nessa tabela X são os melhores Algoritmos da Literatura. Na quinta e sexta coluna são apresentados os valores médios obtidos pelo ILS-PVND e também o tempo médio gasto pelo algoritmo por problema teste, respectivamente. Na sétima coluna é apresentado o GAP Médio (Eq. 4.2), que é calculado utilizando-se o resultado médio obtido pelo ILS-PVND e o melhor resultado da Literatura. Na penúltima coluna é apresentado o Desvio Padrão (σ) para cada um dos problemas-teste. Por fim, na última coluna é apresentado o Coefiente de Variação (Eq. 4.3), uma medida de variabilidade relativa, definida como a razão percentual entre o Desvio Padrão (σ) e a média dos resultados obtidos pelo ILS-PVND, e que mede a robustez do algoritmo. Os empates nos resultados entre a metodologia proposta e a Literatura são destacados em negrito.

$$GAP = \frac{(melhor_{ILS-PVND} - melhor_{Alg_X})}{melhor_{Alg_X}} \tag{4.1}$$

$$GAPMédio = \frac{(media_{ILS-PVND} - melhor_{Alg_X})}{melhor_{Alg_X}} \tag{4.2}$$

$$Coef.Var = \frac{\sigma}{media_{ILS-PVND}} \tag{4.3}$$

Tabela 4.2: ILS-PVND × Literatura

Inst.	Lit.	Melhor	Tempo(s)	GAP	Média	T. Médio(s)	GAP Médio	σ	Coef. Variação
p01	524,61	524,61	23,92	**0,00**	524,61	32,12	**0,00**	0,00	0,00
p02	1322,87	1322,87	42,12	**0,00**	1336,76	53,63	1,05	6,64	0,50
p03	524,60	524,61	22,28	**0,00**	524,61	24,78	**0,00**	0,00	0,00
p04	835,26	835,26	88,28	**0,00**	842,06	116,30	0,81	3,95	0,47
p05	2024,96	2037,97	429,16	0,64	2055,40	533,20	1,50	8,38	0,41
p06	835,26	835,26	256,27	**0,00**	838,66	199,46	0,41	3,44	0,41
p07	826,14	826,14	212,03	**0,00**	829,55	330,08	0,41	1,53	0,18
p08	2022,47	2040,78	403,06	0,91	2053,63	467,17	1,54	8,29	0,40
p09	826,14	826,14	262,20	**0,00**	829,76	331,83	0,44	1,63	0,20
p10	1593,43	1604,26	370,22	0,68	1624,69	327,15	1,96	8,99	0,55
p11	770,89	779,29	700,20	1,09	791,70	695,71	2,70	5,17	0,65
p12	1186,47	1202,72	611,14	1,37	1228,78	626,39	3,57	10,52	0,86
p13	3462,73	3596,60	3600,00	3,87	3660,36	3600,00	5,71	19,94	0,54
p14	954,80	954,81	3,49	**0,00**	954,81	3,54	**0,00**	0,00	0,00
p15	1862,60	1862,63	11,11	**0,00**	1862,63	11,40	**0,00**	0,00	0,00
p16	2875,10	2875,24	31,03	**0,00**	2875,24	55,69	**0,00**	0,00	0,00
p17	1597,75	1597,75	32,79	**0,00**	1608,01	36,20	0,64	8,48	0,53
p18	3131,09	3150,25	100,35	0,61	3158,33	149,83	0,87	4,55	0,14
p19	4834,34	4846,49	295,59	0,25	4846,49	334,66	0,25	0,00	0,00
p20	8367,40	8367,40	1649,73	**0,00**	8367,40	1823,77	**0,00**	0,00	0,00
p21	2170,61	2173,36	122,26	0,13	2182,97	147,72	0,57	3,20	0,15
p22	4193,95	4231,03	549,91	0,88	4288,10	771,21	2,24	35,72	0,83
p23	6420,71	6426,51	1485,95	0,09	6608,13	3460,42	2,92	27,23	0,41
p24	3687,46	3687,46	55,68	**0,00**	3716,95	64,72	0,80	24,68	0,66
p25	3777,15	3781,38	30,48	0,11	3782,25	33,37	0,13	2,61	0,07
p26	3794,95	3795,32	400,72	0,01	3796,43	596,96	0,04	1,76	0,05
p27	21833,87	21998,78	361,82	0,76	22196,46	512,82	1,66	175,18	0,79
p28	22242,50	22357,81	434,36	0,52	22412,13	556,47	0,76	32,45	0,14
p29	22543,75	22667,95	459,00	0,55	22794,92	562,98	1,11	112,99	0,50
p30	73875,19	75352,63	2506,87	2,00	76351,77	2912,84	3,35	510,91	0,67
p31	76001,57	77145,98	2273,87	1,51	78313,88	3089,94	3,04	597,03	0,76
p32	77598,00	78740,91	2379,23	1,47	80215,91	2970,98	3,37	628,79	0,78
Médias				0,55			1,31		0,36

