UNIVERSIDADE FEDERAL FLUMINENSE

MARCOS DE MELO DA SILVA

Uma heurística baseada em Iterated Local Search para o Problema de Roteamento de Veículos com Entregas Fracionárias

NITERÓI

2013

UNIVERSIDADE FEDERAL FLUMINENSE

MARCOS DE MELO DA SILVA

Uma heurística baseada em Iterated Local Search para o Problema de Roteamento de Veículos com Entregas Fracionárias

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:
Luiz Satoru Ochi

NITERÓI

2013

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

S586 Silva, Marcos de Melo da
Uma heurística baseada em Iterated Local Search para o problema de roteamento de veículos com entregas fracionárias / Marcos de Melo da Silva. – Niterói, RJ : [s.n.], 2013.
67 f.

Dissertação (Mestrado em Computação) - Universidade Federal Fluminense, 2013.
Orientador: Luiz Satoru Ochi.

1. Problema de roteamento de veículo. 2. Algoritmo heurístico. 3. Ciência da computação. 4. Heurística. I. Título.

CDD 005.136

Uma heurística baseada em *Iterated Local Search* para o Problema de Roteamento de Veículos com Entregas Fracionárias

Marcos de Melo da Silva

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 por:

Prof. Luiz Satoru Ochi
IC-UFF (Presidente)

Prof. Anand Subramanian
UFPB

Prof. Virgílio José Martins Ferreira Filho
COPPE/UFRJ

Prof. Yuri Abitbol de Menezes Frota
IC-UFF

Niterói
2013

Aos meus pais.

Agradecimentos

Apesar deste espaço ser dedicado aos agradecimentos, a palavra que primeiro me vem à mente e que gostaria de dirigir a todos os citados a seguir é: desculpe.

Agora ao real objetivo desta parte do trabalho. Agradeço aos meus pais, irmãs e irmãos pelo suporte, compreensão e paciência. Também agradeço pela *quasi*-liberdade e momentos de alegria. Agradeço aos bons amigos que conheci no decorrer da vida. Agradeço aos amigos da *OptHouse*, Edcarllos, Igor, Mário, Matheus, Pablo e Sabir, por me acolherem e proporcionarem momentos de alegria. Agradeço aos muitos professores que, seja por meio de bons ensinamentos ou maus exemplos, muito cooperaram para minha formação. Agradeço ao Prof. Satoru pela orientação e amizade. Agradeço também ao amigo e pseudo-coorientador Anand pela orientação. Agradeço às secretárias da pós, Teresa e Viviane, pelo auxílio e paciência. Agradeço ao CNPq e a Capes pelo apoio financeiro, sem o qual seria difícil o bom andamento dos estudos.

Resumo

O Problema de Roteamento de Veículos com Entregas Fracionárias (PRVEF) é obtido através da relaxação da restrição que garante o atendimento das demandas dos clientes em uma única visita no Problema de Roteamento de Veículos Capacitado (PRVC), permitindo, assim, que as demandas sejam divididas em frações e cada uma destas atendida por um veículo diferente. Em relação ao dimensionamento da frota, duas versões do problema são reportadas na literatura, uma considerando frota ilimitada (PRVEF-FI) e a outra uma frota limitada (PRVEF-FL). Ambas as versões serão abordadas neste trabalho. Como método de resolução, foi proposta uma heurística *multi-start* baseada na meta-heurística *Iterated Local Search* (ILS) que incorpora na fase de busca local, além das estruturas de vizinhanças clássicas para o PRVC, vizinhanças dedicadas especificamente para explorar as características do PRVEF. Para mostrar a eficácia do método, experimentos computacionais foram realizados em quatro conjuntos de instâncias disponíveis na literatura, totalizando 324 problemas-teste, somando-se para as duas versões do problema. Os resultados obtidos são bastante competitivos, visto que foram encontradas novas melhores soluções em mais de 75% das instâncias.

Palavras-chave: Problema de Roteamento de Veículos, Entregas Fracionárias, Algoritmo Heurístico, *Iterated Local Search*.

Abstract

The Split Delivery Vehicle Routing Problem (SDVRP) is obtained from the Capacitated Vehicle Routing Problem (CVRP) by removing the constraints that ensure accomplishment of all client demands in only one visit, in such way that the demands are split in several pieces and each one served by a vehicle. Considering the fleet size, two versions of the problem are reported in the literature, the first one consist of an unlimited fleet (SDVRP-UF) and the second comprises a limited fleet (SDVRP-LF). This work deals with both versions. In order to solve them, we propose a multi-start Iterated Local Search (ILS) based heuristic that includes classical CVRP neighborhood structures, as well as those specifically designed for the SDVRP, in the local search phase. Extensive computational experiments were carried out on benchmark instances available in the literature, totaling 324 test problems for both versions of the problem, in order to show the effectiveness of the method. The results obtained are highly competitive, since it was found improved solutions in more than 75% instances.

Keywords: Vehicle Routing Problem, Split Deliveries, Heuristic Algorithm, Iterated Local Search.

Sumário

Lista de Figuras viii

Lista de Tabelas ix

Lista de Abreviaturas e Siglas x

1 Introdução 1

2 O Problema de Roteamento de Veículos com Entregas Fracionárias 5

2.1 Formulação Matemática para o PRVEF 6

2.2 Propriedades e Benefícios de Entregas Fracionárias 7

3 Revisão da Literatura 9

3.1 Abordagens Exatas 9

3.2 Abordagens Heurísticas 10

3.3 Outros Trabalhos Relacionados ao PRVEF 13

4 Metodologia 15

4.1 Representação e Avaliação de Soluções 17

4.2 Procedimentos Auxiliares 19

4.3 Métodos Construtivos 21

4.3.1 Critérios de Inserção 22

4.3.2 Estratégias de Inserção 23

4.4 Busca Local 25

4.4.1 Vizinhanças Inter-rotas 25

4.4.2 Vizinhanças Intrarrotas 31

4.5 Mecanismos de Perturbação 34

4.6 Algoritmo Proposto 34

5 Resultados Computacionais **36**

5.1 Resultados para o PRVEF com Frota Limitada 37

5.2 Resultados para o PRVEF com Frota Ilimitada 43

5.3 Resumo dos Resultados 48

6 Conclusões e Trabalhos Futuros **49**

Referências **51**

Lista de Figuras

4.1 Representação de uma solução. 18

4.2 Vizinhança inter-rotas *Shift(1,0)*. 26

4.3 Vizinhança inter-rotas *Swap(1,1)*. 27

4.4 Vizinhança inter-rotas *Shift(2,0)*. 27

4.5 Vizinhança inter-rotas *Swap(2,1)*. 28

4.6 Vizinhança inter-rotas *Swap(2,2)*. 28

4.7 Vizinhança inter-rotas *Cross*. 29

4.8 Vizinhança inter-rotas *Swap*(1,1)*. 29

4.9 Vizinhança inter-rotas *Swap*(2,1)*. 30

4.10 Vizinhança inter-rotas *RouteAdition*. 31

4.11 Vizinhança intrarrotas *Exchange*. 32

4.12 Vizinhança intrarrotas *2-opt*. 32

4.13 Vizinhança intrarrotas *Reinserção*. 32

4.14 Vizinhança intrarrotas *Or-opt2*. 33

4.15 Vizinhança intrarrotas *Or-opt3*. 33

Lista de Tabelas

5.1 Resultados para o PRVEF-FL nas instâncias de Belenguer et al. [14] . . . 38

5.2 Resultados para o PRVEF-FL nas instâncias de Belenguer et al. [14] usando custos arredondados . . . 39

5.3 Resultados para o PRVEF-FL nas instâncias de Campos et al. [19] 40

5.4 Resultados para o PRVEF-FL nas instâncias de Archetti et al. [13] 41

5.5 Resultados para o PRVEF-FL nas instâncias de Chen et al. [21] 42

5.6 Resultados para o PRVEF-FI nas instâncias de Belenguer et al. [14] 43

5.7 Resultados para o PRVEF-FI nas instâncias de Belenguer et al. [14] usando custos arredondados . . . 44

5.8 Resultados para o PRVEF-FI nas instâncias de Campos et al. [19] 45

5.9 Resultados para o PRVEF-FI nas instâncias de Archetti et al. [13] 46

5.10 Resultados para o PRVEF-FI nas instâncias de Chen et al. [21] 47

5.11 Resumo dos resultados obtidos pelo SplitILS 48

Lista de Abreviaturas e Siglas

AG	:	Algoritmo Genético;
BCP	:	*Branch-and-Cut-and-Price*;
BT	:	Busca Tabu;
ILS	:	*Iterated Local Search*;
PCV	:	Problema do Caixeiro Viajante;
PI	:	Programação Inteira;
PIM	:	Programação Inteira Mista;
PRV	:	Problema de Roteamento de Veículos;
PRVEF	:	Problema de Roteamento de Veículos com Entregas Fracionárias;
PRVEF-FI	:	PRVEF com Frota Ilimitada;
PRVEF-FL	:	PRVEF com Frota Limitada;
PRVJT	:	Problema de Roteamento de Veículos com Janelas de Tempo;
RTR	:	*Record-to-Record*;
SDVRP	:	*Split Delivery Vehicle Routing Problem*;
VND	:	*Variable Neighborhood Descent*;
VRP	:	*Vehicle Routing Problem*.

Capítulo 1

Introdução

O Problema de Roteamento de Veículos (PRV), em inglês *Vehicle Routing Problem* (VRP), teve sua origem a partir do *Truck Dispatcher Problem*, um problema prático de distribuição de combustíveis, descrito em 1959 por Dantzig e Ramser [24], cujo objetivo era determinar os roteiros de visitas com menor custo a serem seguidos por uma frota de caminhões para suprir um conjunto de postos de revenda. Nas décadas seguintes, o PRV tornou-se um dos problemas mais estudados nas áreas de otimização combinatória e pesquisa operacional, devido tanto a sua importância teórica quanto aplicabilidade em situações reais [23].

Uma definição para a versão clássica do PRV é apresentada a seguir. Esta será útil posteriormente no entendimento do problema abordado neste trabalho, uma vez que este pode ser descrito tomando como base o PRV. Desta maneira, considere uma frota homogênea de veículos posicionados na origem (depósito) e um conjunto de clientes geograficamente distribuídos, cujas demandas e os custos entre cada par de vértices do grafo associado são previamente conhecidos, o objetivo no PRV é determinar um conjunto de rotas de custo mínimo que visitem todos os clientes, respeitando as seguintes restrições:

i - Toda rota inicie e termine no depósito;

ii - A soma das demandas dos clientes atendidos por uma dada rota não exceda a capacidade do veículo;

iii - Cada cliente tenha toda sua demanda suprida em uma única visita.

No decorrer do tempo, uma série de outras restrições ou características adicionais foram sendo incorporadas a essa versão clássica do PRV para tratar elementos presentes

em situações práticas. Dando origem, assim, a diversas variantes do problema. Alguns desses elementos são:

- Horizonte de planejamento: um ou mais dias (unidades);
- Número de depósitos: único ou múltiplos depósitos;
- Número de viagens por veículo: única ou múltiplas;
- Restrições de tempo: janelas de tempo e limite de tempo para concluir uma rota;
- Tipo de frota: homogênea ou heterogênea;
- Tipo de operação: coleta, entrega ou ambos;
- Atendimento das demandas: em uma ou mais visitas.

Dentre os elementos apresentados, a variante na qual o atendimento das demandas pode ser feito em mais de uma visita resulta da relaxação da restrição (iii) no problema descrito anteriormente, permitindo assim que a demanda de cada cliente possa ser dividida em varias frações, e cada uma destas, possa ser atribuída a uma rota diferente. Uma consequência direta disto é que os clientes podem ter demandas superiores a capacidade do veículo [29]. Esta versão é referenciada na literatura como Problema de Roteamento de Veículos com Entregas Fracionárias (PRVEF), em inglês *Split Delivery Vehicle Routing Problem* (SDVRP), e será o tema principal deste trabalho.

O desenvolvimento de estudos envolvendo o PRV e suas variantes são motivados tanto por questões práticas quanto teóricas. Do ponto de vista prático, o deslocamento eficiente de bens e pessoas é uma questão de grande importância para as empresas. Estimativas sugerem que os custos de distribuição respondem por quase a metade dos custos logísticos. Em algumas industrias, como as de bebidas e alimentos, os custos de transporte são responsáveis por até 70% do valor final dos produtos [18]. Assim, uma vez que no Brasil boa parte da distribuição ainda é feita por meio de transporte rodoviário, o desenvolvimento de técnicas eficientes capazes de resolver o PRV e variantes poderá gerar economias significativas. Toth e Vigo [61] relatam que o uso de procedimentos computacionais no planejamento do processo de distribuição resulta em uma redução da ordem de 5% a 20% nos custos de transporte.

O interesse teórico no PRV, se deve ao fato de ser considerado um problema complexo, visto que sua versão de otimização pertence à classe de problemas $\mathcal{NP}$-difícil, ou

seja, não se conhece algoritmos capazes de gerar, para qualquer instância de entrada, uma solução ótima em tempo polinomial. Apesar disso, muitos pesquisadores obtiveram grandes avanços no desenvolvimento de algoritmos exatos para tratá-lo, principalmente aqueles baseados em programação matemática. Contudo, mesmo os melhores algoritmos exatos existentes conseguem resolver de forma sistemática apenas instâncias com poucas centenas de clientes, isto ao custo de muitas horas de processamento computacional. Assim, estratégias baseadas em heurísticas ou meta-heurísticas são mais comumente utilizadas para resolver o PRV e suas variantes. Apesar destas não garantirem a otimalidade, sendo portando métodos aproximados, usualmente são capazes de gerar soluções de boa qualidade em tempos computacionais aceitáveis.

Este trabalho apresenta uma heurística *multi-start* baseada na meta-heurística *Iterated Local Search* (ILS) [45] para tratar o PRVEF. Na versão básica do ILS, a cada iteração a solução corrente passa por uma etapa de diversificação, caraterizada pela aplicação de um mecanismo de perturbação, e uma etapa de intensificação, na qual a solução perturbada é melhorada por meio de uma busca local. Na heurística desenvolvida, o algoritmo utilizado para explorar o espaço de soluções na fase de busca local é baseado no procedimento *Variable Neighborhood Descent* (VND) [39]. Em relação às estruturas de vizinhanças, além das estruturas clássicas para o PRV, foram incorporadas vizinhanças dedicadas especificamente para explorar as características do PRVEF.

Além da lista de exemplos de aplicação do ILS aos mais diversos problemas de otimização apresentada por Lourenço et al. [45], muitas outras abordagens que empregaram essa meta-heurística mostraram-se muito eficientes quando aplicadas a uma variedade de problemas de roteamento, como pode ser visto em trabalhos recentes de Subramanian et al. [58], Penna et al. [50], Silva et al. [54] e Subramanian e Battarra [57]. O algoritmo proposto difere dos desenvolvidos por esses autores ao incorporar procedimentos especialmente dedicados ao PRVEF, que por sua vez desempenham um papel crucial na busca por soluções de boa qualidade. Como resultado, várias novas soluções são reportadas. Além disso, quando comparado com outras abordagens heurísticas presentes na literatura, o algoritmo desenvolvido depende de poucos parâmetros, diminuindo assim os esforços de calibração.

O restante deste trabalho está organizado conforme descrito a seguir. O Problema de Roteamento de Veículos com Entregas Fracionárias (PRVEF), propriedades conhecidas e formulações matemáticas são apresentados no Capítulo 2. A revisão dos principais trabalhos da literatura que propuseram abordagens exatas e heurísticas para resolver o

PRVEF, ou que de alguma outra forma exploraram o problema, é apresentada no Capítulo 3. O Capítulo 4 descreve a abordagem proposta para tratar o PRVEF. Os experimentos computacionais e os resultados obtidos são discutidos no Capítulo 5. O Capítulo 6 contém as conclusões e trabalhos futuros.

Capítulo 2

O Problema de Roteamento de Veículos com Entregas Fracionárias

O PRVEF foi proposto por Dror e Trudeau [29] no final da década de 80. Nestes trabalhos, além de descrever o problema e apresentar uma formulação matemática, os autores também desenvolveram algoritmos heurísticos para resolvê-lo e estudaram empiricamente os ganhos obtidos, em termos de distância percorrida e quantidade de veículos utilizados, quando comparado ao PRV. Além disso, os autores também analisaram propriedades que podem ser observadas em soluções ótimas.

O PRVEF é formalmente definido a seguir. Seja $G = (V, E)$ um grafo não orientado, onde $V = \{0, \ldots, n\}$ é o conjunto de vértices e $E = \{\{i, j\} : i, j \in V, i < j\}$ é o conjunto de arestas. O depósito é representado pelo vértice 0, no qual uma frota de K veículos idênticos de capacidade Q estão posicionados, enquanto os outros vértices, $V' = \{1, \ldots, n\}$, representam os clientes, cada um com uma demanda positiva d_i ($i \in V; d_0 = 0$). Um custo de viagem não negativo c_{ij} está associado a cada aresta $\{i, j\} \in E$. O objetivo é minimizar a soma dos custos de transporte de forma que:

i A capacidade do veículo não seja excedida;

ii As demandas de todos os clientes sejam atendidas;

iii Toda rota tenha início e fim no depósito;

Duas versões do PRVEF são estudadas na literatura, uma que considera uma frota limitada (PRVEF-FL) e a outra com frota ilimitada (PRVEF-FI). Em relação ao PRVEF-FL, o número mínimo de veículos, K_{min}, deve ser utilizado. Pode-se verificar que sempre existirá uma solução viável considerando tal frota de veículos. Diferente do PRV, cujo o

número mínimo de veículos necessários é encontrado ao resolver um Problema de Empacotamento (*Bin Packing Problem*), no PRVEF-FL, esta quantidade é facilmente calculada através da equação (2.1), sendo ela o menor inteiro, imediatamente maior que a soma das demandas dos clientes dividido pela capacidade do veículo. As duas versões do problema serão estudadas neste trabalho.

$$K_{min} = \lceil \sum_{i=1}^{n} d_i/Q \rceil \tag{2.1}$$

Embora seja uma relaxação do PRV, Dror e Trudeau [30] mostraram que o PRVEF continua sendo $\mathcal{NP}$-difícil. Archetti et al. [8] provaram que o PRVEF pode ser resolvido em tempo polinomial quando a capacidade do veículo é dois e as demandas dos clientes são unitárias. Além disso, Archetti et al. [7] estudaram a complexidade das duas versões do problema considerando grafos com estruturas especiais, a saber: linha, árvore, estrela e círculo. Em relação ao PRVEF-FL, os autores mostraram que o problema tem complexidade $\mathcal{O}(n^2)$ quando o grafo é um círculo, $\mathcal{O}(n)$ quando o grafo é uma linha e $\mathcal{NP}$-difícil quando o grafo é uma árvore ou uma estrela. Assim como no PRVEF-FL, provaram que o PRVEF-FI é $\mathcal{NP}$-difícil quando o grafo é uma árvore e $\mathcal{O}(n^2)$ quando o grafo é um círculo, mas é $\mathcal{O}(n)$ quando o grafo é uma linha ou uma estrela.

Na literatura que aborda o problema de forma exata, diversas formulações matemáticas para o PRVEF foram propostas por vários autores, a maioria delas baseada em Programação Inteira Mista (PIM). A formulação proposta por Archetti et al. [12] será descrita a seguir. Para tal exposição, a definição do PRVEF mostrada acima será empregada.

2.1 Formulação Matemática para o PRVEF

A formulação apresentada a seguir foi proposta por Archetti et al. [12]. Este modelo utiliza as seguintes variáveis.

- x_{ij}^k é uma variável binária que assume valor 1 se a aresta (i, j) é visitada pelo veículo k, e assume valor 0, caso contrário;
- $y_{ik} \geq 0$ é a quantidade da demanda requerida pelo cliente i entregue pelo veículo k.

$$F_{Arch} = \text{Minimizar} \sum_{k=1}^{K} \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij}^k \tag{2.2}$$

$$\text{Sujeito a:} \qquad \sum_{k=1}^{K}\sum_{i=0}^{n} x_{ij}^{k} \geq 1 \qquad j = 1,\ldots,n \tag{2.3}$$

$$\sum_{i=0}^{n} x_{ip}^{k} - \sum_{j=0}^{n} x_{pj}^{k} = 0 \qquad p = 0,\ldots,n;\ k = 1,\ldots,K \tag{2.4}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij}^{k} \leq |S| - 1 \qquad k = 1,\ldots,K;\ S \subseteq V' \tag{2.5}$$

$$y_{ik} \leq d_i \sum_{j=0}^{n} x_{ij}^{k} \qquad i = 1,\ldots,n;\ k = 1,\ldots,K \tag{2.6}$$

$$\sum_{k=1}^{K} y_{ik} = d_i \qquad i = 1,\ldots,n \tag{2.7}$$

$$\sum_{i=1}^{n} y_{ik} \leq Q \qquad k = 1,\ldots,K \tag{2.8}$$

$$x_{ij}^{k} \in \{0,1\} \qquad i = 0,\ldots,n;\ j = 0,\ldots,n;\ k = 1,\ldots,K \tag{2.9}$$

$$y_{ik} \geq 0 \qquad i = 1,\ldots,n;\ k = 1,\ldots,K \tag{2.10}$$

No modelo descrito por (2.2)-(2.10), a função objetivo (2.2) busca minimizar os custos de atendimento. As restrições (2.3) garantem que cada cliente é visitado ao menos uma vez. A conservação de fluxo é garantida pelas restrições (2.4) enquanto as restrições (2.5) evitam a ocorrência de sub-rotas. As restrições (2.6) - (2.8) tratam da alocação das demandas dos clientes entre os veículos. Nestas, as restrições (2.6) garantem que o cliente i será atendido pelo veículo k, somente se k visita i. As restrições (2.7) asseguram que a demanda de todo cliente i será satisfeita e as restrições (2.8) garantem que a capacidade de cada veículo k não será excedida. As restrições (2.9) e (2.10) definem os domínios das variáveis de decisão.

