UNIVERSIDADE FEDERAL FLUMINENSE

DANIEL MARTINS DA SILVA

Uma Heurística para o Problema de Roteamento de Veículos com Múltiplas Viagens

NITERÓI

2012

UNIVERSIDADE FEDERAL FLUMINENSE

DANIEL MARTINS DA SILVA

Uma Heurística para o Problema de Roteamento de Veículos com Múltiplas Viagens

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

Orientador:
Yuri Abitbol de Menezes Frota

NITERÓI

2012

Daniel Martins da Silva

Uma Heurística para o Problema de Roteamento de Veículos com Múltiplas Viagens

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

Aprovada em Agosto de 2012.

BANCA EXAMINADORA

Prof. D.Sc. Yuri Abitbol de Menezes Frota / IC-UFF / Orientador

---

Prof. D.Sc. Luiz Satoru Ochi / IC-UFF

Profa. D.Sc. Márcia Helena da Costa Fampa / COPPE-UFRJ

Niterói

2012

Á minha familia, aos meus amigos.

Agradecimentos

Obrigado Senhor pelas experiências que me permitiu passar, pelas orientações e conselhos que me destes e pelos amigos que me apresentastes nessa jornada.

Aos amigos Marcos, Eyder, Ivan, Rian, Juliano e Leonardo pelas conversas e apoio.

Ao amigo Satoru pela confiança e pelos constantes incentivos durante toda essa jornada acadêmica.

Ao amigo Yuri pela paciência e orientações.

Ao amigo Anand a quem devo muito pela motivação, orientação e constante ajuda.

Aos meus pais, Guimarães e Marlene e minha irmã Eduarda pelo apoio e força, sem vocês nada disso seria possível.

À minha esposa Adriana, pela sua ajuda e dedicação.

Muito Obrigado Jesus por poder conhecer estes amigos.

Resumo

O Problema de Roteamento de Veículos (PRV) é um problema clássico da otimização onde há um depósito possuindo ofertas de produtos e um conjunto de clientes cujas demandas necessitam ser atendidas. O objetivo do problema é atender as demandas dos clientes minimizando os custos dos deslocamentos. O Problema de Roteamento de Veículos com Múltiplas Viagens (PRVMV) é uma extensão do PRV que se diferencia do problema original por permitir que um veículo atenda mais de uma rota durante um período de planejamento. Este trabalho propõe um algoritmo de duas fases para solucionar o PRVMV. Primeiro, um procedimento baseado na metaheurística *Iterated Local Search* (ILS) combinada com o método *Random Variable Neighborhood Descent* (RVND) que gera conjuntos de rotas que depois são atribuídas a veículos utilizando-se uma heurística inspirada na concepção do *Bin Packing Problem* (BPP). Resultados computacionais são apresentados para um conjunto de instâncias clássicas da literatura.

Palavras-chave: Problema de Roteamento de Veículos com Mútiplas Viagens. Metaheurística. Problema de Empacotamento.

Abstract

The Vehicle Routing Problem (VRP) is a classical optimization problem where there is a depot with an available supply of goods and a set customers whose demands must be satisfied. The objective is to meet the customers demands minimizing the travel costs. The Vehicle Routing Problem with Multiple Trips (VRPMT) is an extension of the VRP where the vehicle is allowed to be assigned to multiple routes in a given time horizon. This work proposes a two-phase algorithm for the VRPMT. Firstly, procedure based on the *Iterated Local Search* (ILS) metaheuristic combined with a *Random Variable Neighborhood Descent* method generate sets of routes that are later assigned to vehicles by using a heuristic inspired on the *Bin Packing Problem* (BPP) concept. Computational experiments are reported for a set of well-known instances from the literature.

Keywords: The Vehicle Routing Problem with Multiple Trips. Metaheuristic. Bin Packing.

Sumário

Lista de Abreviaturas e Siglas

PRV	:	Problema de Roteamento de Veículos;
PRVMV	:	Problema de Roteamento de Veículos com Mútiplas Viagens;
PRVJT	:	Problema de Roteamento de Veículos com Janela de Tempo;
PRVCE	:	Problema de Roteamento de Veículos com Coleta e Entrega;
ILS	:	*Iterated Local Search*;
VND	:	*Variable Neighborhood Descent*;
RVND	:	*Random Variable Neighborhood Descent*;
BPP	:	*Bin Packing Problem*;
VRPMT	:	*Vehicle Routing Problem with Multiple Trips*;
LTR	:	*Longest Tour Ratio*;
PC	:	*Penalized Cost*;
FF	:	*First Fit*;
BF	:	*Best Fit*;
FFD	:	*First Fit Decreasing*;
BFD	:	*Best Fit Decreasing*;
DVRP	:	*Distance Vehicle Routing Problem*;
CDVRP	:	*Capacity and Distance Vehicle Routing Problem*;
HC	:	Heurística Construtiva;
HR	:	Heurística de Refinamento;

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A expansão da demanda por bens tem provocado um significativo crescimento no setor logístico. Este setor engloba o processo de planejamento, operação e controle do fluxo de produtos, por meio do uso intenso de informações. Dentre os custos relacionados com a logística das empresas a maior fatia está empregada no setor de transportes[11].

Os custos logísticos no Brasil no ano de 2008, segundo a Confederação Nacional do Transporte (CNT), atingiram um valor equivalente a 11,6% do Produto Interno Bruto (PIB), ou seja, R$ 349,0 bilhões. O item com maior representatividade foi o transporte, com 6,9% do PIB (207,0 bilhões). Devido a sua relevância econômica, estudos relacionados ao transporte são de vital importância para a competitividade de um país.

Um dos problemas de transporte mais pesquisados da área de Pesquisa Operacional é o Problema de Roteamento de Veículos (PRV), que tem como objetivo minimizar os custos relacionados com transportes, utilizando uma frota de veículos homogênea a partir de um depósito, para atendimento de um conjunto de clientes.

Para obter uma solução viável do PRV, certas restrições devem ser respeitadas: os veículos não podem ultrapassar o seu limite de carga; o início e o fim de cada rota tem como ponto de partida e retorno o depósito; os clientes são atendidos por apenas um único veículo; e cada veículo por sua vez atende apenas uma rota durante um período de planejamento.

O primeiro trabalho a tratar sobre este tema de roteamento de veículos foi datado por Dantzig e Ramser[10] em 1959, onde houve um estudo sobre a distribuição de combustível em postos de gasolina. Desde então diversas abordagens vêm sendo estudadas, cada qual com suas peculiaridades, como por exemplo: Problema de Roteamento de Veículos com Janela de Tempo (PRVJT), Problema de Roteamento de Veículos com Coleta e Entrega

(PRVCE) e Problema de Roteamento de Veículos com Múltiplas Viagens (PRVMV).

No PRVJT além das restrições do PRV, são inseridas restrições temporais, onde o atendimento aos clientes acontece em intervalos temporais específicos. Já no PRVCE as encomendas dos clientes podem ser divididas em duas partes, sendo a entrega realizada em um local e a coleta em outro.

O PRVMV é uma derivação do PRV na qual a última restrição é modificada permitindo múltiplas viagens. Com isso, um veículo pode atender mais que uma rota durante um período de planejamento. Este problema é o foco deste trabalho, conhecido na literatura como *Vehicle Routing Problem with Multiple Trips* (VRPMT) pertencente à classe dos NP-Completos.

Problemas pertencentes a esta classe necessitam de um grande número de operações para sua resolução. Encontrar uma solução viável, isto é, uma solução que atenda todas as restrições do problema, não é uma tarefa trivial ou até mesmo possível. Por este motivo foram criadas certas métricas para mensurar a qualidade das soluções inviáveis que são: *Overtime* (Horas Extras), *Longest Tour Ratio* e *Penalized Cost*. O presente trabalho prioriza a minimização das horas extras em detrimento das demais métricas, a fim de comparar com valores obtidos a partir da literatura [31], [35] e [40].

1.1 Motivação

O PRVMV tem grande importância na área de logística das empresas, em específico no setor de transporte, pois nem sempre a duração das rotas termina juntamente com a jornada de trabalho, ocasionando ociosidade dos veículos. Essa situação se verifica em sua maioria nas empresas da zona urbana no serviço de tele entrega, onde o tempo do atendimento dos itinerários é curto.

Estudos relacionados com a melhor resolução desse problema são de vital importância na gestão das empresas. Do ponto de vista acadêmico, o desenvolvimento de um *Iterated Local Search* (ILS) combinado com o *Random Variable Neighborhood Descent* (RVND) para resolver tal problema é de nosso conhecimento, uma maneira inédita.

1.2 Objetivos do trabalho

Este trabalho tem como objetivo geral a integração do algoritmo ILS-RVND [34], que consiste em uma combinação entre a metaheurística *Iterated Local Search* (ILS) e o método *Random Variable Neighborhood Descent* (RVND), com a heurística *Best Fit Decreasing* (BFD) para resolver o *Bin Packing Problem* (BPP) e solucionar o Problema de Roteamento de Veículos com Múltiplas Viagens.

Os objetivos específicos a serem atingidos incluem:

- efetuar uma revisão bibliográfica detalhada e atualizada sobre o PRVMV, além de uma revisão das metodologias utilizadas para esse problema;
- coletar problemas-teste na literatura para PRVMV;
- estudar heurísticas construtivas e de refinamento;
- avaliar o desempenho do algoritmo desenvolvido, comparando com os melhores resultados da literatura.

1.3 Organização do texto

Este trabalho está organizado em cinco capítulos incluindo esta introdução. O Capítulo 2 define a notação utilizada, apresenta uma revisão de literatura e a formulação matemática do PRVMV. Neste capítulo são discutidas as heurísticas e metaheurísticas empregadas, e apresentadas as medidas alternativas para mensurar a inviabilidade da solução.

O Capítulo 3 expõe de forma detalhada as características de implementação do algoritmo ILS-RVND-BFD. Neste capítulo são apresentados os procedimentos para construir uma solução inicial, as estruturas de vizinhança Inter-Rota e Intra-Rota e os mecanismos de perturbação randômica, concluindo com um detalhamento do *Bin Packing Problem*.

No Capítulo 4 são descritos os experimentos computacionais realizados, onde estes são avaliados e discutidos. Por fim, o Capítulo 5 apresenta as considerações finais e trabalhos futuros.

Capítulo 2

Problema de Roteamento de Veículos com Múltiplas Viagens

O Problema de Roteamento de Veículos com Múltiplas Viagens é uma generalização do clássico PRV com certas propriedades particulares, onde a restrição de exclusividade de atendimento do veículo para com uma rota é modificada, ou seja, permite que um veículo atenda múltiplas viagens durante um período de planejamento com uma frota fixa.

Situações dessa natureza que permitem o atendimento de múltiplas viagens ocorrem com maior frequência nos centros urbanos, onde os itinerários são mais curtos em comparação a jornada de trabalho, provocando ociosidade da frota. Para reduzir o tempo parado dos veículos no deposito e até mesmo minimizar o número de unidades de transporte é que a resolução eficiente do PRVMV é tão importante para o setor logístico.

2.1 Definição do Problema

O PRVMV pode ser definido sobre um grafo não-orientado $G = (V, E)$, em que $V = \{0, 1, ..., n\}$ representa o conjunto de vértices do grafo, (assumimos que 0 é o depósito) e os nós $i \in V \setminus \{0\}$ representam os clientes, cada qual com sua demanda q_i. Cada aresta $E = \{(i, j)|i, j \in V, i < j\}$ tem um custo associado c_{ij} e um tempo t_{ij}, ambos não negativos.

O custo de transporte de mercadorias envolve um conglomerado de fatores como distância, tempo, veículo, volume da mercadoria, entre outros. No entanto, neste presente trabalho assume-se que o custo de viagem é igual ao tempo ($c_{ij} = t_{ij}$).

A frota de veículos $K = \{1, ..., m\}$ é fixa, onde m representa o número de veículos,

cada qual com sua capacidade homogênea Q. Define-se também, um horizonte de tempo T que determina o período de planejamento.

Seja $r = (v_0, v_1, ..., v_{n(r)+1})$ uma rota, onde $v_0 = v_{n(r)+1} = 0$ e $n(r)$ é definido como o número de clientes visitados em r. Considere as seguintes notações:

- $q_r = \sum_{i=1}^{n(r)} q_{v_i}$ é a demanda coberta pela rota (cada vértice tem demanda q).
- $c_r = \sum_{i=0}^{n(r)} c_{v_i v_{i+1}}$ é o custo da rota (c_{ij} é o custo do arco ij).
- $t_r = \sum_{i=0}^{n(r)} t_{v_i v_{i+1}}$ é a duração da rota.

Tais elementos do problema caracterizam o PRVMV que tem como principal objetivo construir um conjunto de rotas e atribuí-las a uma frota de veículos, minimizando o custo total de deslocamento e satisfazendo as seguintes restrições:

1. ciclo: cada rota deve iniciar e terminar no depósito;
2. exclusividade: cada cliente é visitado exatamente por uma rota;
3. capacidade: o somatório das demandas dos clientes pertencentes a uma rota não deve exceder a capacidade do veículo Q ($q_r \leq Q$);
4. duração: cada veículo pode atender uma ou mais rotas, desde que a duração total das rotas não ultrapasse o horizonte de tempo.

Caso as restrições de ciclo, exclusividade, capacidade e duração sejam atendidas a solução é viável. Porém, caso alguma dessas restrições seja violada a solução é inviável. Uma solução para o PRVMV pode ser definida como uma família de conjuntos de rotas:

$$S = (R_1, ..., R_m)$$

onde cada R_k contém as rotas do veículo $k \in K$. Define-se também, R como o conjunto de todas as rotas viáveis da solução, representada pela união de todos R_k:

$$R = \bigcup_{k \in K} R_k.$$

Além disso, o custo de uma solução:

$$f(s) = \sum_{r \in R} c(r)$$

é definido como o somatório dos custos de cada rota. Por fim, considere o símbolo t_{r_k} como sendo a duração total das rotas atribuídas ao veículo k:

$$t_{r_k} = \sum_{r \in R_k} t_r.$$

2.2 Revisão de Literatura

Inicialmente o PRVMV foi abordado por Hommes[25] que realizou estudos empíricos na indústria *Biscuits Burton Ltd.*. Foi observado que com a modificação da restrição de exclusividade de atendimento do veículo durante um período de planejamento, o número de veículos utilizados foi reduzido de 21 para 19, significando uma redução de custo de 5% nas entregas.