Nas Tabelas 4.3, 4.4, 4.5 e 4.6 são feitas comparações de cada um dos algoritmos "estado da arte" para a resolução do PRVP e do algoritmo proposto neste trabalho. Na Tabela 4.3 o ILS-PVND é comparado com o algoritmo IPH, uma heurística híbrida que utiliza programação inteira e um algoritmo *Record-to-Record* Aprimorado, proposto por Gulczynski et al. (2011) [**?**]. Na Tabela 4.4, o ILS-PVND é comparado com o o algoritmo HGSADC, uma adaptação da metaheurística Algoritmos Genéticos, utilizando buscas locais e um mecanismo de diversificação adaptativo da população, proposto por Vidal et al. (2011) [**?**]. Na Tabela 4.5, o ILS-PVND é comparado com o algoritmo LGC, um algoritmo exato baseado numa formulação de *Set Partitioning* e que utiliza relaxações do PRVP para resolvê-lo por um resolvedor de Programação Inteira, proposto por Baldacci et al. (2011)[**?**]. Nesta tabela alguns problemas-teste não foram reportados pelo autor. Por fim, na Tabela 4.6 o ILS-PVND é comparado com o algoritmo ITS, um algoritmo que combina conceitos das metaheurísticas *Iterated Local Search* e Busca Tabu, proposto por Cordeau e Maischberger (2012)[**?**]. Neste último trabalho, são apresentados resultados para a execução do ITS tanto de forma sequencial quanto de forma paralela, nesta tabela foram utilizados os resultados obtidos pela versão sequencial do algoritmo, deixando assim, as metodologias mais comparáveis. Os empates nos resultados entre a metodologia proposta e a Literatura são destacados em negrito, e os ganhos destacados com sublinhado.

Tabela 4.3: ILS-PVND × IPH - Gulczynski et al. (2011) [**?**]

Instância	**IPH**	**ILS-PVND**			
		Melhor	**GAP**	**Média**	**GAP Médio**
p01	524,61	524,61	**0,00**	524,61	**0,00**
p02	1335,08	1322,87	-0,91	1336,76	0,13
p03	524,61	524,61	**0,00**	524,61	**0,00**
p04	838,47	835,26	-0,38	842,06	0,43
p05	2052,81	2037,97	-0,72	2055,40	0,13
p06	838,47	835,26	-0,38	838,66	0,02
p07	827,39	826,14	-0,15	829,55	0,26
p08	2054,25	2040,78	-0,66	2053,63	-0,03
p09	827,39	826,14	-0,15	829,76	0,29
p10	1645,42	1604,26	-2,50	1624,69	-1,26
p11	781,68	779,29	-0,31	791,70	1,28
p12	1225,78	1202,72	-1,88	1228,78	0,25
p13	3624,77	3596,60	-0,78	3660,36	0,98
p14	954,81	954,81	**0,00**	954,81	**0,00**
p15	1862,63	1862,63	**0,00**	1862,63	**0,00**
p16	2875,10	2875,24	**0,00**	2875,24	**0,00**
p17	1597,75	1597,75	**0,00**	1608,01	0,64
p18	3215,43	3150,25	-2,03	3158,33	-1,78
p19	4845,97	4846,49	0,01	4846,49	0,01
p20	8369,72	8367,40	-0,03	8367,40	-0,03
p21	2189,00	2173,36	-0,71	2182,97	-0,28
p22	4317,17	4231,03	-2,00	4288,10	-0,67
p23	6685,07	6426,51	-3,87	6608,13	-1,15
p24	3741,98	3687,46	-1,46	3716,95	-0,67
p25	3817,08	3781,38	-0,94	3782,25	-0,91
p26	3794,95	3795,32	0,01	3796,43	0,04
p27	21946,89	21998,78	0,24	22196,46	1,14
p28	22384,04	22357,81	-0,12	22412,13	0,13
p29	22628,95	22667,95	0,17	22794,92	0,73
p30	75003,85	75352,63	0,47	76351,77	1,80
p31	76842,26	77145,98	0,40	78313,88	1,92
p32	78650,81	78740,91	0,11	80215,91	1,99
Médias			-0,58		0,17