2.2 Propriedades e Benefícios de Entregas Fracionárias

Dror et al. [29, 30] mostraram que naquelas instâncias do PRVEF em que a demanda média dos clientes é maior que 10% da capacidade do veículo, os ganhos ao adotar um esquema de atendimento que incorpora entregas fracionárias, quando comparado com uma abordagem que considera o PRV, são significativos. Além disso, observaram que os ganhos em termos de distância percorrida e número de veículos cresce conforme a porcentagem da demanda média dos clientes, em relação a capacidade do veículo, aumenta. Em um trabalho posterior, Archetti et al. [10] mostraram que as economias ao permitir tal es-

quema é de no máximo 50%. Tais análises, contudo, não oferecem uma ideia exata das relações existentes entre os ganhos obtidos ao permitir o fracionamento das demandas e as características dos clientes.

Archetti et al. [9] realizaram uma análise experimental mais acurada das implicações práticas ao adotar um esquema de entregas fracionárias enfatizando diferentes características para os clientes, estas relacionadas principalmente a localização geográfica e forma como as demandas estavam distribuídas. Os autores observaram que a redução máxima que pode ser obtida, tanto no número de veículos utilizados quanto nos custos, é de 50%, valor condizente com aquele do trabalho anterior. Além disso observaram que os maiores ganhos ocorrem quando a demanda média dos clientes está entre 50% e 75% da capacidade do veículo e a variância é pequena. Verificaram ainda que não existe dependência aparente entre a redução dos custos e a localização dos clientes.

Além da análise dos ganhos obtidos ao permitir entregas fracionárias, Dror et al. [29, 30] mostraram que as seguintes propriedades são observadas em soluções ótimas do PRVEF. Para cada uma dessas propriedades, seja o custo de viagem não negativo c_{ij} associado a cada aresta $\{i, j\} \in E$ do grafo empregado na definição do problema.

Propriedade 1: Se os custos c_{ij} atendem a desigualdade triangular, ou seja, $c_{ij} \leq c_{ik} + c_{kj}$, então duas rotas, em uma solução ótima para o PRVEF, não terão mais do que um cliente em comum [29, 30].

A definição a seguir será necessária para a Propriedade 2. Seja um conjunto C composto de k clientes, $C = \{c_1, c_2, \ldots, c_k\}$ e suponha que tais clientes sejam atendidos por k rotas, $r_1, r_2, \ldots, r_k$, $k \geq 2$, de tal forma que a rota 1 contenha os clientes c_1 e c_2, a rota 2 contenha os clientes c_2 e c_3, a rota $k-1$ contenha os clientes c_{k-1} e c_k e a rota k contenha os clientes c_k e c_1. Tal configuração é chamado um ciclo *k-split*.

Propriedade 2: Se os custos c_{ij} atendem a desigualdade triangular, então uma solução ótima para o PRVEF não conterá ciclos *k-split* (para qualquer valor de k) [29, 30].

Archetti et al. [12] mostraram que se as demandas dos clientes e a capacidade dos veículos forem valores inteiros, sempre existirá uma solução ótima para o PRVEF em que as frações das demandas em todas as rotas serão valores inteiros.

Capítulo 3

Revisão da Literatura

Neste capítulo serão apresentados os principais trabalhos da literatura que abordam o PRVEF, especificando para cada caso a variante tratada. Ele está organizado da seguinte forma. Inicialmente, serão listados os trabalhos que propuseram abordagens exatas para o problema. Em seguida, a literatura que faz uso de abordagens heurísticas será analisada. Por fim, outros trabalhos que de alguma forma exploram o PRVEF serão mostrados, estes incluem extensões do PRVEF, que são obtidas ao adicionar outras características ou restrições ao problema original, e aplicações em casos reais. Uma análise detalhada da literatura que trata do PRVEF pode ser encontrada no trabalho recente de Archetti e Speranza [11].

3.1 Abordagens Exatas

As abordagens exatas para o PRVEF são apresentadas a seguir. Dror et al. [28] desenvolveram uma formulação PIM para o PRVEF-FI e a partir desta derivaram novas classes de desigualdades válidas, que por sua vez foram incorporadas em um algoritmo de plano de cortes. Testes computacionais foram realizados em instâncias com até 20 clientes, sendo capaz de resolver na otimalidade apenas aquelas com até 10 clientes. Na mesma linha da abordagem anterior, Belenguer et al. [14] propuseram uma formulação para o PRVEF-FL e fizeram um estudo do poliedro associado, como resultado, várias classes de desigualdades válidas foram desenvolvidas com algumas delas definindo facetas. Um algoritmo de plano de cortes foi implementado e testado em instâncias com até 100 clientes. Os autores foram capazes de resolver os problemas-teste com até 50 clientes de forma ótima.

Jin et al. [43] propuseram um algoritmo exato de duas fases para o PRVEF-FL.

Na primeira fase, um problema de clusterização é resolvido, enquanto na segunda, uma abordagem baseada no Problema do Caixeiro Viajante (PCV) é usada para otimizar os custos de viagem para cada cluster. Experimentos computacionais foram conduzidos em instâncias com até 22 clientes. Num trabalho posterior, Jin et al. [44] desenvolveram um algoritmo de geração de colunas para o PRVEF-FL. Instâncias com até 100 clientes foram utilizadas para validar o algoritmo, sendo capaz de resolver na otimalidade apenas aquelas com até 20 clientes.

Moreno et al. [46] propuseram um algoritmo *Branch-and-Cut-and-Price* (BCP) e novas classes de desigualdades válidas a partir de uma formulação estendida para o PRVEF-FL. O algoritmo foi aplicado a instâncias com até 288 clientes. Uma abordagem similar foi proposta por Archetti et al. [5] que também desenvolveram um algoritmo BCP. Ambas as versões do problema, PRVEF-FL e PRVEF-FI, foram tratadas. Os autores empregaram um procedimento heurístico para obter soluções inteiras a partir das colunas geradas. Experimentos computacionais foram realizados em três conjuntos de instâncias da literatura, nos quais os problemas-testes possuem até 288 clientes.

Como mostrado em Archetti et al. [7], o PRVEF aparenta ser teoricamente mais tratável que o PRV. Isso porque o PRVEF pode ser resolvido em tempo polinomial quando o grafo subjacente possui propriedades especiais, enquanto o mesmo não acontece com o PRV. Contudo, apesar dos esforços dedicados ao desenvolvimento de abordagens exatas para tratar o PRVEF, tais métodos foram capazes de resolver de forma sistemática apenas instâncias com até 50 clientes. Este número é muito limitado quando comparado aos algoritmos exatos propostos para resolver o PRV, que conseguem resolver instâncias com até 135 clientes [33].

Mas até mesmo para o PRV, as abordagens exatas existentes ainda são muito limitadas, ficando restritas aos problemas-testes de tamanho reduzido. Assim, algoritmos heurísticos parecem ainda ser os mais adequados para tratar instâncias do PRVEF de tamanhos médios e grandes. Neste sentido, vários autores desenvolveram abordagens baseadas em heurísticas e meta-heurísticas. Os principais métodos encontrados na literatura são apresentados na seção seguinte.

3.2 Abordagens Heurísticas

Nos trabalhos em que Dror e Trudeau [29, 30] introduzem o PRVEF, os autores demonstraram empiricamente que economias significativas, tanto em termos do número de

veículos quanto nos custos de transporte, podem ser geradas ao permitir que as demandas dos clientes sejam divididas. Essas observações foram verificadas através de experimentos computacionais envolvendo um conjunto de 540 problemas-teste com até 150 clientes. Uma heurística de duas fases foi desenvolvida para resolver essas instâncias. Na primeira fase, soluções para o PRV são geradas por um procedimento baseado no algoritmo de Clarke e Wright [22], enquanto na segunda fase as opções de atendimento das demandas em mais de uma visita são avaliadas por dois novos procedimentos propostos especificamente para o PRVEF, um denominado *k-Split*, que testa as possibilidades de divisão da demanda de um dado cliente entre k rotas, e o outro denominado *RouteAddition*, que tenta juntar demandas fracionadas ao criar uma nova rota. Em ambas as fases foram aplicados procedimentos de melhora inter-rota e intrarrota baseados em trocas de clientes (*swap*) e remoção de arestas (2-*opt*).

Frizzell e Giffin [31] trataram o PRVEF-FI considerando que as coordenadas dos clientes e do depósito eram dadas em uma rede do tipo grade (*Grid Network*) e definiram limites inferiores e superiores no tamanho da porção de demanda a ser suprida para cada cliente. Desenvolveram uma heurística construtiva e uma heurística que atribui custos de divisão para cada cliente. Testaram o algoritmo em um conjunto de 1050 problemas-teste e observaram que as soluções geradas pelas heurísticas propostas foram capazes de reduzir a distância total e o número de veículos utilizados quando comparadas com as soluções geradas por um método construtivo guloso. Em um trabalho posterior, os mesmos autores desenvolveram um modelo matemático e outras heurísticas para tratar o PRVEF-FI considerando características adicionais, como janelas de tempo e tempo necessário para fazer uma entrega proporcional ao tamanho da carga. Os algoritmos propostos incluíam procedimentos de melhora baseados nos operadores de vizinhanças troca e realocação, considerados clássicos para o PRV. Testes computacionais foram realizados em instâncias clássicas da literatura para o Problema de Roteamento de Veículos com Janelas de Tempo (PRVJT) e em um conjunto de 6040 problemas-teste com até 200 clientes [32].

Archetti et al. [12] propuseram uma heurística de três fases baseada em Busca Tabu (BT) [35] para o PRVEF-FI. Na primeira fase, uma solução é construída usando a estratégia *Route First - Cluster Second*. O algoritmo GENIUS [34] é empregado para gerar as rotas. Na segunda, um procedimento de reinserção é aplicado para todos os clientes. Neste, uma solução vizinha é obtida ao remover um dado cliente de uma ou mais rotas em que ele é atendido para em seguida ser reinserido em outra rota. As cargas desse cliente a serem realocadas dependem da capacidade residual da rota em que ele será inserido. As inserções são feitas usando o critério de inserção mais barata. Na última fase, o algoritmo

GENIUS é aplicado novamente para tentar melhorar ainda mais a solução. Além disso, verificações de algumas das propriedades estruturais presentes em soluções ótimas são aplicadas para ver se elas estão presentes na solução encontrada. Experimentos computacionais foram realizados em um conjunto de instâncias, composto de 49 problemas-testes contendo até 199 clientes, gerado pelos autores a partir de instâncias da literatura para o PRV. Archetti et al. [13] desenvolveram uma abordagem híbrida que combina a Busca Tabu descrita anteriormente e uma formulação PIM. Nesta, as soluções mais promissoras geradas durante a execução da heurística são armazenadas para em seguida serem combinadas pela formulação na tentativa de encontrar soluções ainda melhores. O algoritmo foi testado num conjunto de instâncias similar ao anterior.

Chen et al. [21] desenvolveram um algoritmo híbrido para o PRVEF-FI que combina um procedimento construtivo baseado na heurística das economias de Clarke e Wright [22] com uma formulação PIM. Neste, a partir de uma solução inicial, a formulação considera os clientes que estão em uma ou em ambas as extremidades de cada uma das rotas e um dado número dos clientes mais próximos a estes e, após tentar realocar os clientes selecionados em cada rota, testa as seguintes possibilidades: (i) mm dado cliente de uma das extremidades é removido de todas as rotas em que aparece e movido, junto com sua demanda, para outra ou outras rotas, e (ii) uma fração da demanda de um dado cliente posicionado em uma das extremidades é movida para uma ou mais rotas diferentes juntamente com uma cópia do cliente. Em um estágio seguinte, um procedimento baseado na heurística *Record-to-Record* (RTR) [20] é aplicado para tentar melhorar a solução. Experimentos computacionais foram realizados em dois conjuntos de instâncias, sendo o primeiro gerado conforme [12] e o outro, gerado pelos autores, composto de 21 problemas-teste com até 288 clientes.

Campos et al. [19] propuseram uma heurística populacional baseada na meta-heurística *Scatter Search* (SS) [36] para o PRVEF-FL. A população inicial é gerada por dois procedimentos diferentes. O algoritmo foi testado em um conjunto de instâncias gerado conforme [12]. O algoritmo que Boudia et al. [17] desenvolveram para tratar o PRVEF-FI também mantém uma população de soluções durante sua execução. Este é baseado em Algoritmo Genéticos (AG) [42] incorporando procedimentos de busca local. Os autores também propuseram um procedimento de gerência de população para controlar a diversidade das soluções armazenadas. Na fase de busca local, além das estruturas de vizinhanças clássicas para o PRV baseadas em trocas e realocações de clientes, também adicionaram três novas estruturas específicas para o PRVEF, sendo uma delas a generalização do *k-Split*. Os conjuntos de instâncias de Belenguer et al. [14] e Campos et al. [19] foram utilizados

para avaliar o método.

Aleman et al. [3, 1] apresentaram um procedimento construtivo para o PRVEF-FL que emprega um novo conceito de medida de controle de ângulo (*route angle control measure*) para alocação dos clientes nas rotas sendo geradas. A solução construída é, então, melhorada utilizando o procedimento VND que, além das estruturas de vizinhanças clássicas para o PRV, emprega vizinhanças específicas para o PRVEF. A partir dessas ideias, Aleman e Hill [2] implementaram uma heurística BT com construção de vocabulário para o PRVEF-FI. Ambos os algoritmos foram testados nos conjuntos de instâncias de Belenguer et al. [14], Campos et al. [19] e Chen et al. [21]. Derigs et al. [25] apresentaram um estudo que compara o comportamento de várias meta-heurísticas para o PRVEF-FI. Os conjuntos de instâncias de Campos et al. [19] e Chen et al. [21] foram utilizados nas avaliações. Wilck IV e Cavalier [62] desenvolveram um procedimento construtivo de duas fases para o PRVEF-FL. Em um estudo posterior, os mesmos autores incorporaram este procedimento em um Algoritmo Genético para tratar a mesma variante [63].

A partir das duas versões mais simples do PRVEF que tratam do dimensionamento da frota, PRVEF-FI e PRVEF-FL, diversas variantes do problema foram obtidas ao incorporar outras características ou restrições. Alguns desses trabalhos serão listados na seção seguinte. Problemas encontrados em situações reais que foram modelados como PRVEF também serão apresentados.

3.3 Outros Trabalhos Relacionados ao PRVEF

Iniciando pelas aplicações do PRVEF em situações práticas, Sierksma e Tijssen [53] se basearam no PRVEF para modelar e resolver o problema de agendamento dos voos de helicópteros utilizados no transporte de funcionários para plataformas em alto-mar, localizadas na plataforma continental holandesa, no Mar do Norte. Song et al. [56] também utilizaram o PRVEF como base para tratar um problema de entrega de jornais na cidade de Seul, Coreia do Sul, no qual os principais objetivos eram determinar de forma ótima a atribuição de agentes, o agendamento e os roteiros de entregas. Ambrosino e Sciomachen [4] estudaram um problema de distribuição de alimentos em uma companhia italiana e também empregaram uma abordagem que permite divisão das demandas. Belfiore e Yoshizaki [15] apresentaram um estudo de caso realizado em uma empresa de distribuição do Brasil que, além de divisão nas demandas, considera também restrições de janela de tempo e frota heterogênea. Em todos esses trabalhos foram empregadas abordagens

heurísticas para resolução dos problemas.

Os principais trabalhos que tratam de algumas variantes do PRVEF estão listados a seguir. A versão com janelas de tempo foi examinada por Frizzel e Giffin [32], Ho e Haugland [41], Desaulniers [26], Archetti et al. [6] e Salani e Vacca [52]. O PRVEF com coleta e entrega foi estudada por Nowak et al. [47, 48]. Tavakkoli-Moghaddam et al. [60] generalizaram o PRVEF ao considerar um frota heterogênea de veículos. Gulczynski et al. [37] trataram o PRVEF com quantidades mínimas a serem entregues. Em um trabalho posterior [38], os mesmos autores, estudaram a versão com múltiplos depósitos.

Capítulo 4

Metodologia

O método proposto para resolver o PRVEF consiste em um algoritmo *multi-start* baseado na meta-heurística *Iterated Local Search* (ILS) [45] e no procedimento *Variable Neighborhood Descent* (VND) [40], cuja a ordem em que as estruturas de vizinhança são aplicadas durante a fase de busca local é selecionada aleatoriamente [58]. Contudo, antes de descrever a metodologia empregada, algumas definições referentes a heurísticas e meta-heurísticas serão apresentadas.

Segundo Silver et al. [55], heurísticas são procedimentos dedicados a resolução de problemas cuja estrutura pode ser interpretada e explorada de forma inteligente, permitindo assim que uma solução razoável seja obtida por meio de uma abordagem intuitiva. Dentre as categorias de heurísticas descritas por esses autores, as duas mais comumente encontradas em trabalhos que abordam o PRV e suas variantes são as de Construção e Busca Local. A ideia básica de uma heurística construtiva é incrementalmente adicionar componentes individuais a uma solução parcial até obter uma solução completa viável. Diferente das heurísticas construtivas, as heurísticas de busca local tentam melhorar uma solução viável ao manipular os elementos que a compõem. Devido a estas características é comum a utilização destes dois tipos de heurística de forma combinada [64]. As meta-heurísticas baseiam-se nestas ideias de combinar heurísticas.

Meta-heurísticas são estratégias de alto nível que, por meio de heurísticas subordinadas, tentam explorar o espaço de busca eficientemente, visando encontrar soluções de boa qualidade que estejam próximas do ótimo ou sejam ótimas. Além das funções desempenhadas durante o processo de busca, as heurísticas subordinadas servem também para modelar o conhecimento específico referente a cada problema. Devido ao alto grau de abstração empregado em suas descrições, tais métodos podem ser aplicados em várias

situações, não se limitando a um problema específico. Diferente das heurísticas clássicas, as meta-heurísticas incorporam mecanismos para tentar evitar ficarem presas em ótimos locais de baixa qualidade. As estratégias mais avançadas fazem uso do conhecimento obtido durante o processo de busca (através de algum tipo de memória) para tentar explorar mais eficientemente o espaço de soluções. Alguns são não determinísticos, variando desde simples procedimentos de busca local até complexos processos de aprendizagem. As meta-heurísticas podem ser classificadas de varias maneiras, tudo depende da caraterística que se deseja dar maior enfoque. A classificação que utiliza como critério descritor o número de soluções empregadas ao mesmo tempo, divide as estratégias existentes em *métodos de trajetória* (uma única solução é considerada) e *métodos populacionais* (várias soluções são consideradas) [16].

A meta-heurística *Iterated Local Search* (ILS) [45] é um método de trajetória que explora o espaço de soluções através da aplicação sucessiva de perturbações a ótimos locais. Estes ótimos locais são obtidos a partir de um algoritmo de busca local, que por sua vez, tem como ponto de partida uma solução inicial proveniente de um procedimento construtivo ou uma solução perturbada. Desta definição observa-se que o ILS pode ser visto como um algoritmo de busca local cujo o espaço de soluções é composto apenas de ótimos locais.

Semelhante aos algoritmos de busca local clássicos, o ILS parte de uma solução inicial, na qual um procedimento de busca será aplicado até que um ótimo local seja obtido. Este ótimo local será a solução de partida do laço principal do ILS. A cada iteração deste, a solução de partida é perturbada, em seguida é refinada através do procedimento de busca, e por fim este novo ótimo local é avaliado segundo um critério de aceitação que dirá se ele será utilizado como nova solução de partida na iteração seguinte ou se será descartado. O pseudocódigo de uma versão básica do ILS é mostrado no Algoritmo 1.

Algoritmo 1: ILS

```
1 Procedimento ILS
2 s_0 ← GeraSolucaoInicial();
3 s* ← BuscaLocal(s_0);
4 enquanto critério de parada não satisfeito faça
5     s' ← Perturba(s*, historico);
6     s'* ← BuscaLocal(s');
7     s* ← CriterioAceitacao(s*, s'*, historico);
8 fim enquanto
9 fim ILS
```

Quatro componentes básicos devem ser especificados ao desenvolver algoritmos se-

gundo a estratégia ILS. O primeiro deles, representado no Algoritmo 1 pelo procedimento `GeraSolucaoInicial()` (linha 2), é responsável pela construção de soluções iniciais. É interessante que este seja rápido e as soluções geradas sejam bons pontos de partida para a busca local. O segundo elemento, identificado pelo procedimento `BuscaLocal()` (linhas 3 e 6), é responsável por explorar o espaço de busca e encontrar bons ótimos locais. O terceiro elemento, caracterizado pelo procedimento `Perturba()` (linha 5), deve ser capaz de gerar soluções modificadas de tal forma que o novo ótimo local obtido após a busca local, seja diferente do usado como ponto de partida. O último elemento, identificado pelo procedimento `CriterioAceitacao()` (linha 7), é responsável por determinar se a solução proveniente da busca local será descartada ou utilizada como ponto de partida em uma próxima iteração. Assim, heurísticas baseadas em ILS permitem um algo grau de flexibilidade na escolha de cada um desses quatro componentes.

A forma como cada um desses componentes foi implementado para tratar o PRVEF está descrita nas seções seguintes. Além desses, outros elementos necessários à compreensão do método proposto também são apresentados. A Seção 4.1 mostra como as soluções para o PRVEF foram modeladas e a função de avaliação empregada. A Seção 4.2 descreve procedimentos auxiliares empregados durante as diversas fases do algoritmo. Os algoritmos construtivos são apresentados na Seção 4.3. A busca local e as vizinhanças utilizadas são mostradas na Seção 4.4. A Seção 4.5 descreve os mecanismos de perturbação. O critério de aceitação e o pseudocódigo do algoritmo incorporando todos esses componentes são apresentados na Seção 4.6.

4.1 Representação e Avaliação de Soluções

Na abordagem proposta, uma solução s para o PRVEF é composta basicamente por duas tabelas: R (com K linhas e um número variável de colunas) e Z (com K linhas e n colunas), respectivamente. Cada linha r da tabela R representará uma rota, a qual terá um veículo associado. As colunas de cada uma dessas rotas indicarão uma permutação de clientes, $R = \{r_1 = [r_{1_0}, r_{1_2}, \ldots, r_{1_{\bar{n_1}}}], r_2 = [r_{2_0}, r_{2_2}, \ldots, r_{2_{\bar{n_2}}}], \ldots, r_K = [r_{K_0}, r_{K_2}, \ldots, r_{K_{\bar{n_K}}}]\}$, sendo que as primeiras e últimas colunas conterão o depósito. Na tabela Z, cada linha r estará associada a uma linha da tabela R. Cada uma das n colunas que compõem as linhas da tabela Z representará um cliente, sendo que Z_{rj} armazenará a fração da demanda do cliente j que é atendida pela rota r, isto se o cliente j é visitado pela rota r em R, caso contrário, $Z_{rj} = 0$. A Figura 4.1 ilustra graficamente uma solução s para o PRVEF, na qual o cliente 7 é visitado pelas rotas r_1 e r_2, e sua respectiva re-

presentação utilizando as tabelas R e Z. Com esta representação, a informação acerca da quantidade de demanda suprida por cada rota, para um determinado cliente, é acessada rapidamente.