Logo depois, em 1990, Fleischmann[16] propôs uma heurística de *Bin Packing* com uma estratégia de *Best Fit Decreasing* (BFD) para agrupar as rotas geradas em uma solução para o PRVMV. Taillard[45], fez uso da ideia de Fleischmann combinada com a metaheurística Busca Tabu (BT). O algoritmo apresentado tem três etapas. Primeiramente é gerado um grande conjunto de soluções viáveis que satisfazem o PRV clássico. Em seguida é feita uma seleção de um subconjunto de rotas de acordo com o algoritmo enumerativo e por último, são reunidas todas as rotas selecionadas utilizando um algoritmo de empacotamento.

Brandão e Mercer[4] abordaram novamente o sistema de distribuição da empresa *Biscuits Burton Ltda* adicionando restrições extras como a janela de tempo de atendimento aos clientes, frota heterogênea, contratação de veículos no caso de insuficiência e limites legais de jornada de trabalho. Resultados mostraram que o algoritmo superou a programação manual utilizada pela empresa em cerca de 20%.

No ano de 2002, Petch e Salhi[35] desenvolveram uma heurística construtiva multi-fase que cria uma solução inicial para PRV utilizando a heurística de Clark e Wrigth[7] conhecida como método das economias. Para melhorar a solução, foram utilizadas heurísticas de refinamento (*Exchange*, *2-optimal* e *3-optimal*) e agrupamento de rotas com o objetivo de minimizar as penalidades utilizando o processo *Bin Packing*.

Ainda em 2002 Zhao, Wang, Lai e Xia[47] utilizaram um modelo de programação linear inteira mista e um algoritmo de Busca Tabu para gerar soluções para o PRVMV. Nesse caso, uma discussão mais aprofundada mostra que o uso de múltiplas viagens não é sempre rentável.

Em 2004, Alfredo e Omar[31] empregaram um algoritmo de memória adaptativa para solucionar o PRVMV, na qual se tem uma memória composta por um multiconjunto de rotas e a cada nova iteração, uma nova solução não determinística é gerada, melhorada utilizando BT e adicionada na memória. Depois disso é utilizada a ideia da heurística de empacotamento com a estratégia BFD para agrupar as rotas.

Petch e Salhi[40] em 2007 desenvolveram um algoritmo genético híbrido que não utiliza a representação binária do cromossomo. A estrutura do algoritmo é iniciada com a geração de uma população inicial de cromossomos e depois com um processo recursivo em que novos cromossomos são criados e melhorados com os operadores genéticos. Posteriormente, utilizou-se a ideia de Fleischmann para clusterização das rotas.

Algumas abordagens atuais incorporam a ideia do PRVMV com algumas características particulares, como no caso do trabalho de Battarra, Monaci e Vigo[3] que tem como função minimizar o número de veículos. Este problema é conhecido na literatura como *Minimum Multiple Trip Vehicle Routing Problem* (MMTVRP), rompendo com o clássico PRVMV que tem como principal objetivo minimizar o custo de roteamento. Outro trabalho interessante é o de Yong, Jing e Hongbo[33], que abordam o PRVMV sobre uma ótica baseada na redução do consumo de combustível.

No período de 2010, alguns trabalhos utilizaram métodos exatos para resolver variantes do PRVMV. Podemos destacar a pesquisa de Hernandez, Feillet, Giroudeau e Naud[24] que utiliza métodos exatos para solucionar o problema de roteamento de veículos com múltiplas viagens e janela de tempo. Azi, Gendreau e Potvin[2] também abordaram este mesmo problema utilizando a técnica de geração de colunas em um método *Branch-and-Price*. No entanto, este algoritmo é limitado pelo tamanho da instância, resolvendo problemas com até 40 clientes apenas.

Em 2012, R. Roberti[38] propõe uma abordagem exata para solucionar o PRVMV baseada em duas formulações de particionamento que conseguem resolver na otimalidade 42 dos 52 casos referenciados para instâncias da literatura com até 120 clientes.

2.3 Formulação Matemática

A formulação matemática do PRVMV como um problema de programação inteira, introduzida em [31] é apresentada pelas expressões (2.1) a (2.4) a seguir. Nessa formulação X_r^k assume o valor de 1 se a rota r está na solução e é atribuída ao veículo k, caso contrário a variável assume o valor de 0. O coeficiente a_{ir} tem o valor 1 se a rota $r \in R$ visitou o cliente $i \in V \setminus \{0\}$ e 0, caso contrário.

$$PRVMV = Min \sum_{k \in K} \sum_{r \in R} c_r X_r^k \tag{2.1}$$

$$s.a. \sum_{k \in K} \sum_{r \in R} a_{ir} X_r^k = 1, \forall i \in V \setminus \{0\} \tag{2.2}$$

$$\sum_{r \in R} t_r X_r^k \leq T, \forall k \in K \tag{2.3}$$

$$X_r^k \in \{0,1\}, \forall k \in K, \ \forall r \in R \tag{2.4}$$

A formulação matemática do $PRVMV$ tem uma função objetivo (2.1) que busca minimizar o custo total do deslocamento. As restrições (2.2) determinam que cada cliente deve pertencer a exatamente uma rota. As restrições (2.3) asseguram que a duração das rotas atribuídas a cada veículo não deve exceder o período de planejamento. As restrições (2.4) definem o domínio das variáveis.

2.4 Decomposição do PRVMV

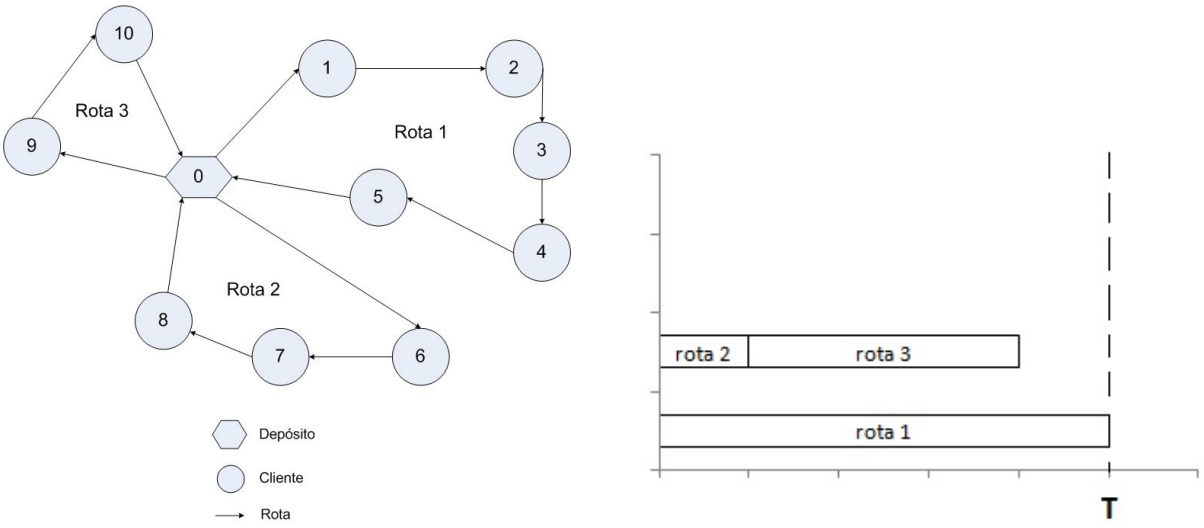
O PRVMV em suma é constituído por um conjunto de rotas a serem atribuídas a uma frota de veículos (K). Uma possível maneira de abordar o PRVMV é visualizar o problema como a solução de dois outros subproblemas definidos como: o Problema de Roteamento de Veículos (PRV) e o Problema de Atribuição de Rotas (PAR).

- O PRV tem a intenção de gerar um conjunto de rotas (R) que satisfaça as restrições de exclusividade, capacidade e ciclo. Um exemplo de sua resolução pode ser visua-

lizada na Figura 2.1(a).

- O PAR tem por finalidade atribuir um conjunto de rotas a uma frota fixa de veículos atendendo a restrição de duração. Sua resolução está ilustrada na Figura 2.1(b).

Com a resolução desses dois subproblemas vemos que é possível criar uma solução para o PRVMV. Esta solução do PRVMV pode ser representada por uma permutação de clientes que são numerados de 1 a n e divididos em k rotas, onde essas rotas são atribuídas a uma frota de m veículos. Por exemplo, na Figura 2.1 temos 10 clientes a serem atendidos, 3 rotas e 2 veículos disponíveis, então uma possível solução é [0 1 2 3 4 5 0] e [0 6 7 8 0],[0 9 10 0]. O primeiro veículo atende a rota 1 ([0 1 2 3 4 5 0]) e o segundo as rotas 2 e 3 ([0 6 7 8 0] e [0 9 10 0]). Nas rotas atendidas pelo segundo veículo, o trajeto começa no depósito, atende os clientes 6, 7 e 8 nessa ordem e depois retorna ao depósito, para então começar o atendimento da próxima rota. Vale ressaltar que não é considerado o tempo em que um veículo permanece parado no depósito durante o intervalo de atendimento entre duas rotas.



(a) Roteamento de Veículos (b) Atribuição de Rotas

Figura 2.1: Exemplo de uma solução do PRVMV

2.5 Complexidade Computacional

Encontrar uma solução ótima para o PRVMV não é uma tarefa trivial, pois é necessário um grande número de operações para sua resolução. Resolver o PRVMV implica solucionar dois subproblemas: o problema de roteamento de veículos e o problema de atribuição de

rotas (na literatura conhecido como problema de empacotamento) e estes têm elevada complexidade computacional [19].

O PRV pertence à classe NP-difícil e o PAR pertence à classe NP-completo todas as duas classes subconjunto da classe NP que consiste no conjunto de todos os problemas de decisão que podem ser resolvidos em tempo polinomial. No entanto, não foi encontrado algoritmos polinomiais que solucione todos os problemas de NP por isso acredita-se que NP $>$ P (polinomial).

Os problemas que necessitam de tempo exponencial para sua resolução são ditos intratáveis ou NP-completos. Por outro lado, problemas que têm algoritmos eficientes onde uma solução ótima é encontrada em tempo polinomial são considerados tratáveis ou de classe P .

Como o PRVMV pertence à classe de problemas NP-Completo (em sentido forte), a sua resolução através de abordagens exatas, torna-se menos indicada devido ao grande esforço computacional necessário para determinar uma solução. Além disso, o número de vértices (clientes) das instâncias em que os métodos exatos conseguem encontrar a otimalidade em um tempo aceitável é bem reduzido, o que inviabiliza sua utilização em situações práticas.

Apesar desta limitação dos métodos exatos terem diminuído com o passar dos anos devido ao aumento do nível de processamento dos computadores, ainda está muito distante da curva exponencial apresentada por este problema. Portanto, abordagens heurísticas e metaheurísticas continuam a ser alternativas mais viáveis para a resolução do PRVMV, pois apresentam boas soluções em um tempo aceitável.

2.6 Heurísticas

As heurísticas são procedimentos que aceitam uma solução aproximada, sendo aplicada na maioria dos casos em problemas onde o número de soluções possíveis é muito elevado ou até infinito. Como principais vantagens da utilização da heurística podemos citar a resolução de problemas de grande porte e sua flexibilidade (facilidade em se adaptar aos dados de entrada).

Podemos categorizar as heurísticas em dois grupos distintos, Heurísticas Construtivas (HC) e Heurísticas de Refinamento (HR). As HC têm por objetivo construir uma solução viável, elemento por elemento, onde a forma em que estes elementos são inseridos na

solução muda de acordo com um critério de aceitação e uma função objetivo. Por outro lado, as HR partem de uma solução obtida pelas HC, e a cada iteração, caminham de solução vizinha para solução vizinha (de acordo com a definição de vizinhança escolhida), tentando melhorar a função objetivo.

No decorrer dos anos, várias HC foram desenvolvidas para solucionar o PRV clássico. Entre elas, destacam-se as heurísticas de Inserção mais Próxima, Inserção mais Barata e Método das Economias. No conjunto das HR, as que sobressaem são as heurísticas do tipo *k-opt*, *Enchange*, *Shift* e *Swap* [7].

As HC e HR, embora flexíveis, em muitos casos possuem um comportamento instável. Além disso, nos problemas de otimização, as heurísticas tendem frequentemente a encontrar apenas ótimos locais que na maioria das vezes, estão distantes dos ótimos globais. Para tentar reduzir as limitações das heurísticas sugiram as metaheurísticas.

2.7 Metaheurísticas

As Metaheurísticas são heurísticas de aprimoramento com mecanismos complementares que tentam escapar dos ótimos locais. São aplicadas em sua maioria em problemas de elevada complexidade computacional (NP-Completo ou NP-Difícil). Algumas das mais conhecidas são *Genetic Algorithms* (GAs) [20], *Tabu Search* (TS) [21], *Neural Networks* (NNs) [8], *Simulated Annealing* (SA) [46], *Greedy Randomized Adaptive Search Procedure* (GRASP) [37], *Variable Neighborhood Search* (VNS) [22], *Iterated Local Search* (ILS) [29] e etc.

De acordo com Fred Glover [17], metaheurísticas são métodos de solução que orquestram uma iteração entre os processos de melhoria e estratégias locais de nível superior para criar um processo capaz de escapar de ótimos locais e realizar uma pesquisa robusta de um espaço de solução. Nas subseções a seguir são descritas as metaheurísticas utilizadas neste trabalho.

2.7.1 Metaheurística ILS

Iterated Local Search[29] é uma metaheurística que explora o espaço de soluções aplicando sucessivas buscas locais e modificações ou perturbações à solução ótima local encontrada. O mecanismo de perturbação utilizado tem função determinante na eficácia do método, pois tem a finalidade de escapar de ótimos locais, diversificar a busca, e encontrar melhores

soluções.

As perturbações não podem ser demasiadamente pequenas e nem grandes. Se elas são muito pequenas o processo de diversificação da solução é penalizado ou seja, não consegue escapar dos ótimos locais. Do contrário, quando as perturbações são muito grandes a solução se tornará completamente aleatória e não haverá tendência na amostragem.

Na Figura 2.2 é apresentado o funcionamento da perturbação e do procedimento de busca local, onde a solução começa com o mínimo s^*, e após a aplicação da perturbação a solução é conduzida para $s^{'}$. Depois que a busca local é empregada, um novo mínimo local $s^{*'}$ é encontrado, podendo ser melhor que s^*.