Tabela 4.4: ILS-PVND × HGSADC - Vidal et al. (2011) [**?**]

Instância	**HGSADC**	**ILS-PVND**			
		Melhor	**GAP**	**Média**	**GAP Médio**
p01	524,61	524,61	**0,00**	524,61	**0,00**
p02	1322,87	1322,87	**0,00**	1336,76	1,05
p03	524,61	524,61	**0,00**	524,61	0,00
p04	835,26	835,26	**0,00**	842,06	0,81
p05	2024,96	2037,97	0,64	2055,40	1,50
p06	835,26	835,26	**0,00**	838,66	0,41
p07	826,14	826,14	**0,00**	829,55	0,41
p08	2022,47	2040,78	0,91	2053,63	1,54
p09	826,14	826,14	**0,00**	829,76	0,44
p10	1593,43	1604,26	0,68	1624,69	1,96
p11	770,89	779,29	1,09	791,70	2,70
p12	1186,47	1202,72	1,37	1228,78	3,57
p13	3492,89	3596,60	2,97	3660,36	4,79
p14	954,81	954,81	**0,00**	954,81	**0,00**
p15	1862,63	1862,63	**0,00**	1862,63	**0,00**
p16	2875,24	2875,24	**0,00**	2875,24	**0,00**
p17	1597,75	1597,75	**0,00**	1608,01	0,64
p18	3131,09	3150,25	0,61	3158,33	0,87
p19	4834,34	4846,49	0,25	4846,49	0,25
p20	8367,40	8367,40	**0,00**	8367,40	**0,00**
p21	2170,61	2173,36	0,13	2182,97	0,57
p22	4193,95	4231,03	0,88	4288,10	2,24
p23	6420,71	6426,51	0,09	6608,13	2,92
p24	3687,46	3687,46	**0,00**	3716,95	0,80
p25	3777,15	3781,38	0,11	3782,25	0,13
p26	3795,32	3795,32	**0,00**	3796,43	0,03
p27	21833,87	21998,78	0,76	22196,46	1,66
p28	22242,51	22357,81	0,52	22412,13	0,76
p29	22543,75	22667,95	0,55	22794,92	1,11
p30	73875,19	75352,63	2,00	76351,77	3,35
p31	76001,57	77145,98	1,51	78313,88	3,04
p32	77598,00	78740,91	1,47	80215,91	3,37
Médias			0,52		1,28

Tabela 4.5: ILSPVND × LCG - Baldacci et al. (2011) [**?**]

Instância	**LCG**	**ILS-PVND**			
		Melhor	**GAP**	**Média**	**GAP Médio**
p01	524,61	524,61	**0,00**	524,61	**0,00**
p02	1322,87	1322,87	**0,00**	1336,76	1,05
p03	524,61	524,61	**0,00**	524,61	0,00
p04	835,26	835,26	**0,00**	842,06	0,81
p05	2027,99	2037,97	0,49	2055,40	1,35
p06	835,26	835,26	**0,00**	838,66	0,41
p07	826,14	826,14	**0,00**	829,55	0,41
p08	2034,15	2040,78	0,33	2053,63	0,96
p09	826,14	826,14	**0,00**	829,76	0,44
p10	1593,45	1604,26	0,68	1624,69	1,96
p11	779,06	779,29	0,03	791,70	1,62
p12	-	1202,72	-	1228,78	-
p13	-	3596,60	-	3660,36	-
p14	954,81	954,81	**0,00**	954,81	**0,00**
p15	1862,63	1862,63	**0,00**	1862,63	**0,00**
p16	2875,24	2875,24	**0,00**	2875,24	**0,00**
p17	1597,75	1597,75	**0,00**	1608,01	0,64
p18	3136,69	3150,25	0,43	3158,33	0,69
p19	4834,34	4846,49	0,25	4846,49	0,25
p20	-	8367,40	-	8367,40	-
p21	2170,61	2173,36	0,13	2182,97	0,57
p22	4193,95	4231,03	0,88	4288,10	2,24
p23	-	6426,51	-	6608,13	-
p24	3687,46	3687,46	**0,00**	3716,95	0,80
p25	3777,15	3781,38	0,11	3782,25	0,13
p26	3795,32	3795,32	**0,00**	3796,43	0,03
p27	21912,85	21998,78	0,39	22196,46	1,29
p28	22242,51	22357,81	0,52	22412,13	0,76
p29	22543,76	22667,95	0,55	22794,92	1,11
p30	74464,26	75352,63	1,19	76351,77	2,53
p31	76322,04	77145,98	1,08	78313,88	2,61
p32	78072,88	78740,91	0,86	80215,91	2,74
Médias			0,28		0,79