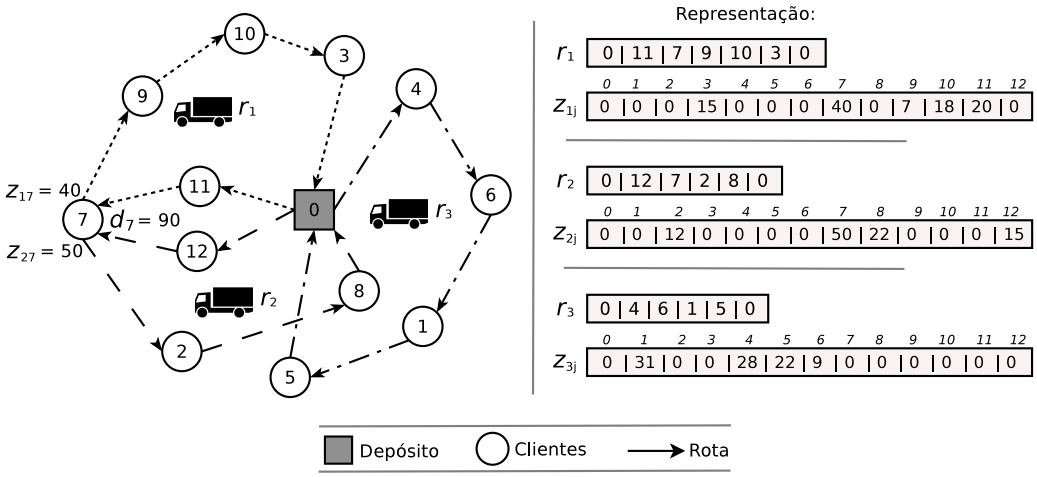

Figura 4.1: Representação de uma solução.

A validade de uma solução s para o PRVEF é facilmente verificada por meio do modelo F_{Arch} descrito no Capítulo 2, uma vez que as tabelas $R \in s$ e $Z \in s$, podem ser transformadas nas variáveis x_{ij}^k e y_{ik}, respectivamente. Baseando-se na função objetivo desse modelo, nas informações armazenadas na tabela R e nos custos de viagem c_{ij}, associados a cada aresta $\{i, j\} \in E$, o custo $f(s)$ da solução s é calculado através da Equação (4.1).

$$f(s) = \sum_{r \in R} \sum_{i=1}^{|r|-1} c_{r_i r_{(i+1)}} \tag{4.1}$$

Como pode-se observar, o custo de uma dada solução é calculado com base apenas na distância percorrida, sem termos adicionais. Porém, isto não é sempre possível, especialmente naqueles problemas onde muitos requisitos devem ser atendidos e os recursos são escassos. Em tais situações, encontrar soluções viáveis pode ser uma tarefa muito difícil, como no caso do PRVEF-FL por exemplo, no qual deve-se determinar um conjunto de rotas, para uma frota limitada, que atenda todos os clientes sem que a capacidade dos veículos seja excedida. Uma das abordagens utilizadas para tentar contornar essas dificuldades é relaxar as restrições mais complicadas do problema e penalizar as violações destas no cálculo do custo das soluções. Nos algoritmos baseados em busca local, isso permite ir além do espaço de soluções viáveis e explorar também o espaço de soluções inviáveis. Esta foi uma das técnicas empregadas durante o desenvolvimento do algoritmo

para tratar o PRVEF, contudo, não foi possível obter um desempenho satisfatório. Por outro lado, quando da utilização de tal abordagem, o algoritmo proposto ainda estava em um estágio inicial de desenvolvimento, logo, um estudo mais acurado se faz necessário.

4.2 Procedimentos Auxiliares

Nesta seção serão descritos três procedimentos auxiliares importantes que são usados em diversas partes do algoritmo proposto. O primeiro é denominado `SplitReinsertion` e é baseado em uma generalização do procedimento *k-Split* de Dror et al. [29]. Essa generalização foi desenvolvida por Boudia et al. [17]. O Algoritmo 2 apresenta o pseudocódigo do procedimento `SplitReinsertion`. Os dados de entrada para este procedimento são: (i) uma solução s; (ii) um cliente i; (iii) a demanda $d'_i \leq d_i$; e (iv) uma dada rota proibida $\bar{r}$. Este último dado será útil quando for necessário esvaziar uma rota ou quando o cliente i é atendido por um único veículo. Inicialmente, cada rota $r \neq \bar{r}$, cuja capacidade residual a_r é maior que zero, é adicionada a lista L (linhas 5 a 12). Se a soma das capacidades residuais não for suficiente para atender toda a demanda de i o procedimento é encerrado. Caso contrário, para cada rota $r \in L$, o menor custo u_r para inserir o cliente i na rota r é calculado (linhas 14 a 17). O problema relacionado a escolha da combinação que minimiza a soma dos custos de inserção de forma que a demanda de i seja completamente atendida pode ser visto como o Problema da Mochila (*knapsack problem*). Embora existam algoritmos baseados em programação dinâmica muito eficientes para resolver este problema, por questões de tempo computacional decidiu-se aplicar uma simples heurística gulosa, no qual as inserções são feitas em ordem crescente de u_r/a_r (linha 18). Boudia et al. [17] apontaram que o uso de programação dinâmica aumenta o tempo de execução em 30% sem haver melhoras significativas na solução encontrada por esta heurística gulosa.

O segundo procedimento auxiliar, denominado `EmptyRoutes`, é utilizado para reduzir o número de rotas de uma solução. O Algoritmo 3 mostra o pseudocódigo do `EmptyRoutes`. Visto que o procedimento é simples, sua explicação será omitida.

O terceiro e último procedimento auxiliar, denominado `ReinsertSingleCustomer`, também pode ser visto como um método de redução de rotas, porém apenas rotas unitárias, cuja remoção do cliente existente e reinserção deste em outras rotas gere uma solução de custo menor, são consideradas. O pseudocódigo do `ReinsertSingleCustomer` é mostrado no Algoritmo 4. Neste, para cada rota $r \in s$ (linhas 4 a 14) composta de um único cliente (linha 5), o procedimento `SplitReinsertion` é aplicado com a rota r

Algoritmo 2: SplitReinsertion

```
1  Procedimento SplitReinsertion(s, i, d'_i, r̄)
2  L ← A ← U ← ∅;
3  Seja K_s o número de veículos em s;
4  sumResidual ← 0;
5  para r ← 1, ..., K_s faça
6      Seja a_r a capacidade residual da rota r;
7      se r ≠ r̄ e a_r > 0 então
8          L ← L ∪ {r};
9          A ← A ∪ {a_r};
10         sumResidual ← sumResidual + a_r;
11     fim se
12 fim para
13 se sumResidual ≥ d'_i então
14     para r ← 1, ..., |L| faça
15         Seja u_r o menor custo adicional para inserir o cliente i em r;
16         U ← U ∪ {u_r};
17     fim para
18     s ← knapGreedyHeuristic(s, i, d'_i, L, A, U);
19 fim se
20 retorne s;
21 fim
```

Algoritmo 3: EmptyRoutes

```
1  Procedimento EmptyRoutes(s)
2  enquanto K > K_min faça
3      r̄ ←rota com menor carga em s;
4      enquanto r̄ ≠ ∅ faça
5          i ← primeiro cliente de r̄;
6          s ← SplitReinsertion(s, i, d'_i, r̄);
7          r̄ ← r̄ − {i};
8      fim enquanto
9  fim enquanto
10 retorne s;
11 fim
```

definida como proibida (linha 7). Se o custo da solução encontrada for menor que o custo da solução corrente, esta nova solução é armazenada (linhas 8 a 12). Enquanto for possível encontrar soluções de melhor qualidade o algoritmo é reiniciado (linhas 15 a 18). Este procedimento é utilizado apenas durante a construção.

Algoritmo 4: ReinsertSingleCustomer

```
1  Procedimento ReinsertSingleCustomer(s)
2  aplicarSplit ← falso;
3  f'' ← f(s);
4  para r ← 1, ..., |s| faça
5      se |r| = 1 então
6          i ← único cliente em r;
7          s' ← SplitReinsertion(s, i, d_i, r);
8          se f(s') < f'' então
9              s'' ← s';
10             f'' ← f(s');
11             aplicarSplit ← verdadeiro;
12         fim se
13     fim se
14 fim para
15 se aplicarSplit = verdadeiro então
16     s ← s'';
17     Vá para a linha 2;
18 fim se
19 retorne s;
20 fim
```

4.3 Métodos Construtivos

O procedimento de construção adotado neste trabalho para a geração de soluções iniciais é baseado em heurísticas de inserção. Nestas, uma solução completa é obtida de forma incremental ao inserir clientes não roteados a uma solução parcial. O cliente a ser inserido e a rota de destino são decididos por meio de dois critérios de inserção, a inserção viável mais próxima (CIMPV) e a inserção viável mais barata modificada (CIMBVM), e duas estratégias de inserção, a inserção sequencial (EISS) e a inserção paralela (EIP), em ambos os casos apenas um selecionado. Uma explicação detalhada de cada uma desses elementos pode ser encontrada em Penna et al. [50].

O funcionamento do procedimento construtivo é descrito a seguir. Inicialmente, são geradas K_{min} rotas vazias (linha 3) e em cada uma dessas rotas é inserido um cliente selecionado aleatoriamente (linhas 5 a 8) da lista LC que contém os clientes ainda não roteados (linha 4). É importante ressaltar que cada rota deve conter um cliente diferente. No passo seguinte, a estratégia de inserção e o critério de inserção são selecionados também aleatoriamente (linhas 9 e 10). Os clientes ainda não roteados são então iterativamente inseridos a essa solução parcial (linhas 11 a 15). Até este ponto, o fracionamento das demandas não é permitido. Se a solução gerada ao final do processo de inserção

contém mais de K_{min} veículos, os procedimentos `ReinsertSingleCustomer` (linha 16) e `EmptyRoutes` (linhas 17 a 19) são chamados para que a solução final tenha exatamente K_{min} veículos. O pseudocódigo do procedimento construtivo é detalhado no Algoritmo 5.

Algoritmo 5: GSolInicial

```
1  Procedimento GSolInicial()
2  K ← K_min;                                   /* Número mínimo de veículos */
3  s ← {r^1, ..., r^K};      /* Solução s é inicializada com K rotas vazias */
4  LC ← {1, ..., n};
5  para r ← 1, ..., K faça
6      s[r] ← i ∈ LC selecionado aleatoriamente;
7      LC ← LC − {i};
8  fim para
9  EstratégiaInserção ← EISS ou EIP, selecionada aleatoriamente;
10 CritérioInserção ← CIMPV ou CIMBVM, selecionado aleatoriamente;
11 se EstratégiaInserção = EISS então
12     s ← InserçãoSemiSequencial(s, LC, CritérioInserção, gammaInterval);
13 senão
14     s ← InserçãoParalela(s, LC, CritérioInserção, gammaInterval);
15 fim se
16 s ← ReinsertSingleCustomer(s);
17 se |s| > K então
18     s ← EmptyRoutes(s);
19 fim se
20 retorne s;
21 fim
```

4.3.1 Critérios de Inserção

No critério CIMPV, o custo de inserção, denotado pela função $g(i)$ na Equação 4.2, é dado pela distância entre um cliente $i \in LC$ e os clientes já inseridos na solução parcial. Assume-se que o cliente i é inserido após o cliente j.

$$g(i) = c^r_{ji} \tag{4.2}$$

No critério CIMBVM, o custo de inserção de um dado cliente ainda não roteado $i \in LC$ em uma das rotas da solução parcial, denotado pela função $g(i)$ na Equação 4.3, é computado a partir da soma de dois termos. O primeiro calcula o custo de inserção do cliente ainda não roteado i entre os pares de clientes não adjacentes j e k de cada rota da solução parcial. O segundo termo é uma penalização cujo proposito é desestimular a inserção tardia de clientes distantes do depósito. A influência do custo da viagem de ida

e volta ao depósito no segundo termo é ponderada pelo fator γ.

$$g(i) = (c^r_{ji} + c^r_{ik} - c^r_{jk}) - \gamma(c^r_{0i} + c^r_{i0}) \tag{4.3}$$

Em ambos os critérios, o cliente a ser inserido na solução parcial é aquele com o menor custo associado, isto é, $i_{min} \leftarrow arg\ min\{g(i)|i \in LC\}$.

4.3.2 Estratégias de Inserção

Duas estratégias de inserção são empregadas para decidir quais rotas existentes na solução parcial serão consideradas ao rotear novos clientes. Essas estratégias são: Inserção Sequencial (EISS) e Inserção Paralela (EIP). A diferença principal entre elas está no número de rotas consideradas a cada inserção. Em ambas as estratégias, quando o critério CIM-BVM é utilizado, o valor de γ é selecionado aleatoriamente do intervalo $gammaInterval = \{0.1, 0.2, 0.3, \ldots, 1.0\}$ (linhas 2 a 4 dos Algoritmos 6 e 7), definido por meio de experimentos empíricos.

Algoritmo 6: InserçãoSemiSequencial

```
1  Procedimento InserçãoSemiSequencial(s, LC, CritérioInserção, gammaInterval)
2  se CritérioInserção = CIMBVM então
3      γ ← valor ∈ gammaInterval, selecionado aleatoriamente;
4  fim se
5  r_0 ← 1;
6  enquanto LC ≠ ∅ faça
7      clienteInserido ← falso;
8      para r ← r_0, ..., |s| faça
9          se LC ≠ ∅ e ∃i ∈ LC que pode ser inserido na rota r então
10             Calcule o custo g(i), ∀i ∈ LC, conforme CritérioInserção;
11             i' ← arg min_(i ∈ LC){g(i)};
12             s[r] ← s[r] ∪ {i'};
13             LC ← LC − {i'};
14             clienteInserido ← verdadeiro;
15         fim se
16     fim para
17     se |LC| > 0 e clienteInserido = falso então
18         Seja r' uma rota vazia;
19         s ← s ∪ {r'};
20         r_0 ← r';
21     fim se
22 fim enquanto
23 retorne s;
24 fim
```

A EISS opera em uma única rota por vez. O pseudocódigo do procedimento é descrito no Algoritmo 6. Após definir o valor de γ, caso o critério CIMBVM seja empregado (linhas 2 a 4), o laço principal do algoritmo (linhas 6 a 22) é executado enquanto a lista LC não estiver vazia. Cada uma das rotas são então preenchidas conforme o critério de inserção selecionado (linhas 8 a 16). Quando não for mais possível executar uma inserção viável devido a falta de capacidade residual nas rotas existentes, um veículo extra é adicionado (linhas 17 a 21) e o procedimento continua até que todos os clientes tenham sido atendidos.

A EIP opera de forma similar ao procedimento anterior, a diferença é que todas as rotas são consideradas ao inserir um novo cliente. O pseudocódigo do procedimento é detalhado no Algoritmo 7. Após definir o valor de γ, caso o critério CIMBVM seja empregado (linhas 2 a 4), o laço principal do algoritmo (linhas 6 a 20) também é executado enquanto a lista LC não estiver vazia. Cada cliente é então adicionado à rota r com menor custo de inserção, calculado de acordo com o critério de inserção selecionado (linhas 8 a 16). Como no caso anterior, quando não for mais possível executar uma inserção viável devido a falta de capacidade residual nas rotas existentes, um veículo extra é adicionado (linhas 17 a 21) e o algoritmo continua até que todos os clientes tenham sido atendidos.

Algoritmo 7: InserçãoParalela

```
1  Procedimento InserçãoParalela(s, LC, CritérioInserção, gammaInterval)
2  se CritérioInserção = CIMBVM então
3      γ ← valor ∈ gammaInterval, selecionado aleatoriamente;
4  fim se
5  r_0 ← 1;
6  enquanto LC ≠ ∅ faça
7      clienteInserido ← falso;
8      se LC ≠ ∅ e ∃i ∈ LC que pode ser inserido em s então
9          Calcule o custo g(i, r), ∀i ∈ LC e ∀r ∈ {r_0, ..., |s|}, conforme CritérioInserção;
10         (i', r') ← arg min_{(i ∈ LC, r∈{r_0,...,|s|})} {g(i, r)};
11         s[r'] ← s[r'] ∪ {i'};
12         LC ← LC − {i'};
13         clienteInserido ← verdadeiro;
14     fim se
15     se |LC| > 0 e clienteInserido = falso então
16         Seja r'' uma rota vazia;
17         s ← s ∪ {r''};
18         r_0 ← r'';
19     fim se
20 fim enquanto
21 retorne s;
22 fim
```

4.4 Busca Local

Conforme mencionado anteriormente, a busca local é realizada por um procedimento baseado no RVND [58]. Uma das vantagens desta abordagem randomizada é que não é necessário especificar uma ordem para as vizinhanças, evitando assim mais um parâmetro a ser calibrado. Penna et al. [50], empregando uma abordagem baseada no ILS, mostraram por meio de testes empíricos que o RVND é capaz de encontrar, em média, resultados de melhor qualidade, quando comparado com o VND.

O conjunto $N = \{N^1, N^2, N^3, \ldots, N^t\}$, composto por t estruturas de vizinhança inter-rotas será adotado na descrição a seguir. O pseudocódigo do RVND é mostrado no Algoritmo 8. Inicialmente, a lista de vizinhanças (NL) é preenchida com as estruturas inter-rotas (linha 2). Em seguida, a cada execução do laço principal (linhas 4 a 16), uma vizinhança $N^{(h)} \in NL$ é selecionada aleatoriamente e o melhor vizinho é determinado (linhas 5 e 6). Em caso de melhora, o algoritmo checa se as rotas modificadas possuem clientes que são visitados mais de uma vez pelo mesmo veículo. Se isto ocorrer, um procedimento de reparo, que elimina as múltiplas cópias de clientes repetidos que geram a maior redução de custos, é chamado (linha 9). Neste caso, a demanda parcial atendida pela cópia removida é transferida para a cópia que permanece na rota. Por fim, uma busca local intra-rota é executada nas rotas que foram modificadas e NL é reinicializada (linhas 7 a 12). Se uma dada vizinhança h falha ao obter um solução de melhor qualidade, $N^{(h)}$ é removida de NL (linha 13). O procedimento termina quando NL se torna vazia.

4.4.1 Vizinhanças Inter-rotas

Um conjunto N composto de dez estruturas de vizinhança inter-rotas foi empregado. Destas, as cinco primeiras estruturas são baseadas no esquema λ-*interchanges* proposto por Osman [49], que consiste em permutar λ clientes entre duas rotas diferentes. Neste trabalho foi utilizado $\lambda = 2$ para limitar o número de possibilidades. A sexta estrutura é baseada no operador *Cross-exchange* de Taillard et al. [59] e consiste em permutar dois segmentos de rotas diferentes. As duas estruturas seguintes foram propostas por Boudia et al. [17] e estendem as vizinhanças *Swap(1,1)* e *Swap(2,1)* para tratar movimentos fracionários. A nona estrutura é baseada no procedimento *RouteAddition*, específico para o PVREF, proposto por Dror et al. [29]. A última estrutura é uma generalização do procedimento *k-Split* proposto por Dror et al. [29], também é específica para o PRVEF e foi proposta por Boudia et al. [17].

Algoritmo 8: RVND

```
1  Procedimento RVND(s)
2  Inicialize a lista de vizinhanças NL;
3  Atualize as estruturas de dados auxiliares;
4  enquanto NL ≠ ∅ faça
5      Escolha uma vizinhança N^(h) ∈ NL aleatoriamente;
6      Encontre o melhor vizinho s' de s ∈ N^(h);
7      se f(s') < f(s) então
8          s ← s';
9          s ← CheckAndRepair(s);
10         s ← IntraRouteRVND(s);
11         Atualize NL;
12     senão
13         Remova N^(h) de NL;
14     fim se
15     Atualize as estruturas de dados auxiliares;
16 fim enquanto
17 retorne s;
18 fim
```

Assim, as seis primeiras estruturas de vizinhanças são consideradas clássicas na literatura de PRV e as últimas quatro foram desenvolvidas especificamente para o PRVEF. Cada uma destas estruturas estão descritas a seguir.

- ***Shift(1,0)*** – $N^{(1)}$ – Consiste na remoção do cliente i da rota r_1 e subsequente reinserção na rota r_2. Na Figura 4.2, o cliente 8 é removido da rota 2 e reinserido entre o cliente 5 e o depósito na rota 3.

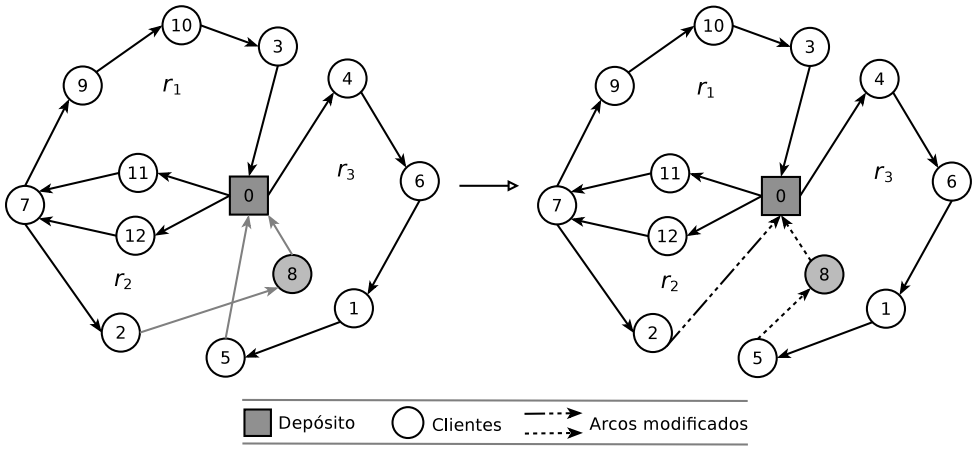

Figura 4.2: Vizinhança inter-rotas *Shift(1,0)*.