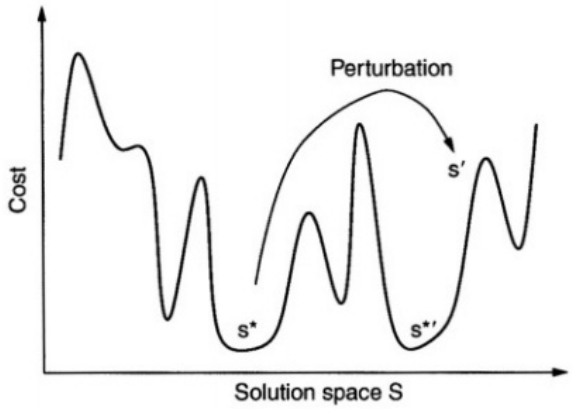


Figura 2.2: Exemplo de pertubação (Fonte: Lourenço et al. [29])

O pseudocódigo da metaheurística ILS é esquematizado no Algoritmo 1. O método começa seu processamento com a geração de uma solução inicial s_0 (linha 2) que pode ser aprimorada por uma busca local (linha 3) e encontrar uma nova solução s^*. Durante $IterILS$ iterações (linha 4-8) ocorrem sistemáticas perturbações e buscas locais sobre a melhor solução corrente. Se $s^{*'}$ atender aos critério de aceitação (linha 7) torna-se o próximo elemento a ser explorado sistematicamente, caso contrário s^* será novamente pertubado e explorado. No final do algoritmo, a melhor solução s^* (linha 9) é retornada.

De acordo com o Algoritmo 1, podemos notar quatro componentes principais que determinam o bom desempenho do método ILS: GeraSolução, BuscaLocal, Perturbação e CriterioAceitação. Todos estes componentes são modularmente independentes e de fácil entendimento. Isso é uma das vantagens da utilização da metaheurística ILS.

Na literatura, o ILS está sendo empregado em diversos setores, desde problemas acadêmicos até aplicações comerciais e industriais. Podemos destacar as pesquisas de (Cotta et al. [9]), (Stützle e Hoos [41]), (Dong, Huang e Chen [12]), (Paquete, Stützle e Thomas [32]) , (Hashimoto, Yagiura e Ibaraki [23]), (Ibaraki et al. [26]), (Subramanian

Algoritmo 1 ILS

```
1: ILS(IterILS)
2: s_0 ← GeraSolução;
3: s* ← BuscaLocal(s_0);
4: enquanto Iter ≤ IterILS faça
5:    s' ← Perturbação(s*, historico);
6:    s*' ← BuscaLocal(s');
7:    s* ← CriterioAceitação(s*, s*', historico);
8: fim enquanto
9: return s*
10: end ILS
```

[42]), (Ayadi et al.[1]), (Dorronsoro et al. [14]), (Liddle e Thomas [28]), (Brito et al. [5]) e (Subramanian e Battarra [43]).

2.7.2 Metaheurística VND

A estratégia de refinamento *Variable Neighborhood Descent* (VND) baseia-se na ideia de Mladenovic e Hansen[30] que consiste em explorar exaustivamente o espaço de soluções por meio de trocas sistemáticas de vizinhança no processo de busca local. Um dos motivos da sua eficiência em comparação com os métodos tradicionais de busca local é que ao invés de utilizar uma única estrutura de vizinhança, várias estruturas são utilizadas.

O algoritmo VND tem três princípios fundamentais:

1. um ótimo local com relação a uma dada estrutura de vizinhança não corresponde necessariamente a um ótimo local com relação a uma outra estrutura de vizinhança;

2. um ótimo global é um ótimo local em relação a todas as estruturas possíveis de vizinhança;

3. para muitos problemas, ótimos locais em relação a uma ou várias estruturas de vizinhanças são relativamente próximos.

Os princípios destacados sugerem o uso de várias estruturas de vizinhança para resolução dos problemas de otimização. O objetivo agora é determinar o conjunto de estruturas que serão utilizadas e a forma que serão aplicadas. Na versão clássica do VND as estruturas de vizinhança seguem uma ordem pré-estabelecida. Ordem esta que não é seguida pelo o método RVND [44] empregado nesse trabalho que faz uma escolha aleatória das estruturas de vizinhanças utilizadas.

2.8 Estruturas de Vizinhança

Em suma, as estruturas de vizinhança são mecanismos utilizados para tentar aprimorar a melhor solução corrente. A Tabela 2.1 apresenta uma breve descrição das estruturas de vizinhança empregadas nesse trabalho que serão detalhadas posteriormente no Capítulo 3. As seis primeiras estruturas de vizinhança exibidas na tabela citada, correspondem aos movimentos que ocorrem entre rotas distintas da solução, enquanto que as demais estruturas correspondem aos movimentos internos na mesma rota.

Tabela 2.1: Estruturas de Vizinhança utilizadas na resolução do PRVMV

Vizinhança	Descrição
Shift(1,0)	consiste em transferir um cliente de uma rota para outra.
Shift(2,0)	consiste em transferir dois clientes consecutivos de uma rota para outra.
Swap(1,1)	permutação entre dois clientes pertencentes a rotas diferentes.
Swap(2,2)	dois clientes consecutivos de uma rota são permutados com dois clientes consecutivos de outra rota.
Swap(2,1)	dois clientes consecutivos de uma rota são permutados com um cliente de outra rota.
Cross	consiste na remoção de dois arcos de rotas distintas e a inserção de outros dois de modo a gerar uma nova solução.
Reinserção	mudança de um cliente de uma posição para outra posição da rota.
Or-opt2	realocação de dois clientes adjacentes de uma posição para outra posição da rota.
Or-opt3	três clientes adjacentes são removidos e inseridos em uma outra posição.
2-opt	remoção de dois arcos não adjacentes enquanto outros dois arcos são adicionados de tal forma que uma nova rota é gerada.
Exchange	permutação de dois clientes.

Os movimentos podem ocorrer em sua maioria de duas formas: *First Improving* e *Best Improving*. Na primeira, logo que é encontrada uma solução vizinha com um custo melhor que a solução corrente, o movimento é executado. No caso da estratégia *Best Improving* todas as soluções vizinhas são investigadas e a solução corrente é alterada, com o movimento que ocasiona menor custo [36]. No entanto, nem sempre é possível melhorar ou mesmo encontrar soluções viáveis para o problema, ou seja, que satisfaçam todas as restrições do problema (descritos na seção 2.1). Por este motivo foram desenvolvidas as medidas de inviabilidade.

2.9 Medidas de Inviabilidade

Normalmente empregadas quando o problema tem um alto grau de complexidade, as medidas de inviabilidade são métricas que mensuram a qualidade de uma solução inviável. Taillard[45] propõem três maneiras de quantificar esta inviabilidade para o PRVMV: *Overtime* (Horas Extras), *Longest Tour Ratio* (Razão da Duração da Viagem) e *Penalized Cost* (Custo mais Penalidade).

Vale ressaltar que as métricas propostas neste trabalho avaliam somente a violação do período de planejamento (Duração). As demais violações como (i) excesso de peso dos veículos (Capacidade), (ii) cada rota começa e termina no depósito (Ciclo) e (iii) cada cliente é visitado exatamente por uma rota (Exclusividade) apessarem de ocorrer, não são mensuradas a sua inviabilidade. Soluções que violem alguma das restrições do PRV como capacidade, ciclo e exclusividade são descartadas.

2.9.1 Overtime

O *Overtime* ocorre toda vez que a duração das rotas atribuidas a um veículo ultrapassa o horizonte de planejamento dado. O valor do *Overtime* Ot_k de um veículo k é dado pela seguinte expressão: $Ot_k = max(0, t_{r_k} - T)$, onde t_{r_k} é a soma da duração das rotas atribuídas ao veículo k. Caso o veículo não exceda T, o valor de horas extras é zero.

O *Overtime* total é definido por:

$$\mathrm{Ot} = \sum_{k=1}^{m} Ot_k. \tag{2.5}$$

A Figura 2.3 mostra uma solução do PRVMV em que o veículo 1 e 3 violam a restrição de horizonte de planejamento ($Ot_1 = 1$ e $Ot_3 = 1$) e o veículo 2 satisfaz ($Ot_2 = 0$). Segundo essa ilustração o *Overtime* total é ígual a dois (Ot = 2).

2.9.2 Longest Tour Ratio

Por estratégica logística, nem sempre a concentração de horas extras por um único veículo é interessante, pois pode acarretar em descartes prematuros e jornada de trabalho muito extensa. Por este motivo foi criada a métrica *Longest Tour Ratio* (LTR) que tem como objetivo avaliar a distribuição das viagens pelos veículos disponíveis. Para isso, definimos

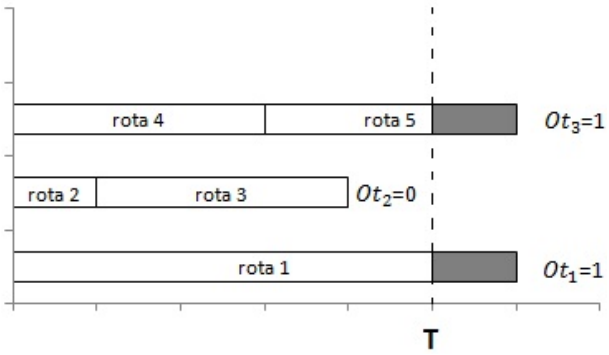
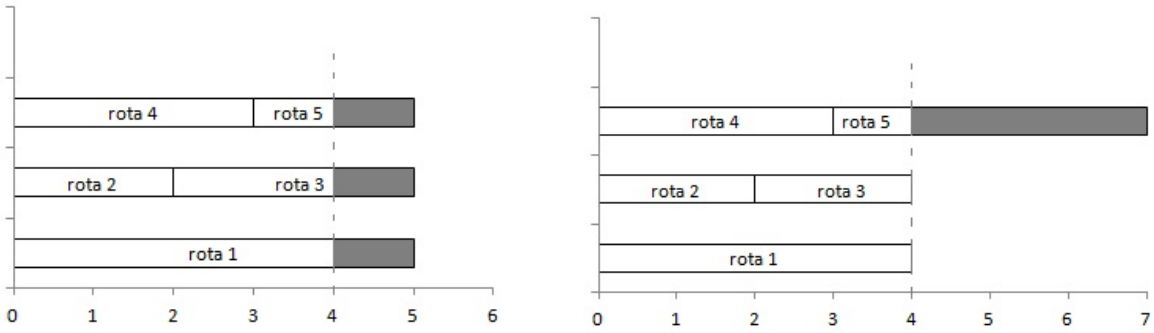


Figura 2.3: Exemplo da métrica Overtime

LTR como a razão entre a maior duração de rota t_{r_k} para todo $k \in K$, sobre o período de planejamento T.

$$\text{LTR} = \frac{max_K(Ot_k)}{T}. \tag{2.6}$$

Quanto mais elevado o valor de LTR maior será a concentração de horas extras por único veículo. Para obter soluções de melhor qualidade (distribuição das horas suplementares) almeja-se menores valores de LTR. No gráfico da Figura 2.4 há um exemplo da utilização da métrica LTR em que os valores de *Overtime* das duas soluções é o mesmo, no entanto o valor do LTR varia de acordo com a concentração de horas extras por veículo. Na Figura 2.4 (a), onde o *Overtime* está distribuído entre os veículos, o LTR é menor e equivale a 5/4. Na Figura 2.4 (b), onde há um acúmulo das horas suplementares, o valor do LTR é maior e equipara-se a 7/4.



(a) Distribuição de Overtime (b) Acúmulo de Overtime

Figura 2.4: Exemplo da métrica LTR

De acordo com esta medida de inviabilidade, também podemos observar se a solução é viável ou não viável. A solução é não viável quando ocorre a violação do período de planejamento (LTR $>$ 1). Quando não há a violação da restrição de duração, a solução é

viável (LTR ≤ 1).

2.9.3 Penalized Cost

A métrica *Penalized Cost* (PC) é simplesmente a adição de uma penalidade ao custo da solução. Esta penalidade é o custo adicional empregado pela violação do período de planejamento, isto é, o *Overtime* total multiplicado por um parâmetro γ.

$$P_\gamma = (\gamma)(Ot). \tag{2.7}$$

Esta medida de inviabilidade destacada nessa subseção, propõe a junção do custo da solução $f(s)$ mais a penalidade P_γ.

$$PC_\gamma = f(s) + P_\gamma. \tag{2.8}$$

O parâmetro γ pode variar de acordo com o custo relacionado com as horas extras. Na maioria dos trabalhos acadêmicos pesquisados assume-se que γ é igual a dois.

Capítulo 3

Metodologia proposta

Este capítulo apresenta os componentes do algoritmo proposto, os procedimentos para construir uma solução inicial, as estruturas de busca local empregadas e os mecanismos de perturbação. Por fim, mostra-se como é resolvido o problema de atribuição (*Bin Packing*).

3.1 Algoritmo Proposto

O algoritmo proposto reúne componentes da metaheurística ILS [29], implementada usando uma abordagem *multi-start* para a geração de soluções iniciais, e do procedimento *Randomized Variable Neighborhood Descent* (RVND), utilizado na fase de busca local. O ILS-RVND-BFD irá gerar um conjunto de rotas que irão compor uma solução para o PRV (i.e. soluções que satisfaçam as restrições de exclusividade, capacidade e ciclo). A seguir, as rotas geradas pela solução do PRV são atribuídas a um conjunto fixo de veículos, no qual cada rota tem uma duração e o tempo de trabalho dos veículos é limitado (i.e. o atendimento da restrição de duração). O algoritmo pode ser visto de forma simplificada no Algoritmo 2, cujos parâmetros são utilizados para determinar o número máximo de iterações ($MaxIter$) e o número máximo de perturbações consecutivas sem melhoras ($MaxIterILS$).

A cada iteração (linhas 6-38) uma nova solução (conjunto de rotas) é gerada (linha 7) e as rotas são agrupadas por veículos usando o BFD (linha 9), caso a penalidade for igual a zero ($P_\gamma = 0$) a solução é viável. No laço principal do ILS (linhas 11-27) há uma tentativa de aprimoramento da solução s efetuando uma varredura no espaço de solução usando o RVND (linha 12). Caso o critério de aceitação da linha 14 ou 19 seja satisfeito é atualizada a nova melhor solução corrente e reiniciado o contador $iterILS$.