Tabela 4.6: ILSPVND × ITS - Cordeau e Maischberger (2012) [**?**]

Instância	**ITS**	**ILS-PVND**			
		Melhor	**GAP**	**Média**	**GAP Médio**
p01	524,61	524,61	**0,00**	524,61	**0,00**
p02	1322,87	1322,87	**0,00**	1336,76	1,05
p03	524,61	524,61	**0,00**	524,61	**0,00**
p04	835,26	835,26	**0,00**	842,06	0,81
p05	2030,54	2037,97	0,37	2055,40	1,22
p06	835,26	835,26	**0,00**	838,66	0,41
p07	826,14	826,14	**0,00**	829,55	0,41
p08	2034,15	2040,78	0,33	2053,63	0,96
p09	826,14	826,14	**0,00**	829,76	0,44
p10	1594,11	1604,26	0,64	1624,69	1,92
p11	774,85	779,29	0,57	791,70	2,17
p12	1187,78	1202,72	1,26	1228,78	3,45
p13	3493,10	3596,60	2,96	3660,36	4,79
p14	954,81	954,81	**0,00**	954,81	**0,00**
p15	1862,63	1862,63	**0,00**	1862,63	**0,00**
p16	2875,24	2875,24	**0,00**	2875,24	**0,00**
p17	1597,75	1597,75	**0,00**	1608,01	0,64
p18	3142,80	3150,25	0,24	3158,33	0,49
p19	4834,34	4846,49	0,25	4846,49	0,25
p20	8367,40	8367,40	**0,00**	8367,40	**0,00**
p21	2184,03	2173,36	-0,49	2182,97	-0,05
p22	4193,95	4231,03	0,88	4288,10	2,24
p23	6516,71	6426,51	-1,38	6608,13	1,40
p24	3687,46	3687,46	**0,00**	3716,95	0,80
p25	3777,15	3781,38	0,11	3782,25	0,13
p26	3795,32	3795,32	**0,00**	3796,43	0,03
p27	21846,40	21998,78	0,70	22196,46	1,60
p28	22244,60	22357,81	0,51	22412,13	0,75
p29	22551,10	22667,95	0,52	22794,92	1,08
p30	74233,20	75352,63	1,51	76351,77	2,85
p31	76636,80	77145,98	0,66	78313,88	2,19
p32	78208,40	78740,91	0,68	80215,91	2,57
Médias			0,32		1,08

4.4 Análise dos Resultados

Ao observar os resultados apresentados na Tabela 4.2 quanto aos melhores resultados obtidos, verifica-se que o algoritmo proposto (ILS-PVND) consegue alcançar os resultados da Literatura em 13 dos 32 problemas-teste, e dentre os demais, em apenas dois casos o GAP para a melhor solução da Literatura ultrapassou os 2,00%. Além disso, em média, houve um GAP das melhores soluções de apenas 0,55%, o que mostra que, de modo geral, o método está muito próximo das melhores soluções obtidas pelos melhores algoritmos presentes na Literatura.

Observando-se os resultados médios, em 18 dos 32 problemas-teste, o algoritmo obteve resultados com GAP Médio abaixo de 1,0%, além de uma média dos GAPs de 1,31% em relação aos melhores resultados da Literatura.

O ILS-PVND se mostrou robusto, visto que apresentou desvios relativamente baixos, observando-se o Coeficiente de Variação. Em 21 dos 32 problemas-teste analisados, o desvio foi menor ou igual a 0,50%. No demais casos restantes, obteve o desvio máximo de 0,86%, mostrando que o algoritmo proposto é capaz de gerar soluções com baixa variabilidade em relação à qualidade.

Considerando a comparação com os algoritmos "estado da arte", observamos que o ILS-PVND se destaca em relação ao IPH (Tabela 4.3), conseguindo melhores resultados em 19 problemas-teste e empatando em seis. Nos demais, o maior GAP obtido foi de apenas 0,47%, obtendo um GAP médio das melhores soluções de -0,58%. Analisando os resultados médios em relação aos melhores resultados obtidos pelo IPH, observa-se ainda um ganho em 9 problemas-teste e empate em outros 5, obtendo soluções em média com um GAP de apenas 0,17%.

Na comparação dos melhores resultados encontrados pelo ILS-PVND em relação ao algoritmo HGSADC (Tabela 4.4), foram obtidos 14 empates, e em 12 dos problemas-teste um GAP menor que 1,0% foi obtido. E, em média, foi conseguido um GAP de apenas 0,52% em comparação com as melhores soluções do HGSADC. Observando os valores médios encontrados pelo ILS-PVND, foram obtidos 5 empates e em 12 problemas-testes foram conseguidos GAPs menores que 1,0%, obtendo em média 1,28% de GAP.