- ***Swap(1,1)*** – $N^{(2)}$ – O cliente i da rota r_1 é trocado com o cliente j da rota r_2.

No exemplo mostrado na Figura 4.3, o cliente 8 da rota 2 é trocado com o cliente 5 da rota 3.

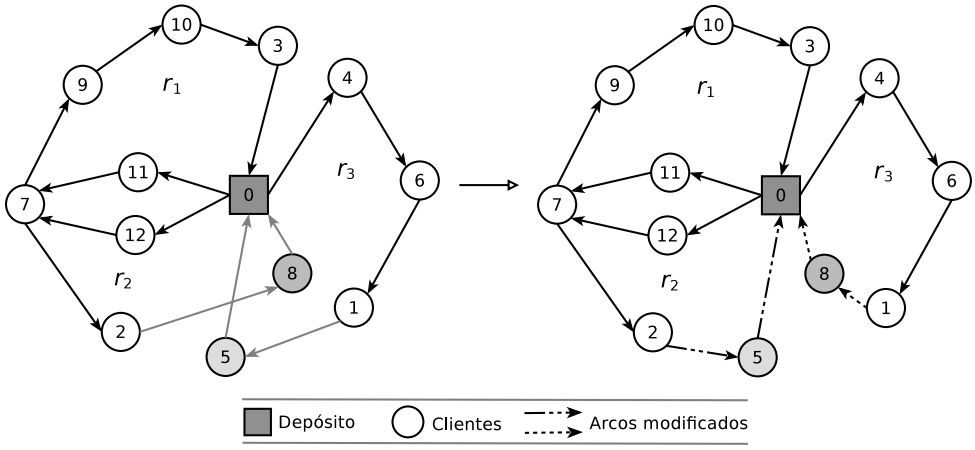

Figura 4.3: Vizinhança inter-rotas *Swap(1,1)*.

- ***Shift(2,0)*** $- N^{(3)}$ – Dois cliente adjacentes i e j (ou o arco (i, j)) são removidos da rota r_1 e logo em seguida reinseridos na rota r_2. A transferência do arco invertido (j, i) também foi considerada. A Figura 4.4 ilustra o movimento no qual os clientes 1 e 5 são removidos da rota 3 e reinseridos entre os clientes 2 e 8 na rota 2.

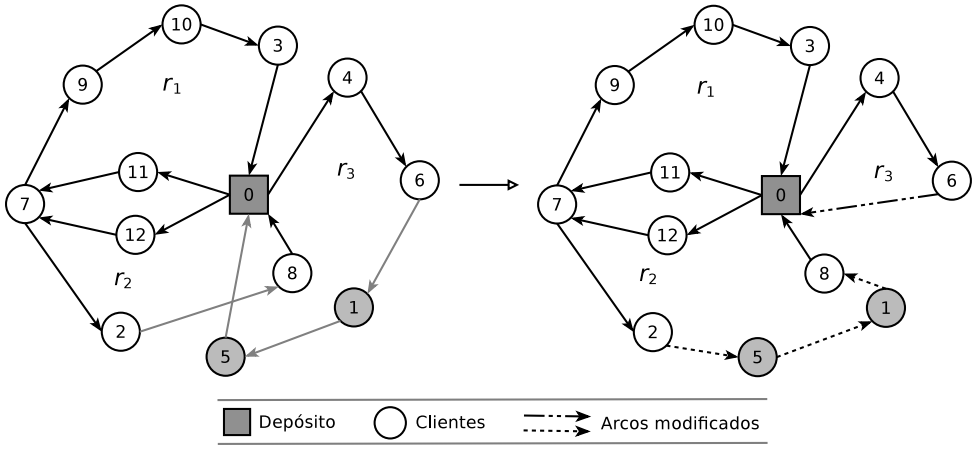

Figura 4.4: Vizinhança inter-rotas *Shift(2,0)*.

- ***Swap(2,1)*** $- N^{(4)}$ – Consiste na troca de dois clientes adjacentes i e j (ou o arco (i, j)) da rota r_1 com um dado cliente k pertencente a rota r_2. Como no caso do *Shift(2,0)*, a troca entre o arco inverso (j, i) e o cliente k também foi considerada. Na Figura 4.5 os clientes 2 e 8 da rota 2 são trocados com o cliente 5 da rota 3.

- ***Swap(2,2)*** $- N^{(5)}$ – Dois clientes adjacentes i e j (ou o arco (i, j)) da rota r_1 são trocados com dois clientes adjacentes k e l (ou o arco (k, l)) da rota r_2. As trocas

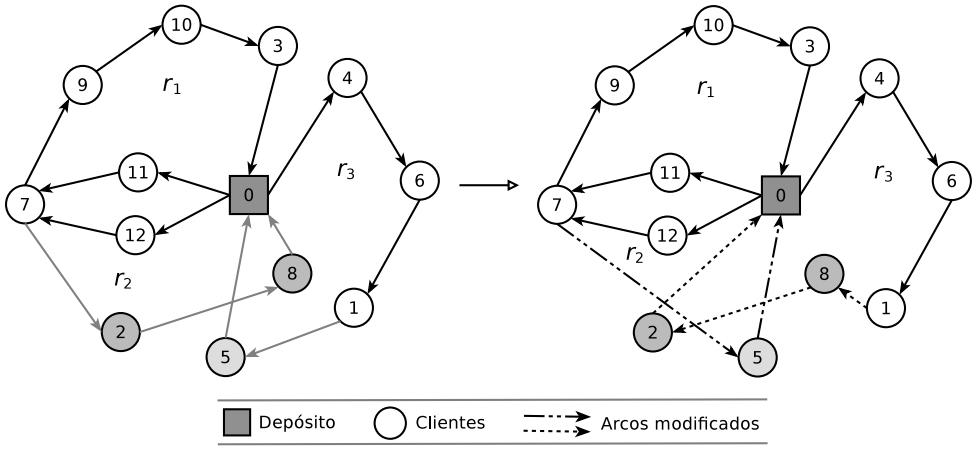

Figura 4.5: Vizinhança inter-rotas *Swap(2,1)*.

entre os arcos opostos (j, i) e (l, k) também foram consideradas, isso resultou em um total de quatro combinações possíveis. No exemplo mostrado na Figura 4.6, os clientes 2 e 8 da rota 1 são trocados com os clientes 1 e 5 da rota 3.

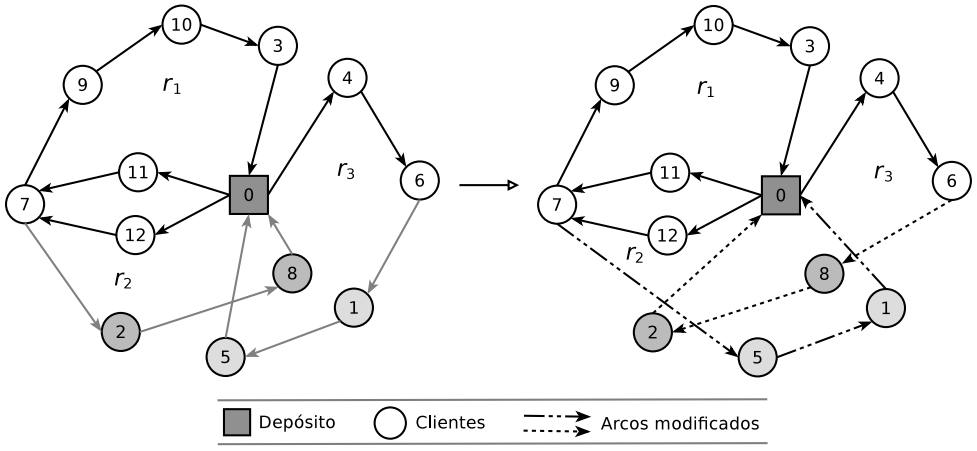

Figura 4.6: Vizinhança inter-rotas *Swap(2,2)*.

- ***Cross*** $- N^{(6)} -$ O arco entre os clientes i e j da rota r_1 e o arco entre os clientes k e l da rota r_2 são removidos. Em seguida, os arcos (i, l) e (k, j) são criados. A Figura 4.7 ilustra o movimento no qual o arco $(2, 8)$ da rota 2 e o arco $(1, 5)$ da rota 3 são removidos e os arcos $(2, 5)$ e $(1, 8)$ são criados.

- ***Swap(1,1)**** $- N^{(7)} -$ Introduzida por Boudia et al. [17], esta estrutura estende a *Swap(1,1)* ao modificar a quantidade de demanda trocada entre os clientes envolvidos no movimento. Dado os clientes $i \in r_1$ e $j \in r_2$, dois casos foram considerados: (i) se $d_i > d_j$ então o cliente j da rota r_2 é inserido antes ou depois do cliente i na rota r_1 (a opção que for mais barata) e uma cópia do cliente i com demanda $d_i^{'} = d_j$ substitui o cliente j na rota r_2; (ii) se $d_j > d_i$, o cliente i da rota r_1 é inserido antes

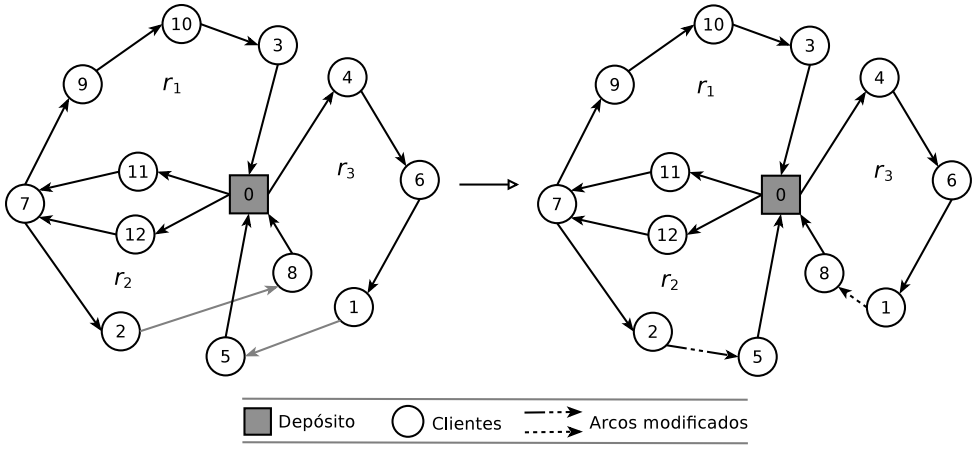

Figura 4.7: Vizinhança inter-rotas *Cross*.

ou depois do cliente j na rota r_2 (a opção que for mais barata) e o cliente i na rota r_1 é substituído por uma cópia do cliente j com demanda $d_j^{'} = d_i$. A configuração em que $d_i = d_j$, corresponde a um caso particular do *Swap(1,1)* e não é considerada nesta estrutura de vizinhança. A Figura 4.8 mostra um exemplo onde uma cópia do cliente 8 da rota 2 é trocada com o cliente 5 da rota 3. Nesta situação, $d_8 > d_5$, o que corresponde ao caso (i).

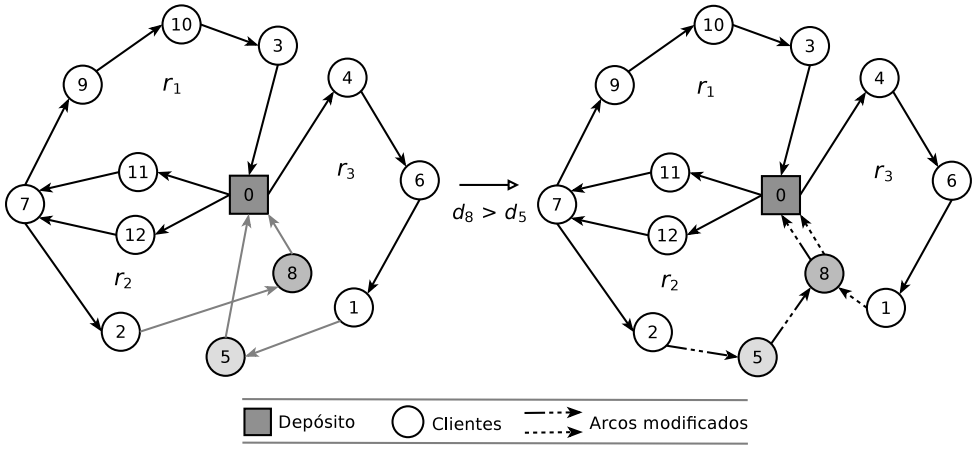

Figura 4.8: Vizinhança inter-rotas *Swap*(1,1)*.

- ***Swap(2,1)**** – $N^{(8)}$ – Também foi proposta por Boudia et al. [17]. Esta vizinhança é uma extensão da *Swap(2,1)* e emprega um esquema similar ao da estrutura anterior. Nesta, dado dois clientes adjacentes $i \in r_1$ e $j \in r_1$ e um cliente $k \in r_2$, também são considerados dois casos: (i) se $(d_i + d_j) > d_k$ e $(d_i < d_k)$ então o cliente i e uma cópia do cliente j com demanda $d_j^{'} = (d_k - d_i)$ substitui o cliente k na rota r_2 e o cliente k é inserido antes ou depois do cliente j na rota r_1 (a opção que proporcionar o maior ganho); (ii) se $d_k > (d_i + d_j)$, os clientes i e j da rota r_1 são

inseridos antes ou depois do cliente k na rota r_2 (a opção que proporcionar o maior ganho) e os clientes i e j na rota r_1 são substituídos por uma cópia do cliente k com demanda $d'_k = (d_i + d_j)$. A configuração onde $(d_i + d_j) = d_k$ corresponde a um caso particular da *Swap(2,1)* e não será considerada nesta vizinhança. A Figura 4.9 mostra um exemplo onde uma cópia do cliente 8 da rota 2 é trocada pelos clientes 5 e 1 da rota 3. Nesta configuração, $d_8 > (d_5 + d_1)$, o que corresponde ao caso (ii).

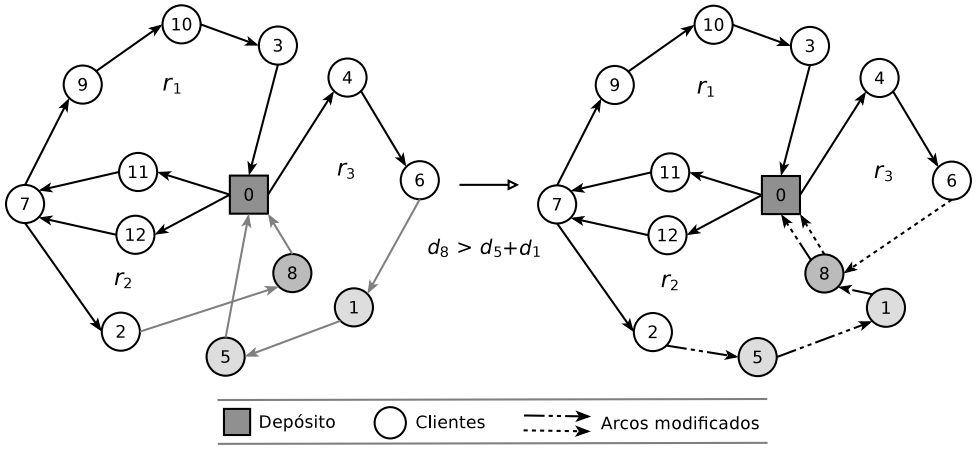

Figura 4.9: Vizinhança inter-rotas *Swap*(2,1)*.

- ***RouteAddition*** $- N^{(9)}$ – Para esta vizinhança, seja o cliente i que aparece em pelo menos duas rotas distintas r_1 e r_2. Inicialmente, o cliente i é removido de ambas as rotas em que aparece. Em seguida, dado os quatro segmentos principais das rotas r_1 e r_2, ou seja, os segmentos que vão do depósito até o cliente anterior a i e do cliente após i até o depósito (em ambas as rotas), escolhe-se a melhor maneira de combinar esses quatro segmentos de rota e o cliente i de tal forma que três novas rotas sejam criadas. Nos casos em que i é atendido por mais de duas rotas, todas as combinações possíveis envolvendo duas rotas são testadas. Esta estrutura de vizinhança foi proposta por Dror et al. [29], mas os autores consideraram combinações que envolviam até três rotas. A Figura 4.10 ilustra a situação onde esta vizinhança é aplicada no cliente 7 atendido pelas rotas r_1 e r_2, dando origem a uma nova rota que atende integralmente a demanda do cliente 7.

- ***k-Split*** $- N^{(10)}$ – Nesta estrutura de vizinhança, o cliente i é removido da solução s e então o procedimento `SplitReinsertion` é chamado para gerar uma nova solução s'. Se existir apenas uma ocorrência do cliente i na solução s, a rota $\bar{r}$ associada a i é considerada proibida para reinserção.

As vizinhanças acima são examinadas exaustivamente e a melhor solução vizinha

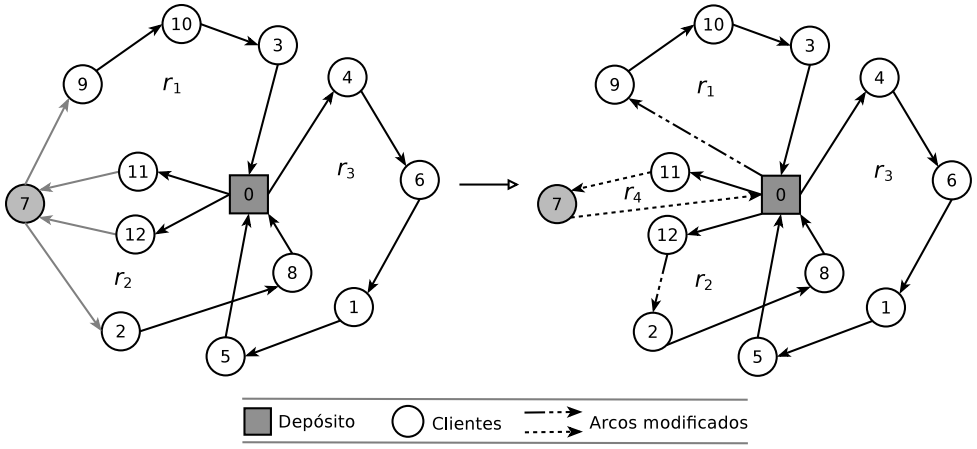

Figura 4.10: Vizinhança inter-rotas *RouteAdition*.

viável encontrada é considerada (estratégia *Best Improvement*). Estruturas de dados auxiliares similares às empregadas por Penna et al. [50] foram utilizadas para melhorar o desempenho da busca local. A complexidade computacional das primeiras oito estruturas é $\mathcal{O}(n^2)$, onde n é o número de clientes. A complexidade da estrutura *k-Split* é $\mathcal{O}(n^2 + nK)$ e da estrutura *RouteAddition* é $\mathcal{O}(n + k^2)$, onde n é o número de clientes e K é o número de rotas. Embora a busca local utilize muitas estruturas de vizinhança, experimentos preliminares mostraram que cada uma delas desempenha um papel importante, dependendo das características da instância. Por exemplo, quando a demanda média dos clientes é baixa, as estruturas clássicas para PRV parecem ser mais efetivas, uma vez que a ocorrência de divisão de demanda é raramente observada em tais casos.

4.4.2 Vizinhanças Intrarrotas

A busca local intrarrota também é realizada por um procedimento RVND. O conjunto N' de estruturas de vizinhança intrarrota utilizadas estão descritas a seguir. Todas elas são consideradas estruturas clássicas na literatura que aborda o Problema do Caixeiro Viajante (PCV) e PRV.

- ***Exchange*** $- N'^{(1)} -$ Troca entre dois clientes. A Figura 4.11 ilustra o movimento no qual os clientes 12 e 2 são trocados, na rota r_2.
- ***2-opt*** $- N'^{(2)} -$ Dois arcos não adjacentes são removidos e outros dois são criados para gerar uma nova rota. O movimento no qual os arcos $(0, 12)$ e $(2, 8)$ são removidos e os arcos $(0, 2)$ e $(12, 8)$ são criados na rota r_2 é ilustrado pela Figura 4.12.

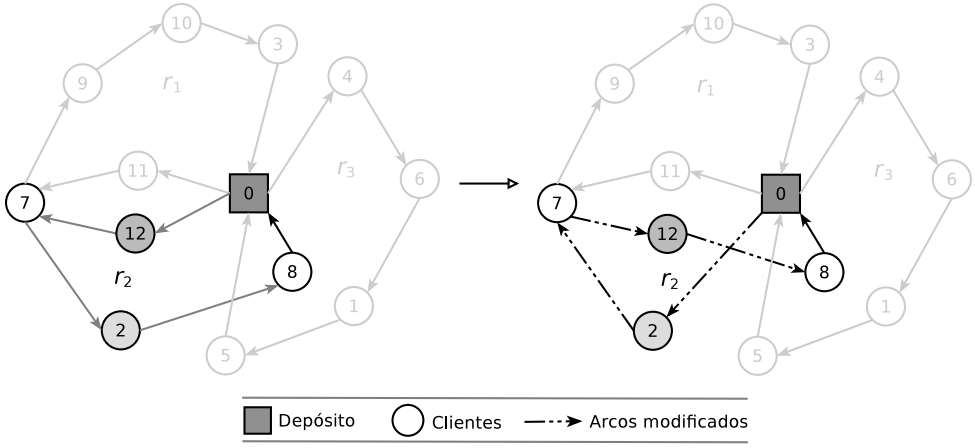

Figura 4.11: Vizinhança intrarrotas *Exchange*.

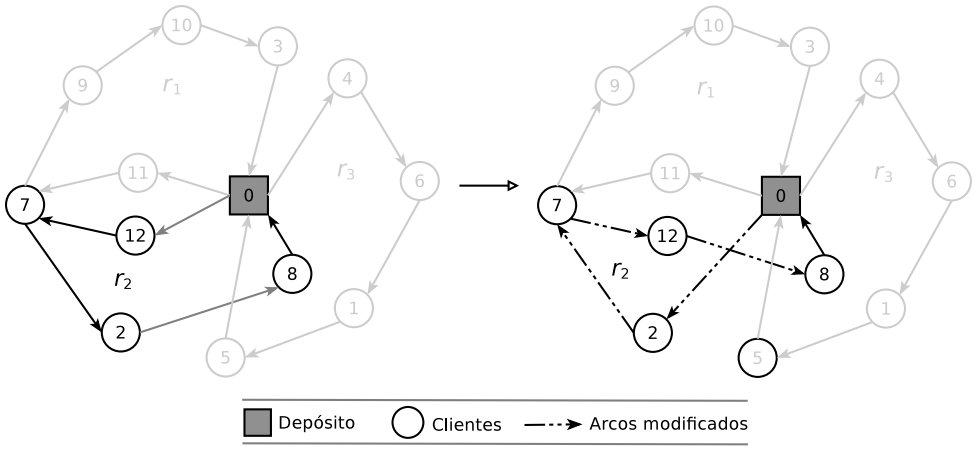

Figura 4.12: Vizinhança intrarrotas *2-opt*.