Algoritmo 2 ILS-RVND-BFD

```
1: ILS-RVND-BFD(MaxIter, MaxIterILS)
2: CarregaInstancia();
3: rota ← EstimaNumeroRotas();
4: f* ← ∞;
5: Pγs* ← ∞;
6: para i := 1, ..., MaxIter faça
7:    s ← GeraSolucaoInicial(rota);
8:    s' ← s;
9:    Pγs' ← γ × BFD(s, K);
10:   iterILS ← 0;
11:   enquanto iterILS ≤ MaxIterILS faça
12:      s ← RVND(s);
13:      Pγs ← γ × BFD(s, K);
14:      se Pγs = 0 and Pγs' = 0 and f(s) < f(s') então
15:         s' ← s;
16:         Pγs' ← Pγs;
17:         iterIls ← 0;
18:      senão
19:         se Pγs < Pγs' então
20:            s' ← s;
21:            Pγs' ← Pγs;
22:            iterIls ← 0;
23:         fim se
24:      fim se
25:      s ← Pertubacao(s');
26:      iterILS ← iterILS + 1;
27:   fim enquanto
28:   se Pγs' = 0 and f(s') < f* então
29:      s* ← s';
30:      f* ← f(s*);
31:      Pγs* ← 0;
32:   senão
33:      se Pγs' < Pγs* então
34:         s* ← s';
35:         Pγs* ← Pγs';
36:      fim se
37:   fim se
38: fim para
39: return s*;
40: end ILS-RVND-BFD
```

Na execução do ILS-RVND-BFD foram adicionadas duas condições extras para aumentar a eficiência do algoritmo. A primeira ocorre quando a penalidade $P_\gamma = 0$ e $i > 25\%$ de $MaxIter$ ocasionando a interrupção do algoritmo. Essa restrição tem como objetivo minimizar o tempo computacional para instâncias fáceis de encontrar soluções viáveis a partir das primeiras iterações. A segunda condição é aplicada quando $i = 95\%$ de $MaxIter$, fazendo com que o número de rotas seja duplicado, com o intuito de diversificação da solução inicial. Nas seções a seguir, serão explicados os principais componentes do Algoritmo 2.

3.2 Procedimento EstimaNumeroRotas

Este procedimento, cujo pseudocódigo é apresentado no Algoritmo 3, é aplicado para determinar o número de rotas que deve ser gerado a cada solução inicial. Inicialmente assume-se que a variável $rota$ é igual a 1. Na linha 3 é criada uma lista de candidatos (LC) com todos os clientes. Enquanto a lista não estiver vazia (linhas 6-17) é feito o cálculo do custo $g(i)$ $\forall i \in LC$ que representa o custo de inserir o cliente i na rota mais próxima (i.e. de menor custo), de acordo com um critério de Inserção Mais Próxima (IMP). O menor custo de inserção (g^{min}) associado ao cliente i^* deve ser adicionado na rota s^{rota}. Caso a rota ultrapasse a capacidade de carga (Q) do veículo, uma nova rota é gerada.

Algoritmo 3 EstimaNumeroRotas

```
1: EstimaNumeroRotas()
2: rota ← 1;
3: LC = V \ {0}
4: s_rota ← i ∈ LC selecionado aleatoriamente
5: Atualiza LC
6: enquanto LC ≠ {} faça
7:    g(i) é o custo associado a i ∈ LC usando o critério IMP.
8:    g^min ← min{ g(i)|i ∈ LC }
9:    i* ← cliente associado g^min
10:   s_rota ← s_rota ∪ {i*}
11:   se demanda da s_rota > Q então
12:      rota ← rota + 1
13:      s_rota ← k ϵ LC selecionado aleatoriamente
14:   senão
15:      Atualiza LC
16:   fim se
17: fim enquanto
18: return rota
19: end EstimaNumeroRotas
```

3.3 Construção da Solução Inicial

Para obter uma solução inicial do PRVMV, primeiramente é preciso encontrar uma solução que satisfaça o PRV. O Algoritmo 4 cria esta solução sendo que, em princípio, em cada rota é colocado um cliente da lista LC escolhido aleatoriamente (linhas 4 - 7). No laço principal (linhas 8 - 14), foi desenvolvida uma estratégia de inserção paralela que considera todas as rotas ao avaliar a inserção de menor custo. Tais inserções são avaliadas usando o critério de inserção mais próxima. Caso a solução gerada alcance uma inviabilidade, i.e. existe um cliente $i \in LC$ onde a sua inserção em qualquer rota R de s induzirá $q_R > Q$, uma nova rota é adicionada e o algoritmo retorna para linha 2.

Algoritmo 4 Gera Solução Inicial

```
1: GeraSolucaoInicial(rota)
2: LC = V \ {0}
3: s = {R_1, ..., R_rota}
4: para r := 1, ..., rota faça
5:    R_r ← i ∈ LC selecionado aleatoriamente
6:    Atualiza LC
7: fim para
8: enquanto LC ≠ {} e existir pelo menos um cliente i ∈ LC que pode ser adicionando
   a s faça
9:    g(i) é o custo associado i ∈ LC usando o critério IMP.
10:   g^min ← min{ g(i)|i ∈ LC }
11:   i* ← cliente associado a g^min e R_min ← rota associada a g^min
12:   R_min ← R_min ∪ {i*}
13:   Atualiza LC
14: fim enquanto
15: se s = Inviavel então
16:   rota = rota + 1
17:   retorne a linha 2
18: fim se
19: return s
20: end GeraSolucaoInicial
```

3.4 Random Variable Neighborhood Descent

O Algoritmo 5 apresenta o pseudocódigo do procedimento RVND, que consiste em uma adaptação do *Variable Neighborhood Descent*, proposto por Mladenovic e Hansen[30]. Inicialmente a lista LV é inicializada com as estruturas de vizinhança Inter-Rotas (descritas na Seção 3.5). Em cada execução do laço principal (3-12) uma vizinhança N é escolhida aleatoriamente (linha 4) e então, a cada movimento dessa estrutura de vizinhança que

gere um menor custo e avaliado se houve um acrésimo na penalidade (calculado através do algoritmo BFD descrito na Seção 3.8), caso ocorra é descartado esse movimento. Armazenado em $s^{'}$ apenas a *Best Improving* que não aumente o *Overtime* (linha 5). Em caso de melhora, a solução $s^{'}$ passa pelo procedimento *IntraRota* (descrito na Seção 3.6) e a lista LV é reinicializada com todas as vizinhanças Inter-Rotas (linha 8). Caso contrário, a estrutura de vizinhança N é removida da lista LV.

Algoritmo 5 RVND

```
1: RVND(s)
2: Inicializa uma Lista (LV) com as vizinhanças Inter-Rotas
3: enquanto LV ≠ {} faça
4:    Escolhe uma vizinhança N ∈ LV aleatoriamente
5:    s' ← Encontre o melhor movimento da vizinhança N em s que não ocasione aumento
      de penalidade
6:    se f(s') < f(s) então
7:       s ← IntraRota(s')
8:       Reinicializa LV
9:    senão
10:      Remove N da lista LV
11:   fim se
12: fim enquanto
13: return s
14: end RVND
```

3.5 Vizinhanças Inter-Rotas

Nas vizinhanças inter-rotas ocorrem movimentos de clientes entre duas rotas distintas. As buscas locais descriminadas abaixo têm como objetivo principal o aprimoramento da solução corrente encontrada. Vale ressaltar que ao encontrar um movimento que diminua o custo de deslocamento, este é submetido a uma avaliação, caso ocorra aumento de penalidade (P_γ) esse movimento é descartado. As vizinhanças Inter-Rotas são descritas a seguir:

3.5.1 Shift(1,0)

A estrutura de vizinhança Shift(1,0) consiste em transferir um cliente de uma rota para outra, conforme mostrado na Figura 3.1, onde o cliente 4 é realocado da Rota 1 para Rota 2.

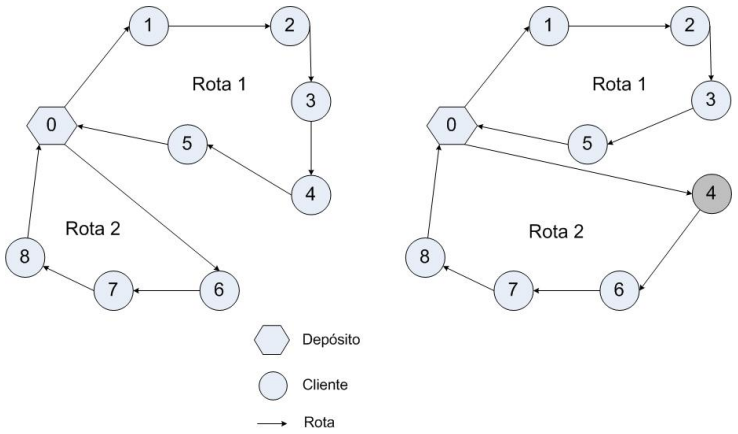


Figura 3.1: Shift(1,0)

3.5.2 Shift(2,0)

Shift (2,0) é análogo ao Shift(1,0), no entanto dois clientes consecutivos são transferidos. A Figura 3.2 ilustra um exemplo em que os clientes 4 e 5 são transferidos da Rota 1 para Rota 2.

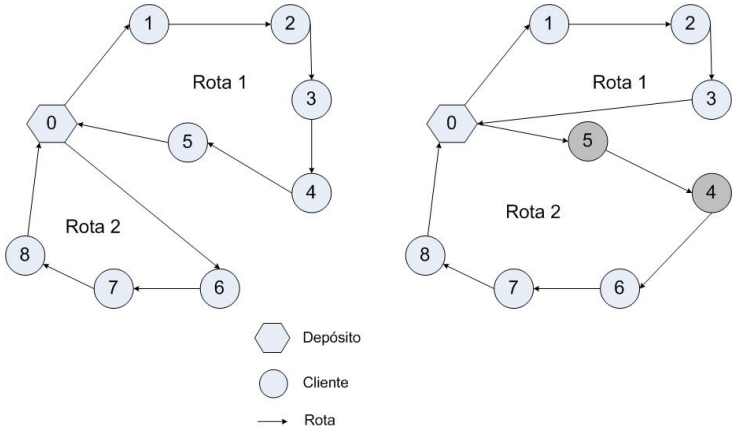


Figura 3.2: Shift(2,0)

3.5.3 Swap(1,1)

A estrutura Swap(1,1) é a permutação entre dois clientes pertencentes a rotas diferentes, conforme exemplificado na Figura 3.3 onde ocorre a troca do cliente 5 pertencente à Rota 1 com o cliente 6 pertencente à Rota 2.

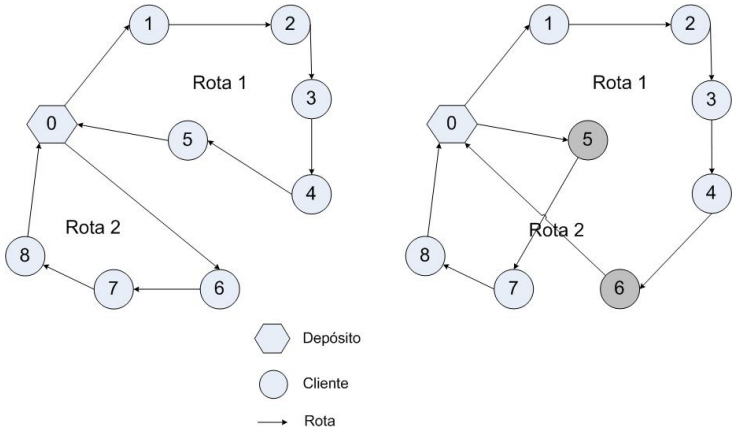


Figura 3.3: Swap(1,1)

3.5.4 Swap(2,2)

Swap(2,2) é similar Swap(1,1), porém dois clientes consecutivos são permutados. A ilustração dessa estrutura de vizinhança pode ser verificada na Figura 3.4, onde os clientes 4 e 5 pertencentes à Rota 1 são trocados pelos clientes 6 e 7 do conjunto de clientes da Rota 2.

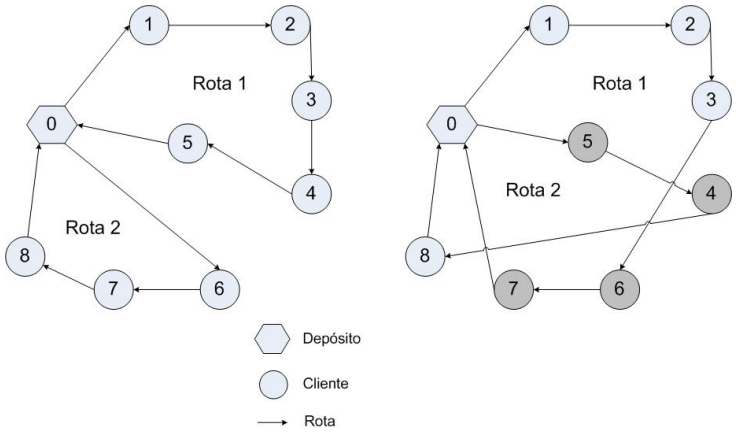


Figura 3.4: Swap(2,2)

3.5.5 Swap(2,1)

Swap(2,1) é a permutação de dois clientes adjacentes com um outro cliente que pertence a uma rota distinta, conforme ilustrado na Figura 3.5, onde os clientes adjacentes 6 e 7 são trocados pelo cliente 5.

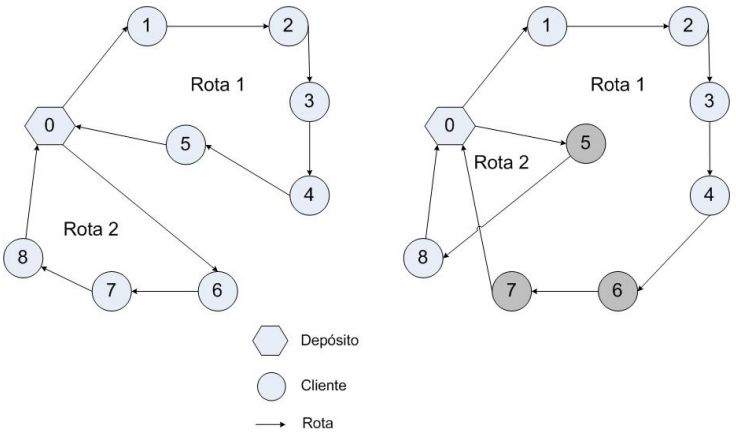


Figura 3.5: Swap(2,1)

3.5.6 Cross

Cross consiste na remoção de dois arcos, sendo um de cada rota, e na inserção de outros dois de modo a gerar uma nova solução. A Figura 3.6 mostra a remoção dos arcos (4,5) e (0,6) e a inserção de dois novos arcos (4,0) e (5,6) formando uma nova solução.