Ao observarmos a comparação da melhores soluções encontradas pela metodologia proposta com o algoritmo LCG (Tabela 4.5), temos 13 empates e um GAP máximo de 1,19%, obtendo um GAP médio de 0,28%. Quanto aos resultados médios, foram conseguidos 4 empates e na média dos GAPs Médios apenas 0,79%. Nota-se que em 4

problemas-teste os autores não reportaram os resultados obtidos.

Por fim, na comparação dos melhores resultados do ILS-PVND com o algoritmo ITS (Tabela 4.6), foram obtidos duas melhoras e 14 empates, um GAP máximo de 2,96%, porém, em média, obteve-se um GAP de apenas 0,32%. Ao observarmos as soluções médias, foi conseguido uma melhora, 6 empates, e 12 problemas-teste com GAP menor que 1,0%. Conseguindo assim um GAP médio de 1,08%.

Capítulo 5

Conclusões e trabalhos futuros

Este trabalho abordou o Problema de Roteamento de Veículos Periódico (PRVP). Devido ao caráter combinatório do problema, sua resolução por métodos exatos é, em muitos casos, computacionalmente inviável. Assim, para sua resolução foi proposto um algoritmo heurístico, denominado ILS-PVND.

O ILS-PVND é um algoritmo *Multistart* que combina a metaheurística *Iterated Local Search*, a heurística *Variable Neigborhood Descent* e o método *Pattern Improvement*. Para a exploração do espaço de soluções foram utilizadas Estruturas de Vizinhança típicas do Problema de Roteamento de Veículos Clássico, aplicadas dia-a-dia na solução.

Para validar o algoritmo proposto, um conjunto de 32 problemas-teste foram usados e os resultados obtidos comparados aos da Literatura. Os experimentos computacionais mostram que a abordagem proposta é competitiva, pois consegue obter bons resultados, empatando em 13 problemas-teste e tendo uma GAP médio das melhores soluções de apenas 0,55% e das soluções médias de 1,31%. Além disso, apresentou um baixo Coeficiente de Variação, que mostra que o algoritmo é robusto, e gera soluções com baixa variabilidade.

Além dessa comparação geral, foram feitas comparações individuais com os 4 algoritmos "estado da arte" presentes na Literatura. Nessa comparação, observa-se que o ILS-PVND apresenta bons resultados comparados às 4 metodologias, encontrando soluções melhores comparadas à uma delas, e nas demais obtendo um GAP médio das melhores soluções e das soluções médias de, no máximo, 0,52% e 1,28%, respectivamente.

É importante ressaltar que o ILS-PVND pode ser considerado como o mais simples dentre os algoritmos apresentados, visto que para sua execução somente dois parâmetros devem ser definidos, $iterMax$ e δ para o POP. Nos demais algoritmos, há um grande

número de parâmetros e técnicas que os tornam mais complexos e também menos reprodutíveis. Além dessa contribuição, destaca-se também a criação de uma nova geração de solução inicial específica para o Problema de Roteamento de Veículos Periódico, denominada Geração de Solução Inicial Baseada na Inserção Mais Barata por Padrões.

Como trabalhos futuros, almeja-se incorporar um método de programação matemática ao ILS-PVND, tanto para melhorar os resultados obtidos, quanto para reduzir o tempo computacional gasto na execução dos testes. Além disso, outras técnicas também podem ser incorporadas, como a paralelização utilizando GPU (*Graphics Processing Unit*), pois como várias execuções ocorrem simultaneamente, diversifica-se o espaço de busca e com isso tenta-se melhorar as soluções geradas, ou Mineração de Dados (*Data Mining*), para encontrar clientes que frequentemente estão alocados à algum Padrão de Visita e com isso fixá-los nesse padrão, fazendo com que a complexidade na resolução do PRVP seja diminuída.

Além disso, pretende-se abordar o PRVPES e o PRVPMD buscando validar a metodologia proposta em problemas que apresentem as mesmas características do PRVP. Tratando primeiramente um problema que há uma generalização do PRVP Clássico, onde a escolha do Padrão de Visita fica à critério do algoritmo, PRVPES. Após isso pretende-se tratar um problema que adiciona mais uma característica ao PRVP, a de possuir múltiplos depósitos, PRVPMD, aumentando ainda mais a complexidade na sua resolução.