- ***Reinsertion*** $-$ $N'^{(3)}$ $-$ Um único cliente é removido e reinserido em outra posição da rota. A Figura 4.13 ilustra o movimento no qual o cliente 12, localizado entre os clientes 0 e 7, é removido e reinserido entre os clientes 2 e 8, na rota r_2.

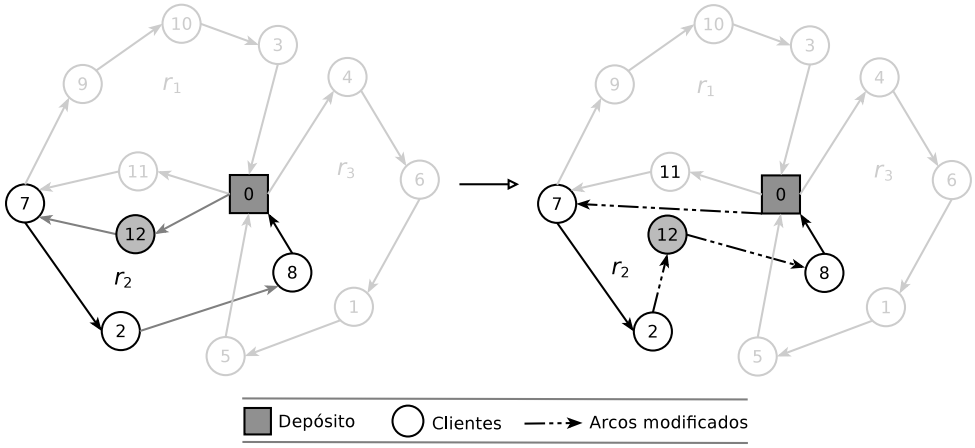

Figura 4.13: Vizinhança intrarrotas *Reinserção*.

- ***Or-opt2*** $-$ $N'^{(4)}$ $-$ Um arco é removido e reinserido em outra posição da rota. A Figura 4.14 ilustra o movimento no qual o arco $(2, 8)$ é removido e reinserido entre os clientes 0 e 12, na rota r_2.

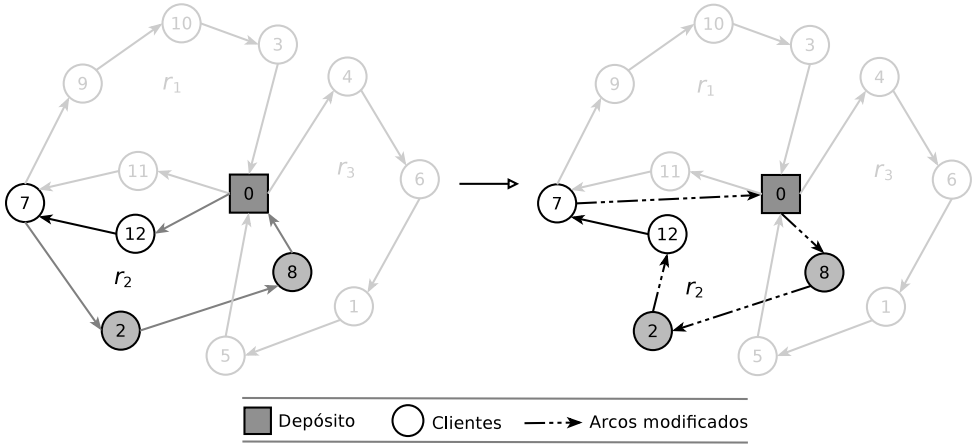

Figura 4.14: Vizinhança intrarrotas *Or-opt2*.

- ***Or-opt3*** $-$ $N'^{(5)}$ $-$ Três clientes consecutivos são removidos e reinseridos para outra posição da rota. A Figura 4.15 ilustra o movimento no qual os clientes 9, 10 e 3 são removidos e reinseridos entre os clientes 0 e 11, na rota r_1.

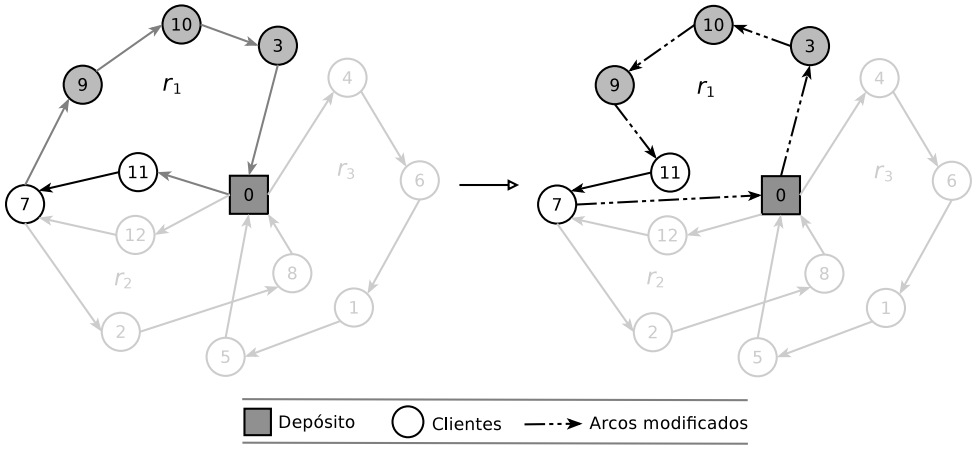

Figura 4.15: Vizinhança intrarrotas *Or-opt3*.

Neste caso, todos os movimentos possíveis são avaliados. A complexidade computacional em todos os casos é $\mathcal{O}(\bar{n}^2)$, onde $\bar{n}$ é o número de clientes nas rotas modificadas.

4.5 Mecanismos de Perturbação

Os mecanismos apresentados são baseados em estruturas de vinhança empregadas durante a fase de busca local. As soluções perturbadas são obtidas por meio da aplicação de um ou mais movimentos selecionados aleatoriamente. Um conjunto P de três mecanismos de perturbação foi desenvolvido. Estes são descritos a seguir.

- ***Multiple-k-Split*** $- P^{(1)} -$ Uma lista composta de 5, 6 ou 7 clientes selecionados aleatoriamente é construída e então o *k-Split* é aplicado para cada cliente da lista.
- ***RouteAddition+k-split*** $- P^{(2)} -$ Inicialmente, se existe pelo menos um cliente sendo visitado por mais de uma rota, um movimento da vizinhança *RouteAddition* é aplicado aleatoriamente, caso não exista, uma rota não unitária é selecionada aleatoriamente e dividida em duas, o ponto em que a rota será dividida também é selecionado aleatoriamente. Em seguida, o mecanismo *Multiple-k-Split* é chamado.
- ***Multiple-Swap(1,1)**** $- P^{(3)} -$ A vizinhança *Swap(1,1)** é aplicada aleatoriamente várias vezes. Após experimentos preliminares, o número de movimentos consecutivos foi selecionado aleatoriamente entre 4, 5 e 6.

Após testes preliminares envolvendo estas três perturbações, embora todas tenham apresentado bom desempenho, optou-se por utilizar na versão final do algoritmo apenas o mecanismo *Multiple-k-Split*. Visto que este, em média, obteve melhor desempenho em termos de qualidade de solução e tempo computacional. Vale notar que, nas situações em que o esforço computacional não é levado em consideração, pode-se obter soluções de melhor qualidade ao aumentar o número de movimentos a serem aplicados tanto no *Multiple-k-Split* quanto no *Multiple-Swap(1,1)**.

4.6 Algoritmo Proposto

O algoritmo proposto, denominado SplitILS, é uma heurística *multi-start* baseada na meta-heurística ILS e no procedimento *Randomized Variable Neighborhood Descent* (RVND) [58]. O Algoritmo 9 mostra o pseudocódigo do SplitILS. A heurística executa I_{Max} iterações (linhas 3 a 23) e em cada uma delas uma solução inicial é gerada utilizando um procedimento construtivo baseado em inserção (linha 4). Em seguida, o laço principal do ILS (linhas 7 a 18) executa enquanto não houver I_{ILS} iterações sem melhora. Em cada uma destas tenta-se melhorar a solução corrente através de um procedimento RVND na

fase de busca local (linha 8). Após busca local, se a frota for limitada e o ótimo local encontrado contém mais rotas que o número mínimo de veículos, a rota com menor carga é esvaziada usando o procedimento `EmptyRoutes` (ver Seção 4.2) (linhas 9 a 11). Se uma solução melhor é encontrada (critério de aceitação), o acumulador de iterações $iterILS$ é reinicializado (linhas 12 a 15). A melhor solução corrente é então perturbada (linha 16). No final, é retornada a melhor solução encontrada durante toda a execução do algoritmo.

Algoritmo 9: SplitILS

```
1  Procedimento SplitILS(I_Max, I_ILS)
2  f* ← ∞;
3  para i ← 1, ..., I_Max faça
4      s ← GSolInicial();
5      s' ← s;
6      iterILS ← 0;
7      enquanto iterILS < I_ILS faça
8          s ← RVND(s);
9          se PRVEF-FL então
10             s ← EmptyRoutes(s);
11         fim se
12         se f(s) < f(s') então
13             s' ← s;
14             iterILS ← 0;
15         fim se
16         s ← Perturba(s');
17         iterILS ← iterILS + 1;
18     fim enquanto
19     se f(s') < f* então
20         s* ← s';
21         f* ← f(s');
22     fim se
23 fim para
24 retorne s*;
25 fim
```

Capítulo 5

Resultados Computacionais

O algoritmo foi codificado em C++ (g++ 4.4.3) e executado em um computador com processador Intel® Core™ i7 2,93 GHz, com 8,0 GB de memória RAM e Sistema Operacional GNU/Linux Ubuntu 10.04 (*kernel* 2.6.32-25). Somente uma *thread* foi utilizada nos experimentos.

O número de iterações, I_{max}, foi fixado em 10, enquanto o número de iterações sem melhora do ILS, I_{ILS}, foi fixado como sendo o $\min\{K_{min} \times n, 5000\}$. Estes valores foram calibrados após experimentos preliminares de forma que, para cada um dos conjuntos de instâncias, a média dos *gaps* médios, entre as médias das soluções do SplitILS e as melhores soluções conhecidas, fossem negativas. O algoritmo foi executado 20 vezes para cada instância.

Nas tabelas apresentadas neste capítulo, a notação descrita a seguir foi utilizada, **Problem** indica o nome da instância, **LB** é o limite inferior (*lower bound* - LB), **BKS** é o valor da melhor solução conhecida (*Best Known Solution* - BKS), **Best Sol.** é o valor da melhor solução encontrada pelo respectivo trabalho, **Avg. Sol** é a média dos valores das soluções encontradas pelo SplitILS, **Avg. Gap** é o *gap* entre a média dos valores das soluções encontradas pelo SplitILS e a BKS (Avg. Gap = (100(Avg. Sol - BKS)/BKS)), *gaps* negativos indicam que o valor da solução encontrada pelo SplitILS é melhor que o valor da solução da literatura. **Avg. Gap LB** é o *gap* entre a média dos valores das soluções encontradas pelo SplitILS e o LB (Avg. Gap = (100(Avg. Sol - LB)/LB)), **Avg. Time** é a média dos tempos em segundos para as 20 execuções. **Avg. Time BKS** é a média dos tempos gastos pelo SplitILS para alcançar ou melhorar a BKS, **Best All Exp.** é o melhor valor de solução encontrada pelo SplitILS considerando todos os experimentos, **cTime** indica os tempos de processamento dos computadores da

literatura convertidos conforme Dongarra [27], com relação ao processador de 2.93 GHz empregado neste trabalho. Os valores das melhores soluções estão indicados em negrito e os casos em que o SplitILS encontrou soluções melhores que a literatura estão indicados com sublinhado.

O algoritmo proposto foi testado em quatro conjuntos de instâncias disponíveis na literatura. O primeiro foi gerado por Belenguer et al. [14] e é composto de 25 problemas-teste, dos quais, 11 foram adaptados da TSPLIB [51] e os outros 14 foram gerados aleatoriamente da forma indicada a seguir. As coordenadas dos clientes são aquelas das instâncias *eil51, eil76* e *eil101* da TSPLIB. A capacidade do veículo é $Q = 160$ e as demandas dos clientes foram geradas de acordo com Dror et al. [29]. Nas 14 instâncias geradas aleatoriamente, o D1 que aparece no nome do problema, indica que a demanda d_i do cliente i foi selecionada aleatoriamente do intervalo $[\lceil 0.01Q \rceil, \lfloor 0.1Q \rfloor]$, D2 $[\lceil 0.1Q \rceil, \lfloor 0.3Q \rfloor]$, D3 $[\lceil 0.1Q \rceil, \lfloor 0.5Q \rfloor]$, D4 $[\lceil 0.1Q \rceil, \lfloor 0.9Q \rfloor]$, D5 $[\lceil 0.3Q \rceil, \lfloor 0.7Q \rfloor]$ e D6 $[\lceil 0.7Q \rceil, \lfloor 0.9Q \rfloor]$.

O segundo conjunto de instâncias foi gerado por Campos et al. [19], seguindo o mesmo procedimento empregado para gerar o conjunto de instâncias apresentado por Archetti et al. [12]. Porém, este último não está mais disponível. O conjunto é composto de 49 problemas-teste com até 199 clientes e está dividido em 7 grupos. O primeiro é formado pelas instâncias originais e os outros seguem a mesma forma de geração descrito acima, levando em conta os intervalos D1-D6, respectivamente.

O terceiro conjunto de instâncias foi proposto por Archetti et al. [13] e também segue a abordagem sugerida por Archetti et al. [12]. As características das instâncias para este conjunto são similares àquelas geradas por Campos et al. [19], com os valores das demandas diferentes.

O quarto conjunto de instâncias foi gerado por Chen et al. [21]. Este conjunto é composto de 21 problemas-teste variando de 8 até 288 clientes. A capacidade do veículo em todos os casos é $Q = 100$ e as demandas assumem o valor 60 ou 90. Os clientes estão distribuídos de forma concêntrica em torno do depósito. Os autores afirmam que com esta forma de distribuição é possível estimar visualmente o custo das soluções que estão próximas do ótimo.

5.1 Resultados para o PRVEF com Frota Limitada

A Tabela 5.1 apresenta os resultados obtidos no conjunto de 25 instâncias de Belenguer et al. [14]. Pode-se verificar que o SplitILS foi capaz de encontrar a mesma solução da

literatura em três instâncias e melhorar as outras 22. O *gap* médio entre a média dos valores das soluções encontradas pelo SplitILS e as melhores soluções da literatura foi de -1.46%. Resultados similares foram encontrados para o mesmo conjunto de instâncias, mas usando valores de custos arredondados para o inteiro mais próximo, conforme mostrado na Tabela 5.2. Neste caso, as mesmas soluções foram encontradas em sete problemas-teste, melhoradas em 18 e *gap* médio -1.87%. Esses resultados mostram que o SplitILS obteve um desempenho superior ao algoritmo proposto por Aleman et al. [3] em termos de qualidade das soluções. Apesar do tempo computacional total do algoritmo proposto ser alto quando comparado àquele obtido pelo algoritmo de Aleman et al. [3], o SplitILS consumiu, em média, apenas 1.69 segundos para encontrar ou superar as melhores soluções conhecidas.

Tabela 5.1: Resultados para o PRVEF-FL nas instâncias de Belenguer et al. [14]

			Aleman et al.[d]		SplitILS						
Problem	LB	BKS	Best Sol.	Time (s)	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
eil22	-	375,28[b]	**375,28**	4,19	**375,28**	**375,28**	-	0,00	0,14	0,00	375,28
eil23	525,65	569,75[b]	**569,75**	3,42	**568,56**	**568,56**	8,16	-0,21	0,12	0,00	568,56
eil30	-	512,72[b]	**512,72**	14,47	**512,72**	**512,72**	-	0,00	0,32	0,00	512,72
eil33	-	853,10[b]	**853,10**	14,03	**837,06**	**837,06**	-	-1,88	0,45	0,00	837,06
eil51	518,26	524,61[b]	**524,61**	54,91	**524,61**	**524,61**	1,23	0,00	1,63	0,24	524,61
eilA76	809,67	829,58[a]	851,24	83,28	**823,89**	**825,22**	1,92	-0,53	27,25	3,19	823,89
eilB76	985,42	1023,23[a]	1059,57	79,00	**1009,04**	**1011,19**	2,62	-1,18	44,98	0,15	1009,04
eilC76	723,55	753,29[b]	**753,29**	148,20	**738,67**	**739,83**	2,25	-1,79	15,68	0,01	738,67
eilD76	672,54	699,35[b]	**699,35**	140,83	**687,60**	**688,37**	2,35	-1,57	9,92	0,03	687,60
eilA101	803,62	851,39[a]	852,74	319,33	**826,14**	**826,26**	2,82	-2,95	36,59	0,01	826,14
eilB101	1055,40	1118,93[a]	1139,27	185,84	**1076,26**	**1078,58**	2,20	-3,61	101,26	0,06	1076,26
S51D1	457,10	464,78[a]	471,92	40,53	**459,50**	**459,50**	0,53	-1,14	1,07	0,02	459,50
S51D2	700,40	719,82[a]	731,01	28,34	**708,42**	**709,54**	1,30	-1,43	9,98	0,15	708,42
S51D3	938,50	955,56[a]	1001,22	14,70	**948,01**	**949,96**	1,22	-0,59	14,15	0,34	947,97
S51D4	1549,70	1614,36[a]	1680,66	16,53	**1561,01**	**1563,25**	0,87	-3,17	59,96	0,01	1560,92
S51D5	1318,90	1343,07[a]	1389,40	13,94	**1333,67**	**1333,85**	1,13	-0,69	32,41	0,21	1333,67
S51D6	2154,70	2179,65[a]	2218,23	16,83	**2169,10**	**2174,71**	0,93	-0,23	83,79	30,90	2169,10
S76D1	592,60	606,47[b]	**606,47**	476,27	**598,94**	**598,98**	1,08	-1,24	4,54	0,25	598,94
S76D2	1071,30	1129,06[a]	1143,36	46,94	**1087,99**	**1089,69**	1,72	-3,49	74,51	0,08	1087,99
S76D3	1407,54	1442,08[a]	1490,08	53,34	**1427,81**	**1429,01**	1,53	-0,91	88,72	1,76	1427,81
S76D4	2059,80	2100,15[a]	2173,61	51,84	**2079,76**	**2080,76**	1,02	-0,92	173,55	2,69	2079,76
S101D1	716,80	749,19[b]	**749,19**	2125,58	**726,59**	**728,44**	1,62	-2,77	14,16	0,01	726,59
S101D2	1358,90	1417,87[a]	1443,44	217,91	**1383,35**	**1386,45**	2,03	-2,22	129,94	0,06	1378,19
S101D3	1853,10	1906,28[a]	1988,78	146,61	**1876,97**	**1881,26**	1,52	-1,31	277,62	1,84	1876,48
S101D5	2767,60	2873,08[c]	2984,48	104,05	**2792,01**	**2795,73**	1,02	-2,69	696,64	0,38	2789,54
Média				176,04			1,87	-1,46	75,97	1,69	

a - Archetti et al. [5]
b - Aleman et al. [3]
c - Wilck IV e Cavalier [63], Xeon 2,49 GHz, uma única execução
d - Pentium 4, 2,8 GHz, uma única execução, Avg. cTime (s): 73,68

A Tabela 5.3 mostra os resultados encontrados para o conjunto de 49 instâncias de Campos et al. [19]. O desempenho do algoritmo proposto neste conjunto de instâncias foi bastante razoável, uma vez que em 44 dos 49 problemas-teste, as soluções encontradas são melhores que as reportadas na literatura e nos cinco problemas restantes, a mesma solução foi alcançada. Comparando com a literatura, o *gap* médio do SplitILS em relação a melhor

Tabela 5.2: Resultados para o PRVEF-FL nas instâncias de Belenguer et al. [14] usando custos arredondados

			Aleman et al.[c]		SplitILS						
Problem	LB	BKS	Best Sol.	Time (s)	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
eil22	375,00	*375[a]	**375**	4,19	**375**	**375,00**	0,00	0,00	0,13	0,01	375
eil23	569,00	*569[a]	570	3,42	**569**	**569,00**	0,00	0,00	0,09	0,00	569
eil30	510,00	*510[a]	**510**	14,47	**510**	**510,00**	0,00	0,00	0,30	0,00	510
eil33	834,70	*835[a]	851	14,03	**835**	**835,00**	0,04	0,00	0,39	0,03	835
eil51	517,80	*521[a]	**521**	54,91	**521**	521,55	0,72	0,11	1,63	0,14	521
eilA76	807,60	832[a]	847	83,28	**818**	**820,45**	1,59	-1,39	25,68	0,35	818
eilB76	981,40	1023[a]	1055	79,00	**1002**	**1005,80**	2,49	-1,68	38,05	0,33	1002
eilC76	717,80	735[a]	746	148,20	**733**	**733,55**	2,19	-0,20	15,17	1,78	732
eilD76	666,10	683[a]	695	140,83	**681**	**683,00**	2,54	0,00	11,02	2,87	681
eilA101	799,80	817[a]	843	319,33	**815**	**815,85**	2,01	-0,14	32,70	8,57	814
eilB101	1040,60	1077[a]	1122	185,84	**1061**	**1065,40**	2,38	-1,08	75,43	8,34	1061
S51D1	454,40	458[a]	466	40,53	**458**	**458,00**	0,79	0,00	1,21	0,09	458
S51D2	694,20	725[b]	**725**	28,34	**703**	**704,65**	1,51	-2,81	8,32	0,06	703
S51D3	930,70	972[a]	994	14,70	**943**	**944,20**	1,45	-2,86	13,58	0,03	943
S51D4	1539,00	1672[b]	**1672**	16,53	**1552**	**1555,55**	1,08	-6,96	47,34	0,01	1551
S51D5	1313,40	1385[b]	**1385**	13,94	**1328**	**1329,15**	1,20	-4,03	33,46	0,02	1328
S51D6	2141,70	2211[b]	**2211**	16,83	**2163**	**2165,70**	1,12	-2,05	65,68	0,01	2162
S76D1	586,60	594[a]	600	476,27	**592**	**592,45**	1,00	-0,26	4,75	1,33	592
S76D2	1061,10	1138[b]	**1138**	46,94	**1081**	**1083,35**	2,10	-4,80	59,20	0,03	1080
S76D3	1395,90	1474[a]	1485	53,34	**1419**	**1422,05**	1,87	-3,52	68,07	0,03	1419
S76D4	2046,10	2160[b]	**2160**	51,84	**2071**	**2074,30**	1,38	-3,97	148,48	0,01	2071
S101D1	704,90	716[a]	740	2125,58	**716**	718,40	1,92	0,34	14,17	13,00	716
S101D2	1337,10	1393[a]	1426	217,91	**1364**	**1370,95**	2,53	-1,58	116,33	0,45	1364
S101D3	1832,20	1974[b]	1974	146,61	**1859**	**1868,75**	1,99	-5,33	233,36	0,01	1859
S101D5	2737,10	2915[a]	2970	104,05	**2772**	**2779,65**	1,55	-4,64	579,68	0,06	2772
Média				176,04			1,42	-1,87	63,77	1,50	

∗ - Solução ótima
a - Belenguer et al. [14]
b - Aleman et al. [3]
c - Pentium 4, 2,8 GHz, uma única execução, Avg. cTime (s): 73,68

solução conhecida foi de -2.23%. Além disso, pode-se observar que o SplitILS foi capaz, na média, de encontrar ou melhorar rapidamente as soluções conhecidas quando comparado com o algoritmo de Campos et al. [19] e Aleman et al. [3]. Para este conjunto de instâncias não foi possível estimar o quão distantes as soluções encontradas pelo SplitILS estão das soluções ótimas, visto que não há limites inferiores disponíveis na literatura.