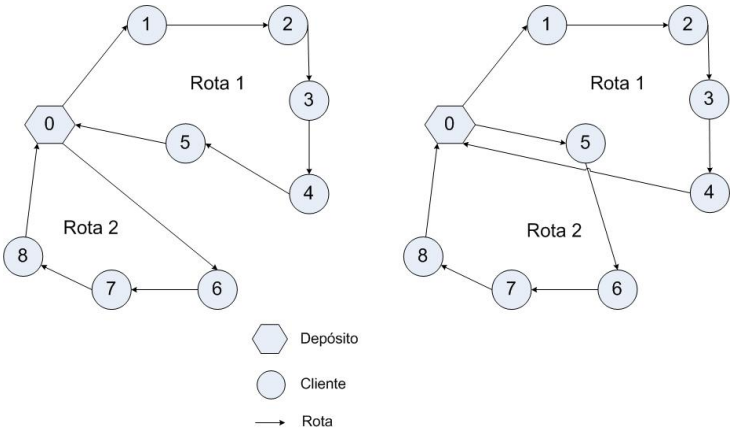


Figura 3.6: Cross

3.6 Vizinhanças Intra-Rotas

As estruturas de vizinhanças Intra-Rotas são definidas como movimentos entre clientes que acontecem em uma mesma rota. Neste trabalho são utilizadas as vizinhanças *Reinserção*, *Or-opt2*, *Or-opt3*, *2-opt* e *Exchange*. Nas descrições a seguir, os movimentos relativos às vizinhanças Intra-Rotas têm como configuração inicial a rota ilustrada na Figura 3.7.a.

- **Reinserção** - Esta vizinhança é definida como o movimento de um cliente de uma posição para outra posição da rota. A Figura 3.7.b ilustra a mudança do cliente 3

para a posição do arco (4,5).

- **Or-opt2** - A vizinhança *Or-opt2* consiste na realocação de dois clientes adjacentes de uma posição para outra posição da rota. Conforme mostrado na Figura 3.7.c, os cliente 2 e 3 são movimentados para a posição do arco (5,6).
- **Or-opt3** - De forma similar, na vizinhança *Or-opt3* três clientes adjacentes (1, 2 e 3) são removidos e inseridos em uma outra posição, conforme mostrado na Figura 3.7.d.
- **2-opt** - Esta vizinhança é definida pela remoção de dois arcos não adjacentes enquanto outros dois arcos são adicionados de tal forma que uma nova rota é gerada. Tal situação é verificada na Figura 3.7.e, onde o arco(2,3) e o arco(4,5) são removidos formando uma nova configuração.
- **Exchange** - A vizinhança *Exchange* consiste na permutação de dois clientes. A Figura 3.7.f ilustra a troca de posição do cliente 2 com o 5.

O pseudocódigo do procedimento *IntraRota* está esquematizado no Algoritmo 6. Este procedimento difere do Algoritmo 5 por não reinicializar a lista de vizinhanças (LV).

Algoritmo 6 IntraRota

```
1: IntraRota(s)
2: Inicializa uma Lista (LV) com as vizinhanças Intra-Rotas
3: enquanto LV ≠ {} faça
4:    Escolhe uma vizinhança N ∈ LV aleatoriamente
5:    s' ← Encontre o melhor movimento da vizinhança N em s
6:    se f(s') < f(s) então
7:       s ← s'
8:    senão
9:       Remove N da lista LV
10:   fim se
11: fim enquanto
12: return s
13: end IntraRota
```

3.7 Mecanismos de Perturbação

Mecanismos de perturbação permitem a movimentação de clientes, sem que haja verificação de redução de custo, com a finalidade de diversificar uma solução. Neste trabalho foram utilizadas três estruturas de perturbação escolhidas aleatoriamente em cada iteração do laço principal do ILS. São as seguintes:

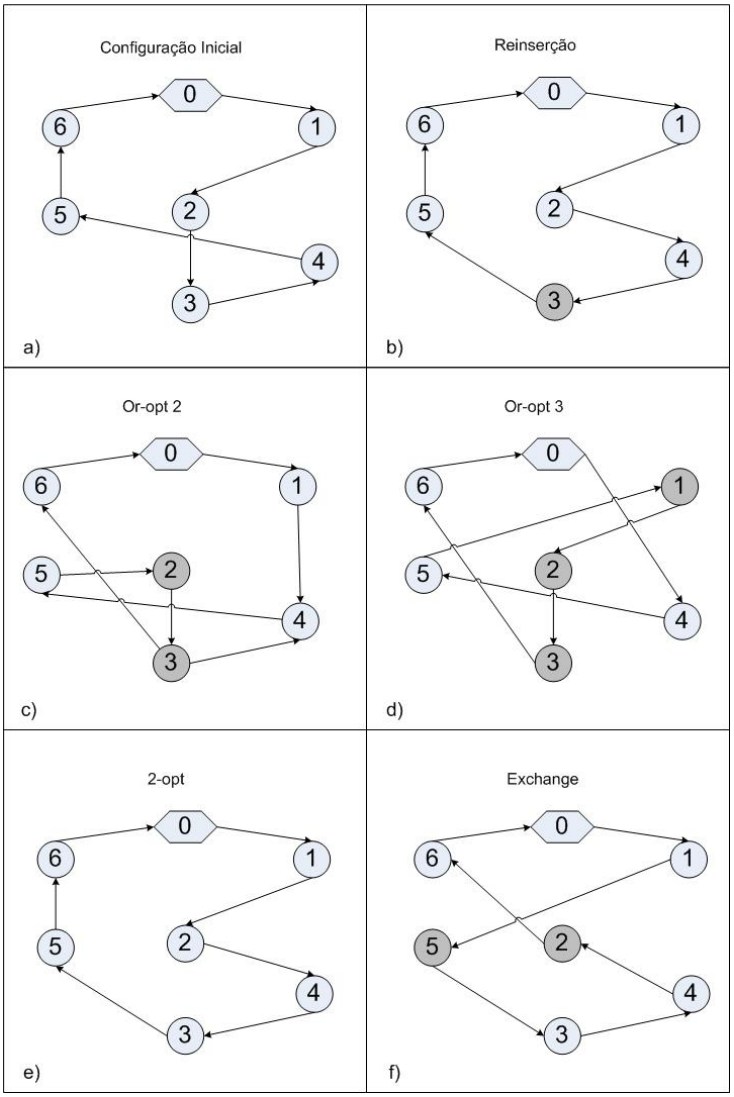


Figura 3.7: Movimentos Intra-Rotas

- **Random-Shift(1,1)** - consiste em transferir um cliente de uma rota r_1 para uma outra rota r_2 e transferir um cliente de r_2 para r_1, ambas as transferências efetuadas aleatoriamente.

- **Random-Swap(1,1)** - consiste na aplicação do movimento Swap(1,1) de forma aleatória.

- **Random-Swap(2,2)** - consiste na aplicação do movimento Swap(2,2) de forma aleatória.

3.8 Bin Packing Problem

O *Bin Packing Problem* (BPP) é um clássico problema de otimização combinatória que pode ser modelado para solucionar diversos problemas do mundo real, como por exemplo, alocação de arquivos, cortes de materiais e carregamento de caminhões sujeito a limitações de peso.

Segundo Garey e Johnson[18] o BPP é um problema NP-Completo em sentido forte, sendo assim é improvável que exista um algoritmo exato que resolva toda instância em tempo polinomial. Por esse motivo há maior aplicabilidade de algoritmos heurísticos clássicos como *First Fit* (FF), *Best Fit* (BF), *First Fit Decreasing* (FFD) e *Best Fit Decreasing* (BFD).

Os algoritmos citados são divididos em duas classes: *on-line* e *off-line*. No *on-line* os itens são processados sem um conhecimento prévio dos itens que vêm a seguir, é o caso do FF e BF. No *off-line* isso não ocorre. Primeiramente é realizado um pré-processamento dos itens, ordenando estes em ordem decrescente e posteriormente havendo o empacotamento. O FFD e BFD são exemplos desses tipos de algoritmos *off-line* [27].

O BPP aplicado ao PRVMV, consiste em atribuir um conjunto de rotas a um conjunto fixo de veículos, no qual cada rota tem uma duração e o tempo de trabalho dos veículos é limitado. Consequentemente, a soma das viagens atribuídas a um veículo não pode ultrapassar a sua jornada de trabalho. Caso ocorra esta violação no horizonte de tempo do veículo é calculado o *Overtime*.

Para solucionar o BPP foi aplicada uma heurística gulosa denominada *Best Fit Decreasing* (BFD), onde as viagens (rotas) são ordenadas em ordem não-crescente de duração. A BFD aloca a viagem ao veículo $k \in K$ de maior t_k, mas que tenha tempo livre o suficiente para armazená-la. Caso nenhum dos veículos já utilizados tiver capacidade de tempo suficiente para acomodar a viagem, um novo veículo é usado. No entanto, caso o número limite de veículos seja atingido, uma nova viagem é atribuída ao veículo que produza menor penalidade. A complexidade do algoritmo é $O(n\ log\ n)$.

Capítulo 4

Experimentos Computacionais

O algoritmo proposto ILS-RVND-BFD foi desenvolvido na linguagem de programação C++ e os testes foram efetuados em uma máquina com microprocessador AMD Turion X2 Ultra Dual-Core Mobile ZM-80 2.10 GHz com 4 GB de memória RAM utilizando o sistema operacional Ubuntu Linux 10.04 Kernel 2.6.32-33.

4.1 Descrição das Instâncias

Os experimentos computacionais foram realizados em um conjunto de nove problemas testes bem conhecidos da literatura e amplamente explorados com diversas configurações distintas. As instâncias apresentadas na Tabela 4.1 são construídas aproveitando os gráficos de distribuição geográfica, de demandas e capacidades dos problemas CMT-1, CMT-2, CMT-3, CMT-4, CMT-5, CMT-11 e CMT-12 propostas por Christofides et al. [6] e os problemas F-11 e F-12 propostos por Fischer [15].

Tabela 4.1: Características das Instâncias

Problema	n	T	m	z^{VRP}
CMT-1	50	T_1, T_2	1,...,4	524.61
CMT-2	75	T_1, T_2	1,...,7	835.26
CMT-3	100	T_1, T_2	1,...,6	826.14
CMT-4	150	T_1, T_2	1,...,8	1028.42
CMT-5	199	T_1, T_2	1,...,10	1291.44
CMT-11	120	T_1, T_2	1,...,5	1042.11
CMT-12	100	T_1, T_2	1,...,6	819.56
F-11	71	T_1, T_2	1,...,3	241.97
F-12	134	T_1, T_2	1,...,3	1162.96

Devido as combinações distintas dos elementos pertencentes à Tabela 4.1 obtém-se

um grupo de 104 problemas testes. Na execução de cada um desses problemas foram considerados dois horizontes de tempos diferentes $T_1 = \{(1.05 * z^{VRP})/m\}$ e $T_2 = \{(1.10 * z^{VRP})/m\}$, onde z^* é o valor encontrado pela solução VRP obtido pelo algoritmo de Rochat e Taillard [39], m é o número de veículos, T é o horizonte de tempo e n é o número de clientes.

4.2 Resultados Detalhados

Para obter uma solução, o algoritmo ILS-RVND-BFD necessita de dois parâmetros, *maxIter* e *maxIterILS*, descritos na Seção 3.1. Estes foram determinados empiricamente e os valores adotados foram: $maxIter = 40$ e $maxIterILS = n + 10 \times rotas$, onde n é o número de clientes.

Para definir a qualidade das soluções viáveis, foi utilizada a métrica GAP que pode ser representada pela razão entre a solução do ILS-RVND-BFD sobre o melhor custo do VRP.

$$GAP_s = 100 * \left(\frac{f(s)}{z^{VRP}} - 1 \right). \tag{4.1}$$

Os resultados detalhados para cada instância com relação a esta métrica, são apresentados nas linhas das Tabelas A.1, A.2 e A.3 do apêndice A, onde cada coluna mostra as seguintes informações:

- Problema: o problema base do PRV;
- m e T: o número de veículos e o horizonte de tempo;
- GAP: é uma métrica que considera apenas os problemas onde foram encontradas soluções viáveis. Com 10 execuções realizadas em cada instância obtêm-se os valores Melhor, Média e Pior;
- AMP: é um algoritmo desenvolvido por Oliveira e Vieira [31] onde são apresentados os seus GAPs;
- TLG, BM, PS, GA: os símbolos indicam se foi encontrado uma solução viável (•) ou não (∘), referente aos trabalhos realizados por Taillard [45], Brandão e Mercer[4], Petch e Salhi [35] e Petch e Salhi [40], respectivamente.

O ILS-RVND-BFD encontrou 99 soluções viáveis dos 104 problemas testes, incluindo uma nova solução ilustrada no apêndice B na Tabela B.1 que não havia sido solucionada por nenhuma outra proposta anterior [45], [4], [35], [31] e [40]. Os algoritmos de Taillard [45], Brandão e Mercer[4], Petch e Salhi [35], Oliveira e Vieira [31] e Petch e Salhi [40] encontraram 86, 89, 76, 98 e 62 soluções viáveis, respectivamente.

Na Tabela 4.2 é apresentado um sumário com relação às informações pertencentes ao apêndice A e um comparativo com o algoritmo de memória adaptativa proposto por Oliveira e Vieira [31], que apresenta os melhores resultados para soluções viáveis na literatura. Cada linha referencia as instâncias do PRV que contém o GAP de dois horizontes de tempo distintos T_1 e T_2, onde são reportadas as seguintes características:

- Viável/Total: o número de soluções encontradas viáveis e o número total de instâncias avaliadas;
- GAP: é a média do GAP considerando apenas os problemas onde foram encontradas soluções viáveis;
- a última linha apresenta o total de soluções viáveis encontradas (Viável/Total) e a média do GAP.

Tabela 4.2: Resultado médio considerando o melhor GAP

		ILS-RVND-BFD				Oliveira e Vieira [31]			
		T_1		T_2		T_1		T_2	
Problema	n	Viável/Total	GAP	Viável/Total	GAP	Viável/Total	GAP	Viável/Total	GAP
CMT-1	50	2/4	**0.80**	4/4	**2.62**	2/4	**0.80**	4/4	**2.62**
CMT-2	75	6/7	**0.47**	7/7	**0.52**	6/7	0.72	7/7	0.95
CMT-3	100	6/6	**0.41**	6/6	**0.35**	6/6	0.87	6/6	0.47
CMT-4	150	8/8	**1.04**	8/8	**0.62**	7/8	1.16	8/8	1.33
CMT-5	199	10/10	**1.56**	10/10	**1.33**	10/10	2.74	10/10	2.15
CMT-11	120	5/5	**0.70**	5/5	**0.11**	5/5	1.52	5/5	0.34
CMT-12	100	5/6	0.65	6/6	**0.18**	5/6	**0.63**	6/6	**0.18**
F-11	71	2/3	**1.84**	3/3	**1.67**	2/3	2.10	3/3	1.8
F-12	134	3/3	**0.00**	3/3	**0.01**	3/3	0.48	3/3	0.64
Média Total		47/52	**0.83**	52/52	**0.82**	46/52	1.37	52/52	1.19

Na Média Total, a metaheurística empregada na resolução do PRVMV apresenta os menores GAPs na ordem de 39% com relação ao período de planejamento T_1 e 31% em comparação a T_2.