Os resultados obtidos no conjunto de 42 instâncias proposto por Archetti et al. [13] são mostrados na Tabela 5.4. Pode-se verificar que o SplitILS foi capaz de encontrar soluções ainda melhores que aquelas reportadas na literatura com um *gap* médio de -2.81%. Assim como nos casos anteriores, as melhores soluções conhecidas foram encontradas ou superadas rapidamente.

Por fim, no conjunto de 21 instâncias de Chen et al. [21], o SplitILS foi capaz de encontrar as mesmas soluções em cinco problemas-teste e de melhorar outras sete, com um *gap* médio de -0.02%, conforme mostrado na Tabela 5.5. O *gap* médio em relação aos limites inferiores foi de apenas 0.53%, indicando que as soluções encontradas pelo algoritmo proposto são, em média, de alta qualidade. Contudo, este é o único conjunto

Tabela 5.3: Resultados para o PRVEF-FL nas instâncias de Campos et al. [19]

		Campos et al.[c]		Aleman et al.[d]		SplitILS					
Problem	BKS	Best Sol.	Time (s)	Best Sol.	Time (s)	Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
p01-50	524,61[a]	**524,61**	49,70	**524,61**	54,91	**524,61**	**524,61**	0,00	1,87	0,22	524,61
p02-75	829,01[a]	**829,01**	166,50	851,24	83,28	**823,89**	**824,77**	-0,51	30,84	6,72	823,89
p03-100	829,45[a]	**829,45**	276,10	852,74	319,33	**826,14**	**826,39**	-0,37	40,81	3,20	826,14
p04-150	1045,22[a]	**1045,22**	527,10	1074,11	1361,16	**1024,59**	**1026,60**	-1,78	251,66	1,06	1023,66
p05-199	1324,73[a]	**1324,73**	588,30	1368,67	3284,64	**1289,40**	**1296,37**	-2,14	1594,46	10,70	1286,08
p06-120	1042,12[a]	**1042,12**	270,30	1201,83	3414,41	**1037,88**	1043,41	0,12	90,06	20,49	1037,88
p07-100	819,56[a]	**819,56**	192,40	824,78	126,08	**819,56**	**819,56**	0,00	27,99	0,13	819,56
p01-50D1	460,79[a]	**460,79**	51,80	471,92	33,70	**460,79**	**460,79**	0,00	1,16	0,02	460,79
p02-75D1	596,99[a]	**596,99**	144,00	597,46	303,77	**596,25**	**596,25**	-0,12	5,00	0,61	596,25
p03-100D1	726,81[a]	**726,81**	272,10	745,35	2194,23	**726,81**	730,01	0,44	17,72	13,70	726,81
p04-150D1	871,26[a]	**871,26**	743,30	891,98	3461,44	**866,31**	**866,31**	-0,57	119,63	2,77	866,31
p05-199D1	1023,14[a]	**1023,14**	1874,80	1073,55	15505,22	**1017,30**	**1018,40**	-0,46	438,21	27,99	1017,28
p06-120D1	976,57[a]	**976,57**	370,90	1087,80	3952,67	**975,96**	**976,42**	-0,02	46,16	8,15	975,96
p07-100D1	636,00[a]	**636,00**	166,50	673,54	1207,42	**632,63**	**633,11**	-0,45	14,38	2,75	632,63
p01-50D2	741,06[a]	**741,06**	66,40	766,19	19,77	**741,06**	741,26	0,03	9,87	4,12	741,06
p02-75D2	1071,87[a]	**1071,87**	143,80	1099,47	73,05	**1064,49**	**1066,87**	-0,47	53,42	3,14	1064,49
p03-100D2	1397,50[a]	**1397,50**	305,10	1425,90	190,53	**1376,09**	**1380,28**	-1,23	182,16	1,85	1374,19
p04-150D2	1937,20[a]	**1937,20**	326,60	1978,01	878,55	**1861,63**	**1866,48**	-3,65	1055,54	0,29	1861,53
p05-199D2	2433,17[a]	**2433,17**	32,10	2464,65	1457,16	**2307,82**	**2313,37**	-4,92	2440,32	0,16	2306,80
p06-120D2	2742,60[a]	**2742,60**	380,80	2806,92	558,56	**2703,75**	**2708,51**	-1,24	762,81	6,43	2703,75
p07-100D2	1418,81[a]	**1418,81**	206,30	1428,27	123,00	**1413,85**	**1413,91**	-0,35	130,70	1,21	1413,85
p01-50D3	997,83[a]	**997,83**	87,10	1039,89	18,16	**982,77**	**983,70**	-1,42	18,44	0,40	982,77
p02-75D3	1463,60[a]	**1463,60**	126,80	1478,67	67,80	**1393,11**	**1393,11**	-4,82	101,77	0,04	1393,11
p03-100D3	1908,02[a]	**1908,02**	225,20	1956,13	154,47	**1823,17**	**1827,47**	-4,22	326,55	0,06	1822,36
p04-150D3	2649,97[a]	**2649,97**	21,30	2671,62	625,83	**2528,51**	**2531,79**	-4,46	1514,55	0,08	2525,51
p05-199D3	3291,96[a]	**3291,96**	31,20	3411,38	2173,84	**3153,01**	**3163,89**	-3,89	3895,07	0,32	3153,01
p06-120D3	3979,67[a]	**3979,67**	329,00	4026,53	358,56	**3907,27**	**3910,03**	-1,75	1543,98	26,71	3907,27
p07-100D3	1995,34[a]	**1995,34**	266,50	2007,11	107,47	**1967,41**	**1967,93**	-1,37	260,65	0,41	1967,41
p01-50D4	1522,43[b]	1554,38	92,60	**1522,43**	16,36	**1456,00**	**1456,87**	-4,31	46,74	0,01	1456,00
p02-75D4	2182,34[a]	**2182,34**	119,90	2200,51	71,11	**2081,38**	**2084,62**	-4,48	219,74	0,02	2081,37
p03-100D4	2865,86[b]	2894,21	177,90	**2865,86**	126,52	**2751,13**	**2754,52**	-3,89	629,59	0,03	2749,05
p04-150D4	4062,88[a]	**4062,88**	50,40	4165,18	671,36	**3988,06**	**3997,49**	-1,61	1986,49	2,41	3988,06
p05-199D4	5074,57[a]	**5074,57**	50,70	5184,57	3650,59	**4844,58**	**4855,82**	-4,31	3806,84	0,25	4842,97
p06-120D4	6357,33[a]	**6357,33**	20,60	6364,87	458,91	**6201,66**	**6215,87**	-2,23	1975,24	0,57	6196,48
p07-100D4	3156,31[b]	3166,31	272,70	**3156,31**	96,98	**3087,75**	**3088,96**	-2,13	435,09	0,04	3087,75
p01-50D5	1532,19[a]	**1532,19**	92,40	1540,39	15,33	**1467,47**	**1467,47**	-4,22	48,93	0,01	1467,47
p02-75D5	2228,90[a]	**2228,90**	11,10	2238,98	80,30	**2112,19**	**2113,38**	-5,18	267,72	0,02	2110,20
p03-100D5	2941,64[b]	2986,33	17,00	**2941,64**	103,94	**2813,82**	**2817,05**	-4,24	737,35	0,04	2813,35
p04-150D5	4165,18[b]	4185,68	23,00	**4165,18**	675,39	**3986,49**	**3996,85**	-4,04	2076,38	0,16	3986,49
p05-199D5	5265,01[a]	**5265,01**	327,30	5363,65	3026,22	**5061,25**	**5070,77**	-3,69	4570,46	0,33	5061,25
p06-120D5	6481,09[a]	**6481,09**	20,50	6545,50	469,17	**6372,58**	**6375,64**	-1,63	2289,79	1,93	6370,97
p07-100D5	3225,63[b]	3248,76	16,00	**3225,63**	110,05	**3125,47**	**3126,22**	-3,08	619,62	0,03	3125,47
p01-50D6	2215,34[b]	2312,48	5,80	**2215,34**	18,70	**2154,21**	**2154,51**	-2,75	83,85	0,01	2154,21
p02-75D6	3304,24[b]	3387,86	10,50	**3304,24**	58,05	**3179,20**	**3181,30**	-3,72	441,77	0,01	3179,08
p03-100D6	4429,21[b]	4576,13	38,30	**4429,21**	94,98	**4294,12**	**4298,50**	-2,95	731,49	0,03	4294,12
p04-150D6	6479,46[a]	**6479,46**	30,50	6482,11	584,84	**6231,01**	**6233,76**	-3,79	1660,06	0,09	6230,49
p05-199D6	8323,72[a]	**8323,72**	215,00	8329,55	2124,66	**8045,18**	**8047,68**	-3,32	4718,09	0,25	8041,01
p06-120D6	10158,32[a]	**10158,32**	20,40	10302,16	636,72	**10001,95**	**10005,18**	-1,51	2209,90	0,08	10001,95
p07-100D6	5028,78[b]	5065,26	13,80	**5028,78**	178,19	**4902,81**	**4907,00**	-2,42	823,19	0,04	4902,37
Média			201,40		1130,15			-2,23	925,59	3,06	

a - Campos et al. [19]
b - Aleman et al. [3]
c - Pentium 4, 2,4 GHz; uma única execução; Avg. cTime (s): 65,91
d - Pentium 4, 2,8 GHz; uma única execução; Avg. cTime (s): 473,04

de instâncias para o PRVEF-FL em que o SplitILS não consegue igualar ou melhorar os resultados em todas as instâncias. Nos casos em que isto acontece, o SplitILS consome muito pouco tempo.

Tabela 5.4: Resultados para o PRVEF-FL nas instâncias de Archetti et al. [13]

			SplitILS						
Problem	LB[a]	BKS[a]	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
p01-50	-	-	524,61	524,61	-	-	1,83	-	524,61
p02-75	-	-	823,89	824,39	-	-	30,31	-	823,89
p03-100	-	-	826,14	826,45	-	-	42,16	-	826,14
p04-150	-	-	1023,87	1026,48	-	-	243,93	-	1023,66
p05-199	-	-	1285,79	1292,79	-	-	1672,72	-	1285,25
p11-120	-	-	1037,88	1038,68	-	-	85,14	-	1037,88
p01-50D1	457,17	459,98	**459,50**	**459,50**	0,51	-0,10	1,21	0,09	459,50
p02-75D1	604,89	637,36	**617,85**	**620,19**	2,53	-2,69	5,72	0,02	617,85
p03-100D1	742,97	794,21	**760,00**	**760,70**	2,39	-4,22	22,00	0,03	760,00
p04-150D1	896,03	1065,94	**921,47**	**923,74**	3,09	-13,34	164,58	0,03	921,44
p05-199D1	1042,37	1188,45	**1074,18**	**1080,65**	3,67	-9,07	629,08	0,04	1074,18
p11-120D1	1023,37	1063,73	**1043,19**	**1043,21**	1,94	-1,93	93,35	8,15	1042,89
p01-50D2	747,25	771,65	**756,71**	**760,52**	1,78	-1,44	14,55	0,14	756,71
p02-75D2	1093,56	1124,70	**1110,43**	**1112,70**	1,75	-1,07	54,11	0,63	1109,62
p03-100D2	1435,23	1492,05	**1458,46**	**1462,37**	1,89	-1,99	200,43	0,48	1458,46
p04-150D2	1986,79	2109,45	**2017,00**	**2021,78**	1,76	-4,16	1156,99	0,13	2016,96
p05-199D2	2423,64	2632,22	**2481,44**	**2487,28**	2,63	-5,51	2846,16	0,14	2478,02
p11-120D2	2879,63	2988,61	**2899,91**	**2905,28**	0,89	-2,79	898,52	0,22	2898,46
p01-50D3	996,93	1011,64	**1005,75**	**1005,93**	0,90	-0,56	21,48	4,19	1005,75
p02-75D3	1483,17	1509,79	**1502,05**	**1503,42**	1,37	-0,42	110,02	2,86	1502,05
p03-100D3	1971,43	2018,09	**1997,76**	**2001,83**	1,54	-0,81	366,31	24,78	1996,76
p04-150D3	2811,64	2956,18	**2849,66**	**2856,41**	1,59	-3,37	1699,36	0,30	2849,66
p05-199D3	3420,17	3548,13	**3472,79**	**3481,37**	1,79	-1,88	3015,92	4,94	3471,71
p11-120D3	4162,99	4308,17	**4219,01**	**4220,59**	1,38	-2,03	2260,39	4,37	4217,59
p01-50D4	1470,91	1490,61	**1488,27**	**1489,05**	1,23	-0,10	52,71	3,54	1488,06
p02-75D4	2270,44	2372,22	**2301,61**	**2304,89**	1,52	-2,84	283,77	0,12	2301,15
p03-100D4	3043,27	3167,29	**3090,65**	**3094,91**	1,70	-2,29	746,44	0,14	3090,40
p04-150D4	4474,18	4708,11	**4543,18**	**4550,63**	1,71	-3,34	2467,51	0,21	4542,52
p05-199D4	5425,69	5894,69	**5526,28**	**5530,56**	1,93	-6,18	5799,52	0,34	5524,97
p11-120D4	6808,07	7020,87	**6856,11**	**6863,96**	0,82	-2,23	3363,54	0,79	6854,51
p01-50D5	1467,09	1498,30	**1481,71**	**1484,62**	1,19	-0,91	42,90	0,57	1481,71
p02-75D5	2192,25	2235,61	**2219,52**	**2222,58**	1,38	-0,58	261,25	18,55	2219,11
p03-100D5	2945,76	3044,92	**2991,22**	**2991,89**	1,57	-1,74	756,18	0,38	2990,27
p04-150D5	4269,77	4637,69	**4336,80**	**4342,45**	1,70	-6,37	2366,91	0,08	4332,45
p05-199D5	5306,11	5669,69	**5404,44**	**5415,31**	2,06	-4,49	5706,50	0,32	5399,91
p11-120D5	6584,11	6860,65	**6674,97**	**6678,58**	1,43	-2,65	2306,51	0,07	6662,11
p01-50D6	2133,94	2162,58	**2156,14**	**2160,60**	1,25	-0,09	84,35	19,39	2155,80
p02-75D6	3177,69	3259,82	**3223,06**	**3226,79**	1,55	-1,01	377,02	0,01	3223,06
p03-100D6	4316,42	4494,53	**4387,32**	**4389,19**	1,69	-2,34	719,27	0,03	4386,72
p04-150D6	6287,09	6529,63	**6396,68**	**6402,63**	1,84	-1,94	2180,59	0,09	6396,44
p05-199D6	8062,24	8400,74	**8188,47**	**8195,06**	1,65	-2,45	3528,41	0,22	8188,47
p11-120D6	10111,11	10456,19	**10215,90**	**10218,78**	1,06	-2,27	2006,24	0,07	10202,34
Média					1,69	-2,81	1159,19		

a - Archetti et al. [5]

Tabela 5.5: Resultados para o PRVEF-FL nas instâncias de Chen et al. [21]

			Wilck et al.[d]		SplitILS						
Problem	LB	BKS	Best Sol.	Time (s)	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
SD1	228,28	*228,28[b]	**228,28**	0,27	**228,28**	**228,28**	0,00	0,00	0,05	0,00	228,28
SD2	708,28	*708,28[b]	**708,28**	1,95	**708,28**	**708,28**	0,00	0,00	0,58	0,00	708,28
SD3	430,40	*430,40[b]	430,58	1,94	430,58	430,58	0,04	0,04	0,59	-	430,58
SD4	630,62	*630,62[b]	631,05	6,24	631,05	631,05	0,07	0,07	2,16	-	631,05
SD5	1381,76	1389,94[b]	1390,57	14,20	1390,57	1390,57	0,64	0,05	5,90	-	1390,57
SD6	830,86	*830,86[b]	833,58	14,97	831,24	831,24	0,05	0,05	5,62	-	831,24
SD7	3640,00	3640,00[b]	**3640,00**	28,61	**3640,00**	**3640,00**	0,00	0,00	13,74	0,00	3640,00
SD8	5068,28	*5068,28[b]	**5068,28**	48,26	**5068,28**	**5068,28**	0,00	0,00	24,07	0,01	5068,28
SD9	2031,22	2042,88[b]	2054,84	48,91	2044,20	2044,43	0,65	0,08	35,86	-	2044,20
SD10	2679,04	2683,73[b]	2746,54	114,16	2684,88	2684,88	0,22	0,04	81,76	-	2684,88
SD11	13275,00	13280,00[b]	**13280,00**	231,64	**13280,00**	**13280,00**	0,04	0,00	136,43	0,01	13280,00
SD12	7175,80	7239,57[b]	7279,97	227,11	**7213,61**	**7216,34**	0,56	-0,32	179,19	0,02	7213,61
SD13	10053,60	10105,86[b]	10110,57	421,95	10110,58	10110,58	0,57	0,05	168,07	-	10110,58
SD14	10588,20	10725,38[b]	10786,52	718,65	**10715,53**	**10722,73**	1,27	-0,02	432,26	196,13	10715,53
SD15	14908,50	15129,68[b]	15160,04	1278,35	**15093,85**	**15102,85**	1,30	-0,18	658,54	0,06	15093,85
SD16	3379,33	*3379,33[b]	3433,83	1225,88	3395,11	3395,16	0,47	0,47	580,27	-	3381,25
SD17	26317,20	26533,39[b]	26559,92	1722,20	**26493,56**	**26499,23**	0,69	-0,13	484,43	0,10	26493,56
SD18	14029,20	14283,51[b]	14302,22	1735,83	**14197,80**	**14202,85**	1,24	-0,56	676,77	0,10	14197,80
SD19	19707,20	20152,53[c]	**20152,53**	3093,17	**19989,95**	**20000,54**	1,49	-0,75	1261,95	0,17	19989,95
SD20	39252,80	39706,51[c]	**39706,51**	6208,16	**39641,91**	**39648,42**	1,01	-0,15	1518,12	0,33	39637,53
SD21	11271,00	*11271,00[a]	11461,20	10565,70	11344,96	11357,62	0,77	0,77	4326,99	-	11344,96
Média				1319,44			0,53	-0,02	504,44		

∗ - Solução ótima
a - Moreno et al. [46]
b - Archetti et al. [5]
c - Wilck IV e Cavalier [63]
d - Xeon 2,49 GHz, uma única execução

5.2 Resultados para o PRVEF com Frota Ilimitada

Na Tabela 5.6 estão listados os resultados encontrados pelo SplitILS no conjunto de instâncias proposto por Belenguer et al. [14]. Na maiorias dos casos, as soluções alcançadas são equivalentes àquelas encontradas para o PRVEF com frota limitada. Entretanto, para alguns problemas-teste, como eil30 e S101D5, existe um ganho em termos de custo da solução quando um veículos extra é considerado. A mesma interpretação pode ser derivada para os resultados obtidos no conjunto de instâncias em que os custos são calculados com arredondamento para o inteiro mais próximo (ver Tabela 5.7).