4.3 Resultados Detalhados para Instâncias Inviáveis

Para mensurar a qualidade das soluções inviáveis são utilizadas certas métricas como LTR, Ot e PC. As únicas instâncias em que o algoritmo ILS-RNVD não conseguiu reportar solução viável foram:

1. CMT-1 | 3 | T_1;
2. CMT-1 | 4 | T_1;
3. CMT-2 | 7 | T_1;
4. CMT-12| 6 | T_1;
5. F-11 | 3 | T_1;

Os melhores resultados encontrados pelo algoritmo durante cinco execuções para as soluções não viáveis são apresentados na Tabela 4.3, onde há um comparativo com os algoritmos de melhores resultados na literatura: Oliveira e Vieira [31], Brandão e Mercer [4] e Taillard [45]. Para cada instância são avaliados os Custos $\boldsymbol{f(s)}$, *Overtime* $\boldsymbol{Ot}$ e *Penalized Cost* $\boldsymbol{PC_\gamma}$. Além disso, a última linha da tabela apresenta as diferenças (%) entre as médias de cada algoritmo em relação ao ILS-RVND-BFD.

É possível observar que a diferença média do custo $\boldsymbol{f(s)}$ de roteamento do algoritmo ILS-RVND-BFD apresenta um acréscimo maior do que os demais, entretanto o maior acréscimo é de apenas 3,53% com relação ao algoritmo de Taillard [45]. Porém, o objetivo principal do algoritmo ILS-RVND-BFD para instâncias inviáveis é a redução do *Overtime* (horas extras). Esta métrica, em relação aos algoritmos de Oliveira e Vieira [31], Brandão e Mercer [4] e Taillard [45], obtém melhores resultados, apresentando uma redução na média da diferença de 11.71%, 39.53% e 67.14% respectivamente. Usando $\lambda = 2$ o custo penalizado do ILS-RVND-BFD é apenas maior 0,14% do que o algoritmo de Busca Tabu de [45], porém, vale ressaltar que quanto maior o valor cobrado pelas horas extras, maior será o valor do PC, consequentemente melhorando os resultados do ILS-RVND-BFD.

A Tabela 4.4 mostra um comparativo da razão da maior rota sobre o período de planejamento (LTR) para soluções inviáveis. O ILS-RVND-BFD iguala-se aos melhores da literatura em três, das cinco instâncias em que não foram encontradas soluções viáveis.

Tabela 4.3: Comparação de total overtime (Ot) e penalized cost (PC) para soluções inviáveis

	ILS-RVND-BFD			Oliveira e Vieira [31]			Brandão e Mercer [4]			Taillard [45]		
Prob.	$f(s)$	Ot	PC_λ	$f(s)$	Ot	PC_λ	$f(s)$	Ot	PC_λ	$f(s)$	Ot	PC_λ
1	556.22	**6.54**	**569.30**	558.82	7.29	573.40	556.34	9.70	575.73	**533.00**	23.24	579.48
2	555.08	**6.63**	568.34	547.10	8.49	**564.07**	547.10	8.49	**564.07**	**546.29**	9.49	565.27
3	868.15	**1.38**	**870.91**	873.40	1.86	877.11	870.07	13.25	896.57	**843.60**	17.35	878.29
4	852.99	**3.81**	860.61	852.99	**3.81**	860.61	**819.56**	14.14	847.85	**819.56**	12.96	**845.48**
5	256.07	1.84	259.75	256.85	1.85	260.54	254.40	**1.54**	257.47	**244.60**	6.36	**257.31**
Média da Diferença%				0.02	-11.71	-0.19	1.19	-39.53	-0.17	3.53	-67.14	0.14

Tabela 4.4: Comparação de longest tour ratio (LTR) para soluções inviáveis

Prob.	ILS-RVND-BFD	Oliveira e Vieira [31]	Brandão e Mercer [4]	Taillard [45]	Petch e Salhi [35]	Petch e Salhi [40]
1	1.032	**1.024**	1.041	1.115	1.026	1.030
2	**1.027**	**1.027**	**1.027**	**1.027**	1.085	1.056
3	**1.009**	**1.009**	1.088	1.073	1.064	1.102
4	**1.014**	**1.014**	1.072	1.064	1.029	1.029
5	1.020	1.020	**1.011**	1.075	1.020	1.020
Média	1.020	1.019	1.048	1.071	1.045	1.047
Média da Diferença%		0,16	-2,53	-4,64	-2,28	-2,49

4.3.1 Eficiência do Método

A eficiência do método pode ser verificada pela Distribuição de Probabilidade Empírica (DPE), que consiste em determinar um alvo, realizar um número fixo de execuções (nf) e armazenar o tempo em que se atinge este alvo. Esses tempos de processamentos são ordenados e, para cada tempo de processamento t_i obtêm-se a probabilidade $p_i = (i - 1/2)/nf$. Após este cálculo os pontos $z_i = (t_i, p_i)$ são gerados no gráfico, onde i vai de 1 até o número de execuções ($i = 1, ..., nf$).

O algoritmo ILS-RVND-BFD desenvolvido foi comparado com dois outros algoritmos que utilizam componentes distintos na fase de Busca Local e no critério de aceitação. O primeiro algoritmo é o *Iterated Local Search* (ILS), que não faz o uso do procedimento RVND na fase de Busca Local, no entanto continua a utilizar os mesmos movimentos do RVND uma unica vez e em uma ordem pré-definida. O segundo algoritmo é o ILS-RVND que altera o critério de aceitação dos movimentos empregados pelas estruturas de vizinhança na fase de busca local, eliminando a restrição que não permitia movimentos com aumento de penalidade.

Foram consideradas duas instâncias com horizonte de planejamento igual a T_1, que são:

1. a instância CMT-1 proposta por Christofides et al. [6] tem 50 vértices. O número de veículos é igual a quatro ($m = 4$) e não tem solução viável;

2. a instância CMT-5 proposta por Christofides et al. [6] tem 199 vértices. O número de veículos é igual a nove ($m = 9$) e tem solução viável;

Para a instância inviável CMT-1 analisada, dois alvos são determinados. Um alvo fácil a 5% do melhor *Overtime* encontrado na literatura, e um alvo difícil a 0%. A instância viável CMT-5 considerada, tem dois alvos a partir da melhor solução (custo) da literatura com um alvo fácil a 5% e um alvo difícil a 1%. Na Tabela 4.5 são apresentadas estas instâncias contendo os melhores custos conhecidos da literatura para o PRVMV, os melhores Ot e os alvos a serem atingidos.

Tabela 4.5: Valores dos alvos a serem atingidos

Instâncias	Melhor Custo	Melhor *Overtime* (Ot)	Alvo Fácil	Alvo Difícil
CMT-1	-	8.49	8.91	8.49
CMT-5	1335.35	0.0	1402.11	1348.70

Para realizar a Distribuição de Probabilidade Empírica, As instâncias CMT-1 e CMT-5 foram executada 100 e 30 vezes respectivamente. O gráfico de DPE ilustrado na Figura 4.1 para o alvo fácil com relação ao problema CMT-1, mostra que o algoritmo ILS-RVND-BFD tem uma probabilidade de quase 100% de alcançar o alvo em menos de 5 segundos, já o algorimto ILS-RVND e ILS, 6 e 45 segundos respectivamente. Para alcançar o alvo difícil o ILS-RVND-BFD e ILS-RVND gastam menos que 7 e 8 segundos respectivamente, no entanto o ILS gasta mais que 93 segundos. Tais informações estão contidas no gráfico apresentado na Figura 4.2.

Os gráficos apresentados nas Figuras 4.3 e 4.4 mostram o comportamento do ILS, ILS-RVND e ILS-RVND-BFD para a instância CMT-5, com um limite de tempo de 10 minutos. Observa-se que para o alvo fácil, tanto o ILS-RVND quanto o ILS-RVND-BFD chegam com 100% de probabilidade no alvo em menos de 200 segundos, enquanto neste mesmo tempo, as chances do ILS encontrar uma solução com este custo é de apenas 10%. No tempo limite estabelecido, o ILS só consegue encontrar o alvo em 40% das execuções.

Para o alvo difícil, nenhum dos algoritmos consegue achar o alvo com 100% de probabilidade no tempo determinado. O algoritmo que possui o melhor desempenho é o ILS-RVND-BFD com 62% de chance de localizar o alvo difícil. O ILS não consegue obter nenhuma solução para o alvo dentro do tempo estipulado, obtendo o pior desempenho.

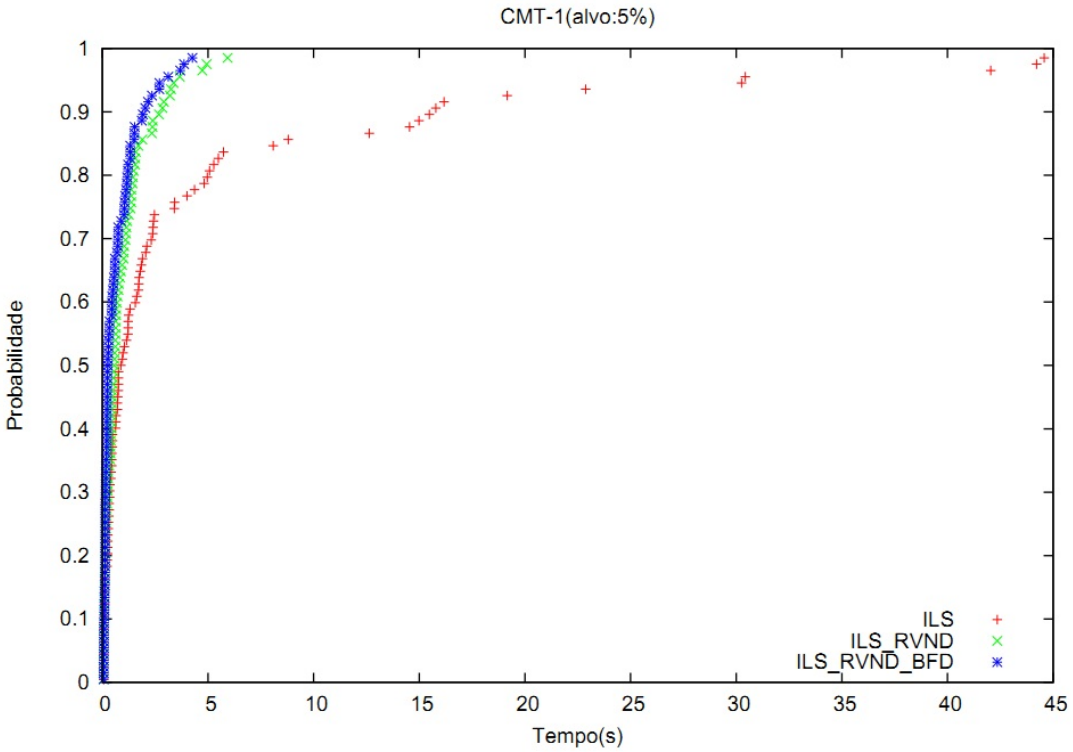


Figura 4.1: DPE para o problema teste CMT-1 alvo 5%

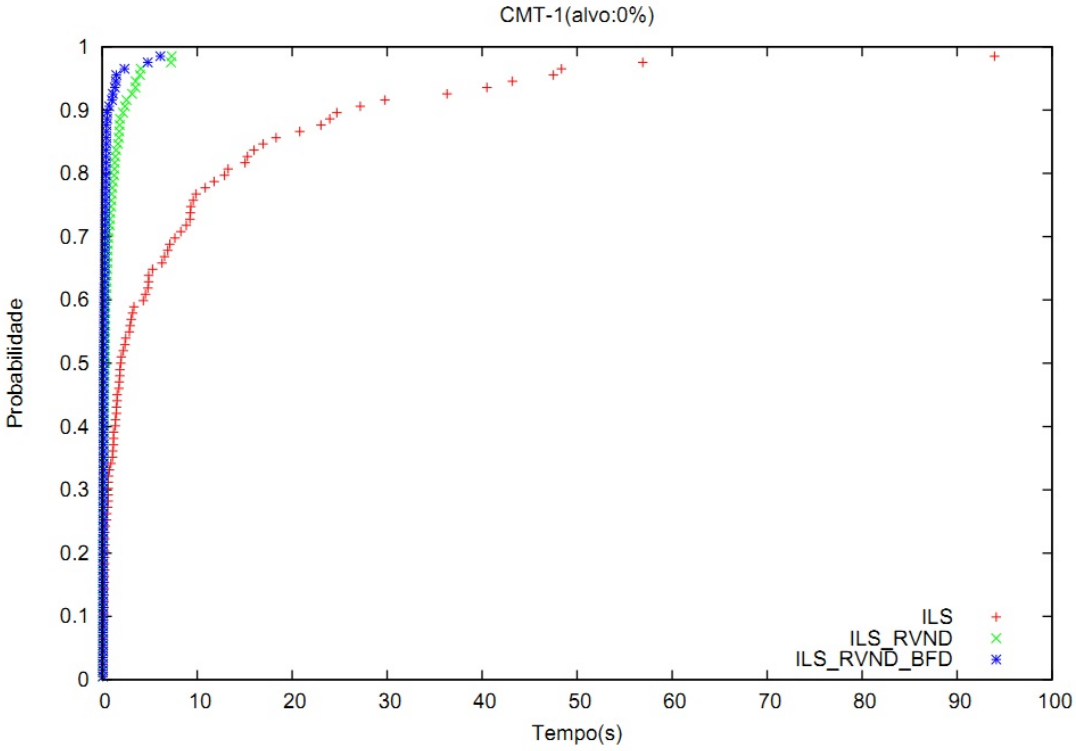


Figura 4.2: DPE para o problema teste CMT-1 alvo 0%

4.3.2 Comparativo de Tempos Computacionais

O comparativo realizado com relação ao tempo não é uma tarefa trivial, pois cada autor tem uma maneira diferente de codificar seus algoritmos e os programas são executados em