Tabela 5.6: Resultados para o PRVEF-FI nas instâncias de Belenguer et al. [14]

			Aleman and Hill[d]			SplitILS							
Problem	LB	BKS	K	Best Sol.	Time (s)	Avg. K	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
eil22	-	375,28[b]	4	**375,28**	2,58	4,00	**375,28**	**375,28**	-	0,00	0,15	0,00	375,28
eil23	451,80	569,75[b]	3	**569,75**	1,59	3,00	**568,56**	**568,56**	25,84	-0,21	0,13	0,00	568,56
eil30	218,92	505,01[b]	4	**505,01**	7,45	4,00	**505,01**	**505,01**	130,68	0,00	0,24	0,00	505,01
eil33	-	843,64[b]	4	**843,64**	8,38	4,00	**837,06**	**837,06**	-	-0,78	0,51	0,00	837,06
eil51	518,23	527,67[b]	5	**527,67**	49,84	5,00	**524,61**	**524,61**	1,23	-0,58	1,79	0,16	524,61
eilA76	809,58	832,71[a]	10	853,20	145,78	10,00	**823,89**	**824,92**	1,89	-0,94	30,76	0,52	823,89
eilB76	984,13	1034,21[b]	14	**1034,21**	91,36	14,00	**1009,04**	**1012,07**	2,84	-2,14	51,83	0,16	1009,04
eilC76	721,39	761,55[b]	8	**761,55**	151,13	8,00	**738,67**	**739,89**	2,56	-2,84	16,96	0,02	738,67
eilD76	672,34	695,96[b]	7	**695,96**	122,52	7,00	**687,60**	**689,36**	2,53	-0,95	11,16	0,04	687,60
eilA101	804,27	844,21[b]	8	**844,21**	295,22	8,00	**826,14**	**826,58**	2,77	-2,09	38,90	0,04	826,14
eilB101	1055,59	1112,15[b]	14	**1112,15**	173,13	14,00	**1076,26**	**1079,15**	2,23	-2,97	110,61	0,16	1076,26
S51D1	457,08	459,50[a]	3	468,79	13,56	3,00	**459,50**	**459,50**	0,53	0,00	1,24	0,01	459,50
S51D2	697,00	717,18[a]	9	718,69	31,66	9,00	**709,29**	**709,49**	1,79	-1,07	11,20	0,03	708,42
S51D3	933,97	960,38[a]	15	969,78	18,75	15,00	**948,06**	**950,12**	1,73	-1,07	15,74	0,03	947,97
S51D4	1545,19	1569,92[a]	27	1628,20	19,77	27,60	**1562,01**	**1563,29**	1,17	-0,42	56,28	2,23	1561,34
S51D5	1316,93	1339,38[a]	23	1362,19	15,39	23,00	**1333,67**	**1333,67**	1,27	-0,43	36,69	1,36	1333,67
S51D6	2149,55	2182,13[a]	41	2236,16	14,38	41,60	**2169,10**	**2177,78**	1,31	-0,20	62,55	0,19	2169,10
S76D1	590,92	613,70[b]	4	**613,70**	252,28	4,00	**598,94**	**598,94**	1,36	-2,41	4,86	0,02	598,94
S76D2	1066,88	1104,56[a]	15	1128,15	60,44	15,00	**1087,40**	**1089,45**	2,12	-1,37	69,36	0,94	1087,40
S76D3	1406,85	1435,10[a]	23	1472,92	51,13	23,00	**1427,86**	**1429,26**	1,59	-0,41	96,50	23,13	1427,81
S76D4	2053,66	2104,87[c]	37	2180,13	53,56	37,00	**2079,76**	**2081,16**	1,34	-1,13	188,38	0,32	2079,76
S101D1	714,50	749,93[b]	5	**749,93**	860,31	5,00	**726,59**	**728,45**	1,95	-2,86	15,93	0,27	726,59
S101D2	1356,78	1397,38[c]	20	1409,03	219,52	20,00	**1378,43**	**1386,03**	2,16	-0,81	151,66	2,67	1378,43
S101D3	1845,07	1911,40[a]	31	1947,62	132,19	31,00	**1874,81**	**1880,62**	1,93	-1,61	317,29	0,32	1874,81
S101D5	2758,21	2824,16[a]	48	2910,71	131,16	49,00	**2791,22**	**2795,36**	1,35	-1,02	572,13	2,18	2790,70
Média					116,92				8,44	-1,13	74,51	1,39	

a - Archetti et al. [5]
b - Aleman et al. [2]
c - Gulczynski et al. [37], Pentium 4, 3,0 GHz, uma única execução
d - Pentium 4, 2,8 GHz, uma única execução, Avg. cTime (s): 48,94

A Tabela 5.8 mostra a superioridade do SplitILS, especialmente em termos de qualidade de solução, no conjunto de instâncias de Campos et al. [19], quando comparado aos algoritmos de Boudia et al. [17] e Aleman et al. [2]. O algoritmo proposto foi capaz de encontrar soluções melhores em 40 problemas-teste e soluções com o mesmo custo nas outras nove, com o *gap* médio, entre a média das soluções e as melhores soluções conhecidas, de -0.41%. Em relação ao esforço computacional, o SplitILS consumiu, na média, 63.55 segundos para igualar ou melhorar as soluções da literatura.

Tabela 5.7: Resultados para o PRVEF-FI nas instâncias de Belenguer et al. [14] usando custos arredondados

		Boudia et al.[c]		Aleman and Hill[d]			SplitILS						
Problem	BKS	Best Sol.	Time (s)	K	Best Sol.	Time (s)	Avg. K	Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
eil22	375[a]	**375**	4,11	4	**375**	2,58	4,00	**375**	**375,00**	0,00	0,15	0,00	375
eil23	569[a]	**569**	5,47	3	570	1,59	3,00	**569**	**569,00**	0,00	0,11	0,00	569
eil30	503[a]	**503**	5,70	4	**503**	7,45	4,00	**503**	**503,00**	0,00	0,23	0,00	503
eil33	835[a]	**835**	5,19	4	844	8,38	4,00	**835**	**835,00**	0,00	0,45	0,01	835
eil51	521[a]	**521**	7,28	5	526	49,84	5,00	**521**	**521,00**	0,00	1,75	0,10	521
eilA76	818[a]	828	35,94	10	847	145,78	10,00	**818**	821,75	0,46	24,63	11,69	818
eilB76	1007[a]	1019	13,09	14	1027	91,36	14,00	**1002**	1007,05	0,00	37,68	25,37	1002
eilC76	733[a]	738	14,75	8	754	151,13	8,00	**733**	733,75	0,10	14,75	6,24	732
eilD76	682[a]	**682**	23,12	7	691	122,52	7,00	**682**	683,05	0,15	10,39	6,43	681
eilA101	815[a]	818	25,25	8	834	295,22	8,00	**814**	816,20	0,15	32,61	20,95	814
eilB101	1082[a]	**1082**	21,81	14	1104	173,13	14,00	**1061**	**1064,00**	-1,66	78,42	5,54	1061
S51D1	458[a]	**458**	8,77	3	465	13,56	3,00	**458**	**458,00**	0,00	1,17	0,05	458
S51D2	706[a]	707	7,44	9	715	31,66	9,00	**703**	**704,75**	-0,18	8,12	0,79	703
S51D3	945[a]	**945**	7,84	15	966	18,75	15,00	**943**	**944,05**	-0,10	13,06	0,81	943
S51D4	1578[a]	**1578**	11,98	27	1621	19,77	27,35	**1553**	**1556,50**	-1,36	39,25	0,04	1552
S51D5	1336[a]	1351	16,72	23	1357	15,39	23,00	**1328**	**1329,25**	-0,51	32,07	1,75	1328
S51D6	2177[a]	2182	9,92	41	2228	14,38	41,15	**2163**	**2166,15**	-0,50	52,95	0,01	2163
S76D1	592[a]	**592**	15,23	4	606	252,28	4,00	**592**	592,30	0,05	4,75	2,06	592
S76D2	1087[a]	1089	30,50	15	1124	60,44	15,00	**1082**	**1083,15**	-0,35	53,60	15,28	1081
S76D3	1420[a]	1427	12,89	23	1466	51,13	23,00	**1420**	1423,05	0,21	67,81	56,14	1419
S76D4	2094[a]	2117	8,76	37	2170	53,56	37,00	**2073**	**2074,95**	-0,91	144,89	7,53	2070
S101D1	716[a]	717	49,75	5	741	860,31	5,00	**716**	718,35	0,33	14,76	12,13	716
S101D2	1372[a]	**1372**	31,72	20	1398	219,52	20,00	**1366**	**1371,40**	-0,04	112,47	60,90	1364
S101D3	1891[a]	**1891**	33,98	31	1936	132,19	31,00	**1864**	**1868,05**	-1,21	236,05	2,91	1860
S101D5	2854[a]	**2854**	18,66	48	2897	131,16	49,00	**2770**	**2779,10**	-2,62	439,49	0,11	2769
Média			17,03			116,92				-0,32	56,87	9,47	

a - Boudia et al. [17]
b - Aleman et al. [2]
c - PC 3,0 GHz, uma única execução, Avg. cTime (s): 9,28
d - Pentium 4, 2,8 GHz, uma única execução, Avg. cTime (s): 48,94

A Tabela 5.9 ilustra os resultados encontrados pelo SplitILS no conjunto de instâncias gerado por Archetti et al. [13]. Pode-se observar que o algoritmo desenvolvido não consegue encontrar uma solução equivalente a da literatura apenas na instância p11-120D5. Apesar disso, verifica-se que uma solução de melhor qualidade foi encontrada quando leva-se em conta todos os experimentos realizados. Para os problemas-teste remanescentes, o SplitILS claramente obteve resultados superiores, na média, àqueles encontrados por Derigs et al. [25]. O *gap* médio entre a média das soluções do SplitILS e as melhores soluções da literatura foi de -1.22%. Soluções de custo igual ou superior foram alcançadas em poucos segundos, na maioria dos casos.

No conjunto de instâncias de Chen et al. [21], o algoritmo proposto alcançou soluções de mesmo custo em cinco problemas-teste, enquanto foi capaz de superar as soluções da literatura em outras 6, como pode ser verificado na Tabela 5.10. Apesar do SplitILS não ter sido capaz de igualar ou melhorar as soluções em 10 problemas-teste, o *gap* médio entre a média das soluções e o limite inferior foi menor que 1%, o que é significativo em termos de qualidade de solução. Comparando os resultados obtidos com aqueles reportados por

Tabela 5.8: Resultados para o PRVEF-FI nas instâncias de Campos et al. [19]

		Boudia et al.[c]		Aleman and Hill[d]			SplitILS						
Problem	BKS	Best Sol.	Time (s)	K	Best Sol.	Time (s)	Avg. K	Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
p01-50	524,61[a]	**524,61**	8,53	5	527,67	49,84	5,00	**524,61**	**524,61**	0,00	1,82	0,28	524,61
p02-75	823,89[a]	**823,89**	35,72	11	853,20	145,78	10,00	**823,89**	824,80	0,11	30,52	10,20	823,89
p03-100	827,39[a]	829,44	34,59	8	844,21	295,22	8,00	**826,14**	**826,39**	-0,12	40,01	3,09	826,14
p04-150	1025,49[a]	1042,37	103,69	12	1079,55	2217,67	12,00	**1023,66**	1026,89	0,14	250,37	198,64	1023,66
p05-199	1293,00[a]	1311,59	353,84	17	1339,49	4514,28	16,05	**1286,92**	1293,71	0,05	1298,10	1040,29	1285,55
p06-120	1037,88[a]	1041,20	50,92	7	1051,24	1944,19	7,00	**1037,88**	1039,13	0,12	81,50	51,32	1037,88
p07-100	819,56[a]	**819,56**	42,89	10	819,60	75,33	10,00	**819,56**	**819,56**	0,00	27,67	0,13	819,56
p01-50D1	460,79[a]	**460,79**	12,38	3	466,74	19,69	3,00	**460,79**	**460,79**	0,00	1,17	0,04	460,79
p02-75D1	596,25[a]	600,06	18,75	4	614,09	136,14	4,00	**596,25**	**596,25**	0,00	4,98	0,15	596,25
p03-100D1	726,81[a]	**726,81**	37,12	5	741,60	1944,09	5,00	**726,81**	730,80	0,55	16,67	14,08	726,81
p04-150D1	866,31[a]	875,61	100,27	8	891,10	2640,95	8,00	**866,31**	**866,31**	0,00	120,92	18,68	866,31
p05-199D1	1018,38[a]	1018,71	356,22	10	1069,24	11215,52	10,00	**1017,28**	1018,59	0,02	431,97	261,10	1017,28
p06-120D1	976,57[a]	**976,57**	72,98	6	990,59	2736,34	6,00	**975,96**	**976,57**	0,00	44,82	13,57	975,96
p07-100D1	634,57[a]	649,73	34,97	6	658,99	461,75	5,15	**632,63**	636,76	0,35	12,28	7,29	632,63
p01-50D2	741,06[a]	751,41	10,22	10	753,98	23,17	10,00	**741,06**	741,26	0,03	9,72	5,86	741,06
p02-75D2	1067,80[a]	1074,46	34,14	15	1085,70	97,17	15,00	**1064,49**	**1066,36**	-0,13	53,25	23,82	1064,49
p03-100D2	1377,28[a]	1392,85	78,06	20	1416,35	160,95	20,00	**1376,22**	1380,23	0,21	187,76	173,95	1374,15
p04-150D2	1875,09[a]	1878,71	147,89	30	1929,91	755,08	29,00	**1861,63**	**1866,95**	-0,43	1041,64	147,76	1861,51
p05-199D2	2329,37[a]	2340,14	347,14	39	2408,16	1544,36	38,00	**2305,70**	**2313,04**	-0,70	2296,08	445,68	2305,70
p06-120D2	2720,35[a]	2720,38	144,19	24	2744,74	463,97	23,00	**2707,52**	**2710,15**	-0,37	704,44	25,93	2703,88
p07-100D2	1415,78[a]	1417,28	43,27	-	1441,48	98,31	20,00	**1413,85**	**1413,99**	-0,13	128,52	1,08	1413,85
p01-50D3	988,31[a]	**988,31**	12,49	15	1023,24	17,72	15,00	**982,79**	**983,59**	-0,48	18,04	0,21	982,77
p02-75D3	1398,53[a]	1413,80	37,38	23	1458,59	67,66	22,00	**1393,11**	**1393,11**	-0,39	97,13	12,47	1393,11
p03-100D3	1827,65[a]	1845,30	28,39	29	1886,70	145,05	29,00	**1823,58**	1827,81	0,01	313,83	123,39	1823,41
p04-150D3	2539,75[a]	2561,65	224,89	45	2647,17	470,34	43,00	**2527,96**	**2531,50**	-0,32	1445,25	256,58	2526,04
p05-199D3	3180,30[a]	3191,25	436,20	56	3296,69	1216,69	56,15	**3156,02**	**3163,26**	-0,54	3316,93	131,70	3155,85
p06-120D3	3934,39[a]	**3934,39**	163,14	35	4010,80	340,53	34,00	**3907,27**	**3909,28**	-0,64	1487,97	32,80	3907,27
p07-100D3	1994,59[a]	**1994,59**	51,31	30	2010,00	84,50	29,00	**1967,41**	**1968,09**	-1,33	265,71	0,37	1967,41
p01-50D4	1467,06[a]	**1467,06**	21,42	26	1530,81	19,11	25,00	**1456,00**	**1457,37**	-0,66	42,86	0,64	1456,00
p02-75D4	2087,22[a]	2102,58	46,11	39	2164,74	61,81	37,05	**2081,38**	**2084,91**	-0,11	191,06	47,31	2076,41
p03-100D4	2780,95[a]	**2780,95**	84,38	48	2874,86	125,28	48,00	**2749,53**	**2753,99**	-0,97	647,44	2,39	2749,53
p04-150D4	4045,87[a]	**4045,87**	244,91	74	4151,90	451,95	73,15	**3988,64**	**3996,55**	-1,22	1901,05	4,31	3987,58
p05-199D4	4941,22[a]	**4941,22**	725,69	93	5066,24	108,63	93,00	**4843,83**	**4855,49**	-1,73	3739,98	3,38	4843,83
p06-120D4	6308,76[b]	6318,37	196,14	56	**6308,76**	418,98	56,00	**6195,37**	**6219,01**	-1,42	1805,91	1,97	6195,37
p07-100D4	3113,72[a]	**3113,72**	52,13	48	3157,48	97,58	48,00	**3088,47**	**3089,41**	-0,78	416,34	0,22	3088,23
p01-50D5	1477,01[a]	**1477,01**	24,53	26	1505,38	19,09	25,00	**1467,47**	**1467,47**	-0,65	49,42	2,16	1467,47
p02-75D5	2132,16[a]	**2132,16**	51,78	38	2182,33	55,17	37,95	**2111,83**	**2114,03**	-0,85	212,38	6,17	2110,28
p03-100D5	2858,87[a]	**2858,87**	100,16	49	2929,29	134,84	49,00	**2813,52**	**2817,05**	-1,46	737,91	1,63	2813,37
p04-150D5	4045,87[a]	**4045,87**	244,86	74	4151,90	449,34	73,05	**3985,76**	**3995,25**	-1,25	1836,14	3,54	3985,76
p05-199D5	5155,36[a]	**5155,36**	749,94	96	5281,55	119,04	96,00	**5063,89**	**5072,74**	-1,60	4222,85	3,52	5056,93
p06-120D5	6399,42[a]	6424,71	271,39	58	6511,08	436,80	58,00	**6373,24**	**6376,25**	-0,36	2303,70	16,18	6372,92
p07-100D5	3155,69[a]	**3155,69**	91,31	50	3200,62	96,39	49,00	**3125,47**	**3125,98**	-0,94	568,97	0,61	3125,35
p01-50D6	2154,35[a]	**2154,35**	22,91	41	2219,32	24,41	41,00	**2150,97**	**2152,95**	-0,06	51,94	5,02	2150,97
p02-75D6	3200,35[a]	**3200,35**	27,48	60	3278,33	86,27	60,25	**3178,47**	**3181,28**	-0,60	412,86	0,59	3178,47
p03-100D6	4312,95[a]	**4312,95**	55,75	80	4435,56	185,55	80,00	**4294,12**	**4299,40**	-0,31	737,85	0,03	4294,10
p04-150D6	6267,48[a]	**6267,48**	401,62	119	6416,12	678,94	119,00	**6232,37**	**6234,56**	-0,53	1543,69	0,09	6229,85
p05-199D6	8081,58[a]	**8081,58**	571,70	158	8333,61	153,12	158,00	**8037,88**	**8048,57**	-0,41	4616,79	3,46	8037,88
p06-120D6	10017,47[a]	10063,47	298,08	95	10186,06	30,32	95,00	**10003,99**	**10005,29**	-0,12	2161,59	1,91	10001,12
p07-100D6	4919,48[a]	**4919,48**	180,11	80	4996,88	152,92	80,00	**4903,00**	**4906,56**	-0,26	799,40	8,30	4902,73
Média			152,73			771,18				-0,41	872,02	63,55	

a - Boudia et al. [17]
b - Aleman et al. [2]
c - PC 3,0 GHz, uma única execução, Avg. cTime (s): 83,21
d - Pentium 4, 2,8 GHz, uma única execução, Avg. cTime (s): 322,79

Tabela 5.9: Resultados para o PRVEF-FI nas instâncias de Archetti et al. [13]

			Derigs et al.[c]		SplitILS							
Problem	LB	BKS	K	Best Sol.	Avg. K	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
p01-50	-	524,61[b]	5	**524,61**	5,00	**524,61**	**524,61**	-	0,00	1,88	0,48	524,61
p02-75	-	823,89[b]	10	829,89	10,00	**823,89**	825,06	-	0,14	29,69	14,04	823,89
p03-100	-	826,14[b]	8	**826,14**	8,00	**826,14**	826,45	-	0,04	40,52	12,91	826,14
p04-150	-	1028,42[b]	12	**1028,42**	12,00	**1023,87**	**1027,28**	-	-0,11	233,73	59,28	1023,66
p05-199	-	1296,66[b]	16	1302,89	16,00	**1289,89**	**1293,24**	-	-0,26	1355,25	564,00	1283,27
p11-120	-	1042,12[b]	7	**1042,12**	7,00	**1037,88**	1043,38	-	0,12	84,84	9,02	1037,88
p01-50D1	456,63	459,50[a]	-	-	3,00	**459,50**	**459,50**	0,63	0,00	1,16	0,03	459,50
p02-75D1	604,70	652,93[a]	-	-	5,00	**617,85**	**619,59**	2,46	-5,11	5,70	0,00	617,85
p03-100D1	743,06	788,23[a]	-	-	6,00	**760,00**	**760,46**	2,34	-3,52	21,65	0,09	760,00
p04-150D1	895,46	984,69[a]	-	-	9,00	**921,91**	**923,69**	3,15	-6,19	161,90	0,04	921,47
p05-199D1	1042,37	1268,79[a]	-	-	12,00	**1074,18**	**1080,64**	3,67	-14,83	626,12	0,04	1074,18
p11-120D1	1023,39	1071,58[a]	-	-	8,00	**1043,19**	**1043,22**	1,94	-2,65	109,76	0,14	1042,89
p01-50D2	750,45	758,20[a]	11	776,42	11,00	**757,15**	761,12	1,42	0,39	13,24	10,87	756,71
p02-75D2	1095,65	1116,75[b]	16	1123,97	16,00	**1109,62**	**1112,11**	1,50	-0,42	53,26	8,69	1109,62
p03-100D2	1437,78	1472,53[b]	22	1478,59	22,00	**1458,46**	**1462,68**	1,73	-0,67	179,69	18,52	1458,46
p04-150D2	1986,34	2037,00[b]	32	2055,18	32,00	**2016,97**	**2021,36**	1,76	-0,77	1109,88	98,57	2016,93
p05-199D2	2423,99	2528,82[b]	41	2540,06	41,05	**2478,40**	**2486,54**	2,58	-1,67	2661,25	4,76	2477,44
p11-120D2	2867,79	2907,39[b]	26	2913,09	26,00	**2898,50**	**2907,07**	1,37	-0,01	895,11	413,06	2898,46
p01-50D3	996,72	1007,51[b]	16	1012,56	16,00	**1005,75**	**1005,75**	0,91	-0,17	20,70	2,39	1005,75
p02-75D3	1482,50	1504,74[b]	24	1508,73	24,00	**1502,05**	**1503,57**	1,42	-0,08	108,37	26,32	1502,05
p03-100D3	1971,34	2018,94[b]	33	2035,91	33,00	**1996,76**	**2002,23**	1,57	-0,83	362,02	6,92	1996,76
p04-150D3	2811,98	2901,62[b]	49	2912,08	49,00	**2849,66**	**2857,28**	1,61	-1,53	1518,82	3,72	2849,66
p05-199D3	3420,23	3548,31[b]	63	3581,66	63,00	**3471,41**	**3480,76**	1,77	-1,90	3014,82	6,03	3471,33
p11-120D3	4156,68	4259,94[a]	40	4270,38	40,00	**4219,01**	**4220,79**	1,54	-0,92	1957,37	83,32	4218,50
p01-50D4	1471,54	1488,58[b]	27	1489,64	27,00	**1488,58**	1488,89	1,18	0,02	43,04	28,41	1488,40
p02-75D4	2272,05	2318,28[a]	41	2340,09	41,00	**2298,58**	**2301,85**	1,31	-0,71	207,22	2,60	2298,33
p03-100D4	3042,93	3116,61[b]	56	3145,33	56,00	**3085,69**	**3093,64**	1,67	-0,74	736,52	8,56	3085,69
p04-150D4	4474,92	4581,32[b]	84	4638,74	83,05	**4545,46**	**4550,85**	1,70	-0,67	2410,48	60,96	4543,23
p05-199D4	5422,95	5669,26[b]	105	**5669,26**	106,00	**5521,57**	**5529,06**	1,96	-2,47	4349,61	0,33	5520,69
p11-120D4	6780,19	6881,04[b]	67	6890,39	67,00	**6854,09**	**6865,23**	1,25	-0,23	3442,31	159,93	6854,03
p01-50D5	1466,34	1487,81[b]	26	1488,28	26,00	**1481,71**	**1483,36**	1,16	-0,30	44,11	6,69	1481,71
p02-75D5	2195,44	2228,69[b]	39	2243,93	39,00	**2219,97**	**2224,06**	1,30	-0,21	265,00	85,83	2219,52
p03-100D5	2945,42	3002,64[b]	53	3014,08	53,00	**2989,30**	**2992,75**	1,61	-0,33	742,88	55,01	2988,38
p04-150D5	4267,33	4374,56[b]	79	4435,95	79,00	**4334,71**	**4341,15**	1,73	-0,76	2357,52	19,48	4334,71
p05-199D5	5304,09	5487,55[b]	102	5541,09	102,95	**5409,76**	**5417,75**	2,14	-1,27	4524,33	5,43	5403,35
p11-120D5	6593,28	6658,52[b]	64	6671,04	64,00	6673,95	6678,11	1,29	0,29	2354,02	-	6652,64
p01-50D6	2134,96	2160,66[b]	41	2174,54	41,00	**2156,14**	2161,31	1,23	0,03	78,49	39,87	2155,80
p02-75D6	3177,20	3234,64[b]	61	3266,78	61,00	**3223,40**	**3226,20**	1,54	-0,26	378,95	3,86	3223,40
p03-100D6	4319,16	4411,32[b]	82	4447,47	82,00	**4387,32**	**4389,43**	1,63	-0,50	675,01	0,08	4387,32
p04-150D6	6284,76	6467,17[b]	122	**6467,17**	122,85	**6395,41**	**6402,15**	1,87	-0,94	1926,20	0,09	6393,22
p05-199D6	8062,14	8297,71[b]	161	**8297,71**	161,00	**8192,03**	**8195,67**	1,66	-1,23	3258,29	0,22	8188,15
p11-120D6	10113,55	10233,37[b]	98	**10233,37**	98,00	**10204,81**	**10216,80**	1,02	-0,16	2279,57	13,91	10204,79
Média								1,68	-1,22	1062,86		

a - Archetti et al. [5]
b - Derigs et al. [25]
c - PC 3,0 GHz, 3600 segundos para todas as instâncias (cTime (s): 1944)
c - Melhor de 5 execuções usando a heurística ABHC

Aleman et al. [2], nota-se que o SplitILS não é capaz de encontrar uma solução igual ou superior apenas no problema SD17. Embora não esteja explicitamente mostrado na Tabela 5.10, o SplitILS na realidade consome, em média, 0.8 segundos para igualar ou superar as melhores soluções de Aleman et al. [2], excluindo as instâncias SD12 e SD17.