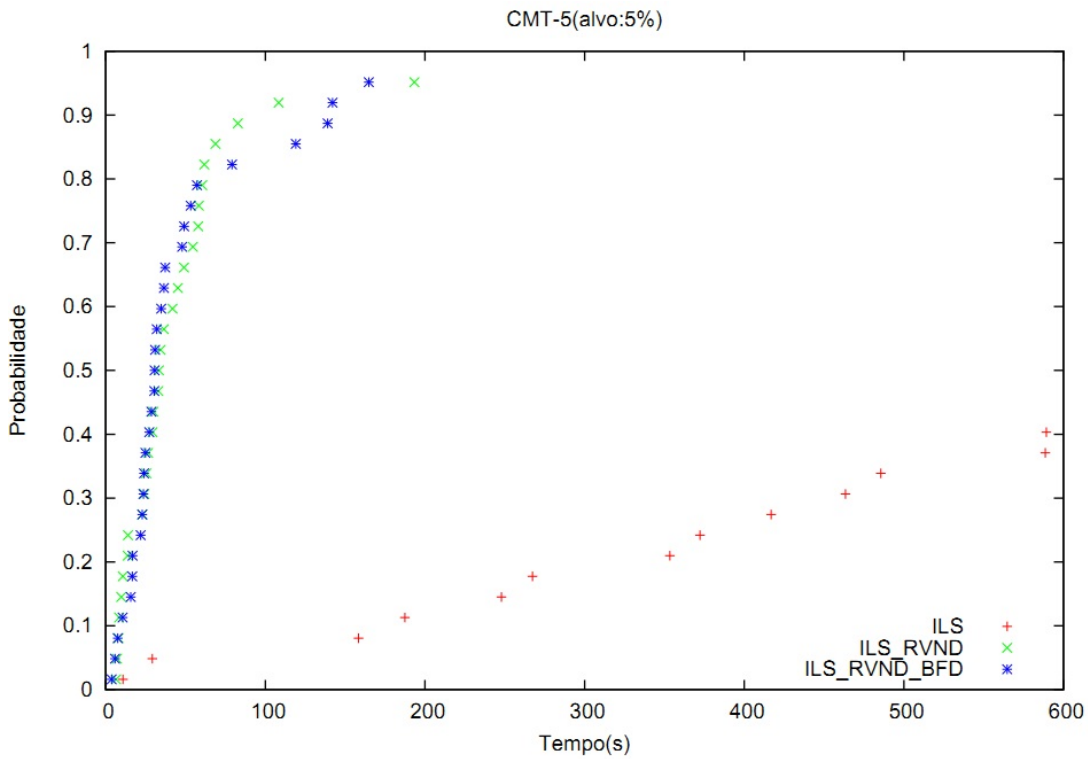


Figura 4.3: DPE para o problema teste CMT-5 alvo 5%

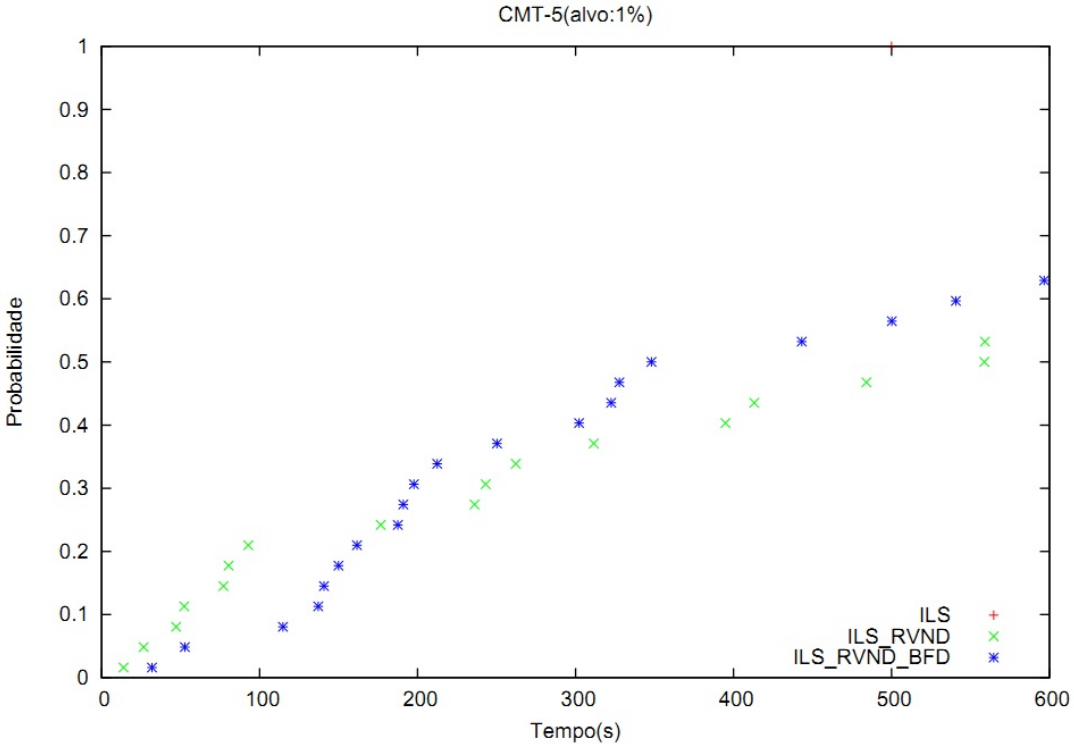


Figura 4.4: DPE para o problema teste CMT-5 alvo 1%

máquinas de configurações e modelos distintos. Tais fatores influenciam drasticamente no tempo computacional de execução de cada programa. Para criar-se um comparativo

mais justo foi utilizado o relatório de velocidade em *Milion of Floatin-Point Operations per Second* (Mflop/s) de Dongarra [13] que apresenta a velocidade de processamento de vários computadores. Caso o processador não seja encontrado, assume-se a velocidade de um processador com características semelhantes.

Pelo relatório de Dongarra [13], assume-se que as velocidades dos computadores em Mflop/s são: 172 MFlop/s - Ultra Enterprise 450 dual 300MHz , 15 MFlop/s - SGI Indigo2 Extreme R400/100MHz, 38 MFlop/s - Pentium Pro 200MHz, e 732 MFlop/s - AMD Athlon XP 2200+ 1.8GHz. No caso do processador AMD Turion X2 Ultra Dual-Core 2.10 GHz utilizado neste trabalho, obteve-se uma velocidade de 1385 MFlops/s.

Na Tabela 4.6 há um comparativo de tempo computacional em segundos com relação aos melhores algoritmos da literatura. Cada tempo é multiplicado por um fator obtido pela razão da velocidade da máquina sobre a menor velocidade encontrada entre os computadores comparados, que são: 11 - Petch e Salhi [40], 1 - Taillard [45], 2,5 - Brandão e Mercer [4], 11 - Petch e Salhi [35], 49 - Oliveira e Vieira [31] e 92 - ILS-RVND-BFD. O algoritmo proposto obteve tempos competitivos com relação às metaheurísticas comparadas, ficando 45% acima do tempo se comparado ao algoritmo de Brandão e Mercer [4] (que obteve a melhor média dos tempos). Quanto aos demais algoritmos de Petch e Salhi [40] , Taillard [45], Petch e Salhi [35] e Oliveira e Vieira [31], estes obtiveram um tempo maior (relativo ao algoritmo de Brandão e Mercer [4]) de 119%, 45%, 586% e 34% respectivamente.

Tabela 4.6: Tempo médio de CPU

Prob.	n	ILS-RVND-BFD	Petch e Salhi [40]	Taillard [45]	Brandão e Mercer [4]	Petch e Salhi[35]	Oliveira e Vieira [31]
CMT-1	50	257	175	300	150	1,188	784
CMT-2	75	550	330	420	300	3,630	1,421
CMT-3	100	976	770	1,440	600	9,108	1,323
CMT-4	150	3,564	2,271	3,060	1,500	10,824	3,332
CMT-5	199	7,310	5,319	3,960	3,750	26,994	6,125
CMT-11	120	2,971	12,448	2,700	1,500	26,730	1,372
CMT-12	100	835	498	1,380	600	1,320	1,323
F-11	71	497	1,026	1,560	150	2,838	637
F-12	134	2,425	6,426	4,500	4,800	8,910	1,519
Média		2,154	3,251	2,147	1,483	10,171	1,982

Capítulo 5

Conclusão

O problema de roteamento de veículos com múltiplas viagens tem grande relevância na área de transporte e logística das empresas, pois os veículos podem ser reutilizados durante um período de planejamento para o atendimento de outras viagens. Com essa realocação de veículos para atender outras rotas, temos uma diminuição do tempo de ociosidade da frota de veículos.

Portanto, ferramentas que encontrem soluções eficientes para resolução do PRVMV, são almejados pelos gestores, pois provocam a melhor gestão dos recursos da empresa no setor de transporte. O ILS-RVND-BFD vai de encontro a este problema gerando soluções que minimizem o tempo ocioso dos veículos e também diminui o custo total de deslocamento.

O algoritmo proposto baseado na metaheurística *Iterated Local Search* integrado com *Random Variable Neighborhood Descent* junto com o *Best Fit Decreasing*, mostrou resultados eficientes e robustos para solucionar o PRVMV. Para o conjunto de instâncias testadas, o algoritmo ILS-RVND-BFD mostrou um melhor desempenho, encontrando uma nova solução para PRVMV e o GAP foi reduzido em 39% com relação ao período de planejamento T_1 e 31% com relação a T_2 em comparação ao algoritmo de Oliveira e Vieira [31].

Com relação as soluções não viáveis, o algoritmo ILS-RVND-BFD apresentou uma redução na média da diferença de 11.71%, 39.53% e 67.14% com relação ao *Overtime* em comparação com os algoritmos de Oliveira e Vieira [31], Brandão e Mercer [4] e Taillard [45] respectivamente.

5.1 Trabalhos Futuros

Como trabalhos futuros, propõe-se a criação de um algoritmo híbrido que incorpora ao método ILS-RVND-BFD um método exato baseado no modelo de programação matemática $PRVMV$. Esse método buscaria aprimorar a melhor solução encontrada pelo algoritmo ILS-RVND-BFD. Essa estratégia tende a reduzir consideravelmente o tempo computacional, o que torna o seu uso atrativo para o problema.

Além disso, pretende-se estender o algoritmo proposto para tratar outras variantes do PRVMV tal como o Problema de Roteamento de Veículos com Múltiplas Viagens e Janela de Tempo (PRVMVJT) e o Problema de Roteamento de Veículos com Múltiplas Viagens e Coleta e Entrega (PRVMVCE).

Referências

[1] Ayadi, W.; Elloumi, M.; Hao, J.-k. Iterated local search for biclustering of microarray data. *Pattern Recognition in Bioinformatics* (2010), 219–229.

[2] Azi, N.; Gendreau, M.; Potvin, J.-Y. An exact algorithm for a single vehicle routing problem withtime windows and multiple routes. 755–766.

[3] Battarra, M.; Monaci, M.; Vigo, D. An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers & Operations Research 36*, 11 (2009), 3041 – 3050.

[4] Brandão, J.; Mercer, A. The multi-trip vehicle routing problem. *The Journal of the Operational Research Society 49*, 8 (1998), 799–805.

[5] Brito, J.; Ochi, L.; Montenegro, F.; Maculan, N. An iterative local search approach applied to the optimal stratification problem. *International Transactions in Operational Research 17*, 6 (2010), 753–764.

[6] Christofides, N.; Mingozzi, A.; Toth, P. The vehicle routing problem. en combinatorial optimization. *N. Christofides, A. Mingozzi, P. Toth, C. Sandi, editores* (1979), 315-338.

[7] Clarke, G.; Wright, J. Scheduling of vehicles for a central depot to a number of delivery points. *Operations Research 12* (1964), 568–581.

[8] Cochocki, A.; Unbehauen, R. *Neural Networks for Optimization and Signal Processing*, 1st ed. John Wiley & Sons, Inc., New York, NY, USA, 1993.

[9] Cotta, C.; Alba, E.; Troya, J. M. *Stochastic Reverse Hillclimbing and Iterated Local Search*, vol. 2. IEEE Press, 1999, pp. 1558–1565.

[10] Dantzig, G. B.; Ramser, J. H. The truck dispatching problem. *Management Science 6*, 1 (1959), 80–91.

[11] do Transporte, C. N. *Plano CNT de Transporte e Logística 2011*. Setor de Autarquias Sul, Quadra 01, Bloco J, Ed. CNT, 13° Andar, 2011.

[12] Dong, X.; Huang, H.; Chen, P. An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion. *Computers & Operations Research 36*, 5 (2009), 1664 – 1669. Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

[13] Dongarra, J. J. Performance of various computers using standard linear equations software. *Computer Science Mathematics Division* (June 2011).

[14] Dorronsoro, B.; Bouvry, P.; Alba, E. *Iterated Local Search for de Novo Genomic Sequencing*. 2010, pp. 428–436.

[15] Fischer, M. Optimal solution of vehicle routing problems using minimum k–trees. *Operations Research 42* (1994), 626-642.

[16] Fleischmann, B. The vehicle routing problem with multiple use of vehicles. Working paper, Fachbereich Wirtschaftswissenschaften, Universitat Hamburg.

[17] Fred Glover, G. A. K. *Handbook of Metaheuristics*. Boston, 2003.

[18] Garey, M.; Johnson, D. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman San Francisco, 1979.

[19] Garey, M.; Johnson, D. *Computers and intractability*, vol. 174. Freeman San Francisco, CA, 1979.

[20] Gen, M.; Cheng, R. *Genetic Algorithms and Engineering Optimization (Engineering Design and Automation)*. Wiley-Interscience, Dec. 1999.

[21] Glover, F.; Kluwer, M. L. *Tabu Search*. Kluwer Academic, 1997.

[22] Hansen, P.; Mladenovic, N. *Handbook of Metaheuristics*. Kluwer Academic Publishers, Norwell, MA, 2002, ch. Variable Neighborhood Search, pp. 145–184.

[23] Hashimoto, H.; Yagiura, M.; Ibaraki, T. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization 5*, 2 (2008), 434 – 456. In Memory of George B. Dantzig.

[24] Hernandez, F.; Feillet, D.; Giroudeau, R.; Naud, O. An exact method to solve the multi-trip vehicle routing problemwith time windows and limited duration.

[25] Hommes, P.; Howe, E.; Pape, C. Burton's biscuits: a study of the load planning operation atthe new depot at risley. Master's thesis, Lancaster University, 1989.

[26] Ibaraki, T.; Imahori, S.; Nonobe, K.; Sobue, K.; Uno, T.; Yagiura, M. An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Applied Mathematics 156*, 11 (2008), 2050 – 2069. In Memory of Leonid Khachiyan (1952 - 2005).

[27] JUNQUEIRA, N. M. P. Algoritmos aproximados para solucionar o problema de bin packing unidimensional. Master's thesis, Instituto Tecnológico de Aeronáutica, São José dos Campos, Brasil, 2007.

[28] Liddle, T. Kick strength and online sampling for iterated local search. *Operations Research*, November (2010), 130–139.

[29] Lourenço, H. R.; Martin, O. C.; Stützle, T. *Handbook of Metaheuristics*. Kluwer Academic Publishers, Norwell, MA, 2002, ch. Iterated Local Search, pp. 321–353.

[30] Mladenovic, N.; Hansen, P. Variable neighborhood search. *Computers & Operations Research 24*, 11 (1997), 1097 – 1100.

[31] Olivera, A.; Viera, O. Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research 34*, 1 (2007), 28 – 47.

[32] Paquete, L.; Stützle, T. An experimental investigation of iterated local search for coloring graphs. In *Applications of Evolutionary Computing*, S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. Raidl, Eds., vol. 2279 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2002, pp. 191–208. 10.1007/3-540-46004-7-13.

[33] PENG, Y.; PENG, J.; LI, H. Multi trip vehicle routing problem based on fuel consumptionreduction.

[34] Penna, P.; Subramanian, A.; Ochi, L. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics* (2011), 1–32. 10.1007/s10732-011-9186-y.

[35] Petch, R.; Salhi, S. A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics 133* (2003), 69 – 92.

[36] RESENDE, M. G.; RIBEIRO, C. C. Greedy randomized adaptive search procedures: Advances and applications. *Handbook of Metaheuristics* (July 2008).

[37] Resende, M. G. C.; Ribeiro, C. C. *Handbook of Metaheuristics*. Kluwer Academic Publishers, Norwell, MA, 2002, ch. Greedy Randomized Adaptive Search Prodedures, pp. 219–250.

[38] Roberti, R. Exact algorithms for different classes of vehicle routing problems. *4OR* (2012).

[39] Rochat, Y.; Taillard, . D. Probabilistic intensification and diversification in local search for vehicle routing. *Journal of Heuristics* (1995), 147–167.

[40] Salhi, S.; Petch, R. A ga based heuristic for the vehicle routing problem with multiple trips. *Journal of Mathematical Modelling and Algorithms 6* (2007), 591–613.

[41] Stützle, T.; Hoos, H. H. Analyzing the run-time behaviour of iterated local search for the tsp. In *III Metaheuristics International Conference* (1999), Kluwer Academic Publishers.

[42] Subramanian, A. *Metaheurística Iterated Local Search aplicada ao problema de roteamento de veículos com coleta e entrega simultânea*. Tese de Doutorado, 2008.

[43] Subramanian, A.; Battarra, M. An iterated local search algorithm for the travelling salesman problem with pickups and deliveries. *J Oper Res Soc* (2012), 1476–9360.

[44] Subramanian, A.; Drummond, L. M. A.; Bentes, C.; Ochi, L. S.; Farias, R. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research 37*, 11 (2010), 1899–1911.

[45] Taillard, E. D.; Laporte, G.; Gendreau, M. Vehicle routing with multiple use of vehicles. *THE JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY 47*, 8 (1995), 1065–1070.

[46] van Laarhoven, P. J. M.; Aarts, E. H. L. *Simulated Annealing: Theory and Applications*, vol. 37. Kluwer Academic, 1987.

[47] Zhao, Q.; Wang, S.; Lai, K.; Xia, G. A vehicle routing problem with multiple use of vehicles. *Advanced Modeling and Optimization 4* (2002), 21–40.

APÊNDICE A – Resultados com relação ao GAP

Tabela A.1: Resultados detalhados para instâncias CMT-1, CMT-2 e CMT-3

			GAP ILS-RVND-BFD			GAP AMP [31]						
Problema	m	T	Melhor	Media	Pior	Melhor	Media	Pior	TLG	BM	PS	GA
CMT-1	1	551	0.0	0.0	0.0	0.0	0.0	0.0	●	●	●	●
$n = 50$	2	275	1.6	1.7	2.2	1.6	1.7	2.2	●	●	○	○
$z^* = 524.61$	3	184	-	-	-	-	-	-	○	○	○	○
	4	138	-	-	-	-	-	-	○	○	○	○
	1	577	0.0	0.0	0.0	0.0	0.0	0.0	●	●	●	●
	2	289	1.0	1.3	1.6	1.0	1.1	1.6	●	●	●	●
	3	192	5.3	6.2	6.6	5.4	5.5	6.0	○	●	●	●
	4	144	4.1	4.2	4.3	4.1	4.2	4.3	●	●	○	●
CMT-2	1	877	<0.1	0.2	0.4	0.0	0.2	0.3	●	●	●	●
$n = 75$	2	439	<0.1	<0.1	0.1	0.2	0.5	1.0	●	●	●	●
$z^* = 835.26$	3	292	0.0	0.1	0.1	0.2	0.5	1.3	●	●	●	○
	4	219	0.0	0.2	0.5	0.1	0.7	1.7	●	●	●	●
	5	175	0.1	0.3	1.0	1.2	1.6	2.1	●	●	●	○
	6	146	2.7	2.7	2.7	2.6	2.6	2.6	○	○	○	○
	7	125	-	-	-	-	-	-	○	○	○	○
	1	919	0.0	0.1	0.3	0.1	0.7	1.3	●	●	●	●
	2	459	<0.1	0.4	0.9	0.1	0.5	1.0	●	●	●	●
	3	306	0.1	0.5	0.8	0.1	0.4	0.8	●	●	●	●
	4	230	0.1	0.5	0.7	0.1	0.2	0.6	●	●	●	●
	5	184	<0.1	0.5	1.2	0.1	0.9	1.3	●	●	●	●
	6	153	0.9	2.3	2.9	2.5	3.4	4.8	●	●	●	○
	7	131	2.4	3.2	3.7	3.6	4.2	4.7	○	●	○	○
CMT-3	1	867	0.0	0.1	0.3	0.0	0.4	0.6	●	●	●	●
$n = 100$	2	434	0.0	0.1	0.6	0.4	0.5	1.0	●	●	●	●
$z^* = 826.14$	3	289	0.0	0.4	0.7	0.5	0.7	0.9	●	●	●	○
	4	217	0.3	0.4	0.5	0.4	0.8	1.2	●	●	●	○
	5	173	0.9	2.0	2.7	2.6	3.2	4.1	○	●	○	○
	6	145	1.2	2.5	3.3	1.2	2.0	3.0	○	○	○	○
	1	909	0.0	0.2	0.4	0.4	0.5	0.7	●	●	●	●
	2	454	0.0	<0.1	0.1	0.3	0.4	0.7	●	●	●	●
	3	303	0.0	0.1	0.4	0.2	0.4	0.4	●	●	●	●
	4	227	0.0	0.2	0.5	0.0	0.3	0.5	●	●	●	●
	5	182	1.0	1.1	1.2	0.8	1.0	1.9	●	●	●	●
	6	151	1.1	1.4	2.2	1.1	1.7	1.9	●	●	●	●

Tabela A.2: Resultados detalhados para instâncias CMT-4 e CMT-5

			GAP ILS-RVND-BFD			GAP AMP [31]						
Problema	m	T	Melhor	Media	Pior	Melhor	Media	Pior	TLG	BM	PS	GA
CMT-4	1	1080	0.3	0.6	0.8	0.5	0.9	1.3	●	●	●	●
$n = 150$	2	540	0.0	0.5	1.2	0.8	1.7	3.5	●	●	●	●
$z^* = 1028.42$	3	360	0.6	1.0	1.3	0.7	1.3	1.8	●	●	●	○
	4	270	0.2	0.8	1.4	0.8	1.6	2.5	●	●	○	○
	5	216	<0.1	0.7	1.5	0.4	1.6	2.8	●	●	●	○
	6	180	0.8	1.6	2.5	2.0	3.2	4.5	●	●	○	○
	7	**154**	3.5	3.5	3.5	-	-	-	○	○	○	○
	8	135	2.7	2.7	2.7	3.0	3.6	4.3	○	○	○	○
	1	1131	0.3	0.9	1.3	1.0	1.3	1.6	●	●	●	●
	2	566	0.2	0.4	0.8	0.9	1.5	2.1	●	●	●	●
	3	377	0.3	0.7	1.6	0.5	1.1	1.6	●	●	●	●
	4	283	0.4	0.7	1.4	1.0	1.7	2.5	●	●	●	●
	5	226	0.2	1.0	1.4	0.8	1.5	2.3	●	●	●	●
	6	189	0.5	1.0	1.4	0.4	1.4	2.0	●	●	●	●
	7	162	1.2	2.2	2.6	2.6	3.3	3.9	●	●	○	○
	8	141	1.7	2.6	2.9	3.5	4.4	5.6	○	●	●	○
CMT-5	1	1356	1.6	1.7	1.8	1.9	2.8	3.4	●	●	●	●
$n = 199$	2	678	1.6	1.7	1.9	2.0	3.1	3.9	●	●	●	●
$z^* = 1291.44$	3	452	1.2	1.5	1.7	1.6	2.1	2.8	●	●	●	○
	4	339	1.0	1.7	2.3	2.5	3.0	3.2	●	●	●	○
	5	271	1.4	2.3	3.3	2.5	3.1	4.1	●	●	○	○
	6	226	1.5	2.2	2.9	3.1	3.5	4.0	●	●	●	○
	7	194	1.7	2.0	2.7	3.6	3.8	4.1	●	●	○	○
	8	170	1.1	1.9	2.5	2.8	3.5	4.2	●	●	○	○
	9	151	1.8	2.9	3.5	3.4	3.7	4.0	●	○	○	○
	10	136	2.8	2.8	2.8	4.1	4.1	4.1	○	○	○	○
	1	1421	1.0	1.4	1.7	2.1	2.7	3.5	●	●	●	●
	2	710	1.3	1.6	1.8	1.8	2.4	2.9	●	●	●	●
	3	474	1.2	1.5	1.9	1.6	2.3	2.6	●	●	●	●
	4	355	1.4	1.5	1.7	1.3	2.4	3.6	●	●	●	●
	5	284	1.6	1.7	1.7	2.0	2.7	3.3	●	●	●	●
	6	237	1.2	1.5	1.8	1.5	2.7	4.1	●	●	●	●
	7	203	1.5	1.6	2.0	2.3	3.2	3.7	●	●	●	●
	8	178	1.2	1.4	1.6	1.8	2.0	2.3	●	●	●	○
	9	158	1.2	1.5	1.7	3.0	3.4	4.2	●	●	●	○
	10	142	1.8	2.7	3.3	4.1	4.6	5.0	●	●	○	○

Tabela A.3: Resultados detalhados para instâncias CMT-11, CMT-12, F-11 e F-12

			GAP ILS-RVND-BFD			GAP AMP [31]						
Problema	m	T	Melhor	Media	Pior	Melhor	Media	Pior	TLG	BM	PS	GA
CMT-11	1	1094	0.0	1.9	3.0	0.1	2.5	3.2	●	●	●	●
$n = 120$	2	547	0.0	2.4	3.0	3.0	3.1	3.2	●	●	○	○
$z^* = 1042.11$	3	365	0.0	2.3	3.7	0.2	2.7	4.5	●	●	○	○
	4	274	3.5	3.5	3.5	4.2	4.3	4.4	○	○	○	○
	5	219	0.0	0.0	0.0	0.1	0.8	1.3	●	●	○	○
	1	1146	0.1	2.4	3.0	0.2	2.5	3.1	●	●	●	●
	2	573	0.4	4.4	8.7	1.2	2.6	3.1	●	●	●	●
	3	382	0.0	3.3	9.51	0.1	0.8	3.3	●	●	●	●
	4	287	0.0	0.1	0.7	0.1	0.2	0.4	●	●	○	○
	5	229	0.0	2.7	3.4	0.1	0.3	0.6	●	●	●	●
CMT-12	1	861	0.0	0.0	0.0	0.0	0.1	0.2	●	●	●	●
$n = 100$	2	430	0.0	0.0	0.0	<0.1	0.2	0.2	●	●	●	●
$z^* = 819.56$	3	287	0.0	0.0	0.0	0.0	<0.1	<0.1	●	●	●	●
	4	215	0.1	0.1	0.1	0.0	0.0	0.0	●	○	●	●
	5	172	3.1	3.1	3.1	3.1	3.3	3.4	○	○	●	○
	6	143	-	-	-	-	-	-	○	○	○	○
	1	902	0.0	0.0	0.0	0.0	<0.1	<0.1	●	●	●	●
	2	451	0.0	0.0	0.0	0.0	<0.1	<0.1	●	●	●	●
	3	301	0.0	0.0	0.0	0.0	<0.1	<0.1	●	●	●	●
	4	225	0.0	0.0	0.0	0.0	<0.1	<0.1	●	●	●	●
	5	180	0.6	0.6	0.6	0.6	0.6	0.6	●	●	●	●
	6	150	0.4	0.8	2.3	0.4	0.5	0.7	●	●	●	●
F-11	1	254	0.0	0.0	0.0	0.0	0.3	0.7	●	●	●	○
$n = 71$	2	127	3.7	3.7	3.7	4.3	4.3	4.3	○	○	○	○
$z^* = 241.97$	3	85	-	-	-	-	-	-	○	○	○	○
	1	266	0.0	0.0	0.0	0.0	0.4	0.7	●	●	●	●
	2	133	0.0	0.8	1.7	0.0	0.8	1.8	●	●	●	●
	3	89	5.4	6.4	7.9	5.4	7.0	8.2	○	●	●	●
F-12	1	1221	0.0	1.4	2.1	0.5	0.6	0.9	●	●	●	●
$n = 134$	2	611	0.0	2.2	3.5	0.7	0.9	1.1	●	●	●	●
$z^* = 1162.96$	3	407	0.0	1.5	2.7	0.3	0.5	0.7	●	●	●	●
	1	1279	0.0	1.7	2.1	0.8	1.0	1.3	●	●	●	●
	2	640	0.0	1.0	2.4	0.7	0.9	1.0	●	●	●	●
	3	426	0.3	1.7	2.2	0.4	0.8	1.0	●	●	●	●

APÊNDICE B – Nova Solução para o PRVMV

Tabela B.1: Nova Solução CMT-4 com m=7 e T_1 = 154

Veículo	Duração	Sequência de Clientes
1	152.730873	(0 6 132 96 86 43 99 114 113 26 140 82 31 28 116 70 22 0)
2	148.104473	(0 104 30 105 75 74 79 21 128 84 35 85 36 115 121 3 101 51 32 0)
3	50.526527	(0 139 47 146 149 4 143 135 148 109 144 0)
3	103.246205	(0 17 142 87 150 141 41 94 19 64 88 40 136 66 111 56 0)
4	58.624501	(0 78 126 16 118 50 130 34 9 62 38 0)
4	94.980772	(0 76 49 10 54 39 89 117 73 106 122 71 0)
5	64.140630	(0 77 119 1 120 80 8 60 81 27 46 0)
5	89.551148	(0 102 57 98 24 97 23 69 7 61 112 48 138 0)
6	68.489544	(0 103 108 52 15 45 91 72 33 125 124 123 90 5 0)
6	85.275729	(0 68 133 14 58 25 95 67 13 134 55 18 110 0)
7	73.696645	(0 11 100 2 83 131 59 20 29 129 53 127 0)
7	80.155114	(0 12 145 147 92 42 93 65 107 44 137 37 63 0)