Tabela 5.10: Resultados para o PRVEF-FI nas instâncias de Chen et al. [21]

			Aleman and Hill[d]			SplitILS							
Problem	LB	BKS	K	Best Sol.	Time (s)	Avg. K	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
SD1	228,28	*228,28[a]	6	**228,28**	0,00	6,00	**228,28**	**228,28**	0,00	0,00	0,05	0,00	228,28
SD2	708,28	*708,28[a]	12	**708,28**	0,02	12,00	**708,28**	**708,28**	0,00	0,00	0,63	0,02	708,28
SD3	430,40	*430,40[a]	12	430,58	0,03	12,00	430,58	430,58	0,04	0,04	0,62	-	430,58
SD4	630,62	*630,62[a]	18	631,05	0,08	18,00	631,05	631,05	0,07	0,07	2,26	-	631,05
SD5	1373,25	1389,94[a]	24	1390,57	0,13	24,00	1390,57	1390,57	1,26	0,05	6,07	-	1390,57
SD6	830,86	*830,86[a]	24	831,24	0,14	24,00	831,24	831,24	0,05	0,05	5,81	-	831,24
SD7	3638,45	3640,00[a]	30	**3640,00**	0,09	30,00	**3640,00**	**3640,00**	0,04	0,00	14,12	0,00	3640,00
SD8	5068,28	*5068,28[a]	36	**5068,28**	0,14	36,00	**5068,28**	**5068,28**	0,00	0,00	24,93	0,01	5068,28
SD9	2028,08	2042,88[a]	36	2071,03	0,36	36,00	2044,20	2044,20	0,79	0,06	38,78	-	2044,20
SD10	2678,83	2683,73[a]	48	2747,83	0,89	48,00	2684,88	2684,88	0,23	0,04	101,10	-	2684,88
SD11	13111,11	13280,00[a]	60	**13280,00**	0,41	60,00	**13280,00**	**13280,00**	1,29	0,00	152,42	0,02	13280,00
SD12	7089,27	7213,62[b]	60	**7213,62**	0,84	60,00	**7213,61**	7216,60	1,80	0,04	210,71	160,60	7213,61
SD13	9969,93	10105,86[a]	72	10110,58	1,20	72,00	10110,58	10110,58	1,41	0,05	189,45	-	10110,58
SD14	10502,76	10754,70[a]	90	10802,87	2,31	90,00	**10717,53**	**10723,79**	2,10	-0,29	479,85	0,04	10715,52
SD15	14787,05	15152,88[c]	108	15153,45	3,20	108,00	**15094,48**	**15105,90**	2,16	-0,31	731,98	0,06	15093,85
SD16	3379,33	*3379,33[a]	108	3446,43	7,59	108,00	3381,26	3394,48	0,45	0,45	930,72	-	3381,25
SD17	26166,80	26493,56[b]	120	**26493,56**	7,27	120,15	26496,06	26499,32	1,27	0,02	577,29	-	26496,06
SD18	13892,74	14323,04[b]	120	**14323,04**	27,95	120,25	**14202,53**	**14205,07**	2,25	-0,82	834,60	0,10	14199,22
SD19	19584,84	20157,10[b]	144	**20157,10**	11,95	144,55	**19995,69**	**20007,52**	2,16	-0,74	1524,67	0,16	19991,18
SD20	38901,37	39722,86[b]	180	**39722,86**	11,02	180,00	**39635,51**	**39647,61**	1,92	-0,19	1563,38	0,32	39635,51
SD21	11254,83	11271,06[c]	216	11458,76	111,56	216,35	11345,68	11365,37	0,98	0,84	5034,56	-	11345,68
Média					8,91				0,96	-0,03	591,62	14,67	

∗ - Solução ótima
a - Archetti et al. [5]
b - Aleman et al. [2]
c - Derigs et al. [25], PC 3,0 GHz, uma única execução
d - Pentium 4, 2,8 GHz, uma única execução, Avg. cTime (s): 3,73

5.3 Resumo dos Resultados

O resumo dos resultados obtidos pelo SplitILS são mostrados na Tabela 5.11. Nesta, #**Instâncias** indica o número de problemas-teste que compõem cada um dos conjuntos de instâncias, #**Melhores** denota o número de problemas nos quais encontrou-se soluções de melhor qualidade que aquelas da literatura e #**Empates** representa o número de empates. Pode-se observar que o SplitILS foi capaz de melhorar os custos em 243 casos, o que representa 75% das instâncias, e de igualar os resultados em outras 55.

Tabela 5.11: Resumo dos resultados obtidos pelo SplitILS

Conjunto	#Instâncias	#Melhores	#Empates	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)
Belenguer [14][a,c]	25	18	7	−1,87	63,77	1,50
Belenguer [14][a]	25	22	3	−1,46	75,97	1,69
Campos [19][a]	49	44	5	−2,23	925,59	3,06
Archetti [13][a]	42	36	-	−2,81	1159,19	2,68
Chen [21][a]	21	7	5	−0,02	504,44	16,41*
Belenguer [14][b,c]	25	13	12	−0,32	56,87	9,47
Belenguer [14][b]	25	22	3	−1,13	74,51	1,39
Campos [19][b]	49	40	9	−0,41	872,02	63,55
Archetti [13][b]	42	36	5	−1,22	1062,86	44,74*
Chen [21][b]	21	5	6	−0,03	591,62	14,67*
Total	324	243	55			

a - PRVEF-FL
b - PRVEF-FI
c - Custos arredondados
* - Desconsiderando as instâncias em que o SplitILS não foi capaz de encontrar ou melhorar a BKS

A partir dos resultados obtidos, pode-se observar que a heurística proposta possui boa acurácia mesmo possuindo uma estrutura simples, visto que é capaz de gerar soluções de boa qualidade dependendo de poucos parâmetros. Estas são duas das qualidades apontadas por muitos autores como essenciais a uma heurística. Silver et al. [55], por exemplo, consideram que uma boa heurística deve possuir as seguintes propriedades, (i) Esforço computacional realista na obtenção de uma solução; (ii) A solução deve, na média, estar próxima à solução ótima (desempenho médio deve ser bom); (iii) A chance de obter uma solução muito ruim (longe do ótimo) deve ser baixa; e (iv) a heurística deve ser simples de entender e intuitiva. Tais propriedades estão de acordo com aquelas apontadas por Cordeau et al. [23].

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho apresentou uma abordagem heurística para tratar o Problema de Roteamento de Veículos com Entregas Fracionárias considerando dois esquemas de dimensionamento de frota, limitada e ilimitada. O algoritmo proposto, chamado SplitILS, é baseado na meta-heurística *Iterated Local Search* (ILS) e no procedimento *Randomized Variable Neighborhood Descent* (RVND). Experimentos computacionais foram realizados em quatro conjuntos de instâncias da literatura, totalizando 324 problemas-teste, e na maioria dos casos, acima de 90%, o SplitILS foi capaz de igualar ou encontrar soluções de qualidade superior àquelas reportadas na literatura em uma questão de segundos. Além disso, é válido mencionar que o SplitILS depende basicamente de dois parâmetros, mais precisamente o número de reinicializações e o número de iterações sem melhora, o que corresponde ao critério de parada do ILS.

Os procedimentos adotados nas fases de construção, busca local e perturbação, descritos na metodologia deste trabalho e incorporados ao algoritmo final, foram aqueles que apresentaram, em média, melhor desempenho em relação a qualidade das soluções geradas e tempo computacional. Contudo, durante os estudos várias outras abordagens foram desenvolvidas para cada uma dessas fases. Na construção de soluções iniciais por meio de heurísticas de inserção, o fracionamento de demandas na versão apresentada é aplicado somente após todos clientes terem sido roteados. Outras versões que aplicavam o fracionamento durante a inserção também foram testados. Além das heurísticas de inserção, foram implementados vários procedimentos baseados na heurística das economias de Clarke e Wright. Nos mecanismos de perturbação, um procedimento *EjectionChain* adaptado para tratar o fracionamento de demandas foi testado, mas este se mostrou pouco eficaz. Também foram implementadas versões modificadas dos mecanismos adotados no algoritmo final. Na busca local, versões generalizadas da vizinhança *RouteAddition* foram

estudadas, a vizinhança *Cross* foi adaptada para tratar fracionamento de demandas. Estas não apresentaram ganhos significativos. Em relação ao esforço computacional do algoritmo, observou-se que este é influenciado principalmente pelo número de clientes e pela quantidade de divisões das demandas.

Como trabalhos futuros, pretende-se estender o SplitILS para tratar outras variantes do PRVEF que incluem características adicionais como janelas de tempo, frota heterogênea e coleta e entrega. Além disso, abordagens híbridas poderiam ser estudadas. Por exemplo, a partir de uma coleção de rotas armazenadas durante a execução do algoritmo proposto, um procedimento exato baseado em uma formulação de Cobertura de Conjuntos poderia ser utilizado para selecionar o subconjunto de rotas de menor custo. Outra possibilidade seria armazenar as arestas mais utilizadas e usar esta informação para eliminar arestas no grafo original, gerando assim um grafo reduzido, e a partir desse, executar um modelo exato. Tais abordagens podem ser vistas como mecanismos de memória de longo prazo. O uso de mecanismos de memória, tanto de longo quanto de curto prazo, podem ajudar a melhorar o desempenho do algoritmo, visto que de uma iteração para outra da heurística *multi-start*, nenhuma informação, além da melhor solução encontrada, é armazenada.

Referências

[1] Aleman, R., Zhang, X., Hill, R. An adaptive memory algorithm for the split delivery vehicle routing problem. *Journal of Heuristics 16*, 3 (2010), 441–473.

[2] Aleman, R. E., Hill, R. R. A tabu search with vocabulary building approach for the vehicle routing problem with split demands. *International Journal of Metaheuristics 1*, 1 (2010), 55–80.

[3] Aleman, R. E., Zhang, X., Hill, R. R. A ring-based diversification scheme for routing problems. *International Journal of Mathematics in Operational Research 1*, 1 (2009), 163–190.

[4] Ambrosino, D., Sciomachen, A. A food distribution network problem: a case study. *IMA Journal of Management Mathematics 18*, 1 (2007), 33–53.

[5] Archetti, C., Bianchessi, N., Speranza, M. G. A column generation approach for the split delivery vehicle routing problem. *Networks 58*, 4 (2011), 241–254.

[6] Archetti, C., Bouchard, M., Desaulniers, G. Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. *Transportation Science 45*, 3 (2011), 285–298.

[7] Archetti, C., Feillet, D., Gendreau, M., Speranza, M. G. Complexity of the VRP and SDVRP. *Transportation Research Part C: Emerging Technologies 19*, 5 (2011), 741 – 750.

[8] Archetti, C., Mansini, R., Speranza, M. G. Complexity and reducibility of the skip delivery problem. *Transportation Science 39*, 2 (2005), 182–187.

[9] Archetti, C., Savelsbergh, M. W., Speranza, M. G. To split or not to split: That is the question. *Transportation Research Part E: Logistics and Transportation Review 44*, 1 (2008), 114 – 123.

[10] Archetti, C., Savelsbergh, M. W. P., Speranza, M. G. Worst-case analysis for split delivery vehicle routing problems. *Transportation Science 40*, 2 (2006), 226–234.

[11] Archetti, C., Speranza, M. G. Vehicle routing problems with split deliveries. *International Transactions in Operational Research 19*, 1-2 (2012), 3–22.

[12] Archetti, C., Speranza, M. G., Hertz, A. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science 40*, 1 (2006), 64–73.

[13] Archetti, C., Speranza, M. G., Savelsbergh, M. W. P. An optimization-based heuristic for the split delivery vehicle routing problem. *Transportation Science 42*, 1 (2008), 22–31.

[14] Belenguer, J. M., Martinez, M. C., Mota, E. A lower bound for the split delivery vehicle routing problem. *Operations Research 48*, 5 (2000), 801–810.

[15] Belfiore, P., Yoshizaki, H. T. Y. Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in brazil. *European Journal of Operational Research 199*, 3 (2009), 750 – 758.

[16] Blum, C., Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys 35*, 3 (2003), 268–308.

[17] Boudia, M., Prins, C., Reghioui, M. An effective memetic algorithm with population management for the split delivery vehicle routing problem. In *Hybrid Metaheuristics*, T. Bartz-Beielstein, M. Blesa Aguilera, C. Blum, B. Naujoks, A. Roli, G. Rudolph, and M. Sampels, Eds., vol. 4771 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2007, p. 16–30.

[18] Bräysy, O., Gendreau, M. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science 39*, 1 (2005), 104–118.

[19] Campos, V., Corberán, A., Mota, E. A scatter search algorithm for the split delivery vehicle routing problem. In *Advances in Computational Intelligence in Transport, Logistics, and Supply Chain Management*, A. Fink and F. Rothlauf, Eds., vol. 144 of *Studies in Computational Intelligence*. Springer Berlin / Heidelberg, 2008, p. 137–152.

[20] Chao, I.-M., Golden, B. L., Wasil, E. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. *American Journal of Mathematical and Management Sciences 13*, 3-4 (1993), 371–406.

[21] Chen, S., Golden, B., Wasil, E. The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks 49*, 4 (2007), 318–329.

[22] Clarke, G., Wright, J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research 12*, 4 (1964), 568–581.

[23] Cordeau, J. F., Gendreau, M., Laporte, G., Potvin, J. Y., Semet, F. A guide to vehicle routing heuristics. *The Journal of the Operational Research Society 53*, 5 (2002), 512–522.

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

[25] Derigs, U., Li, B., Vogel, U. Local search-based metaheuristics for the split delivery vehicle routing problem. *Journal of Operational Research Society 61*, 9 (2010), 1356 – 1364.

[26] Desaulniers, G. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Operations Research 58*, 1 (2010), 179–192.

[27] Dongarra, J. J. Performance of various computers using standard linear equations software. Relatório Técnico CS-89-85, Computer Science Department, University of Tennessee, 2011.

[28] Dror, M., Laporte, G., Trudeau, P. Vehicle routing with split deliveries. *Discrete Applied Mathematics 50*, 3 (1994), 239–254.

[29] Dror, M., Trudeau, P. Savings by split delivery routing. *Transportation Science 23*, 2 (1989), 141–145.

[30] Dror, M., Trudeau, P. Split delivery routing. *Naval Research Logistics 37*, 3 (1990), 383–402.

[31] Frizzell, P. W., Giffin, J. The bounded split delivery vehicle routing problem with grid network distances. *Asia-Pacific Journal of Operational Research 9*, 1 (1992), 101–116.

[32] Frizzell, P. W., Giffin, J. W. The split delivery vehicle scheduling problem with time windows and grid network distances. *Computers & Operations Research 22*, 6 (1995), 655–667.

[33] Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., Werneck, R. F. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming 106* (2006), 491–511.

[34] Gendreau, M., Hertz, A., Laporte, G. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research 40*, 6 (1992), 1086–1094.

[35] Glover, F. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research 13*, 5 (1986), 533–549.

[36] Glover, F., Laguna, M., Marti, R. Scatter Search and Path Relinking: Advances and Applications Handbook of Metaheuristics. In *Handbook of Metaheuristics*, F. Glover and G. A. Kochenberger, Eds., vol. 57 of *International Series in Operations Research & Management Science*. Springer New York, Boston, 2003, cap. 1, p. 1–35.

[37] Gulczynski, D., Golden, B., Wasil, E. The split delivery vehicle routing problem with minimum delivery amounts. *Transportation Research Part E: Logistics and Transportation Review 46*, 5 (2010), 612 – 626.

[38] Gulczynski, D., Golden, B., Wasil, E. The multi-depot split delivery vehicle routing problem: An integer programming-based heuristic, new test problems, and computational results. *Computers & Industrial Engineering 61*, 3 (2011), 794 – 804.

[39] Hansen, P., Mladenović, N., Brimberg, J., Pérez, J. A. M. Variable neighborhood search. In *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Eds., vol. 146 of *International Series in Operations Research & Management Science*. Springer US, 2010, p. 61–86.

[40] Hansen, P., Mladenović, N., MorenoÂ Pérez, J. Variable neighbourhood search: methods andÂ applications. *Annals of Operations Research 175*, 1 (2010), 367–407.

[41] Ho, S., Haugland, D. A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. *Computers & Operations Research 31*, 12 (2004), 1947 – 1964.

[42] Holland, J. H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[43] Jin, M., Liu, K., Bowden, R. O. A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. *International Journal of Production Economics 105*, 1 (2007), 228 – 242.

[44] Jin, M., Liu, K., Eksioglu, B. A column generation approach for the split delivery vehicle routing problem. *Operations Research Letters 36*, 2 (2008), 265 – 270.

[45] Lourenço, H. R., Martin, O. C., Stützle, T. Iterated local search: Framework and applications. In *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Eds., vol. 146 of *International Series in Operations Research & Management Science*. Springer US, 2010, p. 363–397.

[46] Moreno, L., de Aragão, M. P., Uchoa, E. Improved lower bounds for the split delivery vehicle routing problem. *Operations Research Letters 38*, 4 (2010), 302 – 306.

[47] Nowak, M., Ergun, Ö., White III, C. C. Pickup and delivery with split loads. *Transportation Science 42*, 1 (2008), 32–43.

[48] Nowak, M., Ergun, Ö., White III, C. C. An empirical study on the benefit of split loads with the pickup and delivery problem. *European Journal of Operational Research 198*, 3 (2009), 734 – 740.

[49] Osman, I. H. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research 41*, 4 (1993), 421–451.

[50] Penna, P., Subramanian, A., Ochi, L. An Iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem. *Journal of Heuristics* (2011), 1–32. To appear.

[51] Reinelt, G. TSPLIB–A Traveling Salesman Problem Library. *INFORMS Journal on Computing 3*, 4 (1991), 376–384.

[52] Salani, M., Vacca, I. Branch and price for the vehicle routing problem with discrete split deliveries and time windows. *European Journal of Operational Research 213*, 3 (2011), 470 – 477.

[53] Sierksma, G., Tijssen, G. A. Routing helicopters for crew exchanges on off-shore locations. *Annals of Operations Research 76*, 0 (1998), 261–286.

[54] Silva, M. M., Subramanian, A., Vidal, T., Ochi, L. S. A simple and effective metaheuristic for the minimum latency problem. *European Journal of Operational Research 221*, 3 (2012), 513–520.

[55] Silver, E. A., Victor, R., Vidal, V., de Werra, D. A tutorial on heuristic methods. *European Journal of Operational Research 5*, 3 (1980), 153 – 162.

[56] Song, S. H., Lee, K. S., Kim, G. S. A practical approach to solving a newspaper logistics problem using a digital map. *Computers & Industrial Engineering 43*, 1-2 (2002), 315 – 330.

[57] Subramanian, A., Battarra, M. An iterated local search algorithm for the travelling salesman problem with pickups and deliveries. *Journal of Operational Research Society* (2012). To appear.

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

[59] Taillard, É., Badeau, P., Gendreau, M., Guertin, F., Potvin, J.-Y. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science 31*, 2 (1997), 170–186.

[60] Tavakkoli-Moghaddam, R., Safaei, N., Kah, M., Rabbani, M. A new capacitated vehicle routing problem with split service for minimizing fleet cost by simulated annealing. *Journal of the Franklin Institute 344*, 5 (2007), 406 – 425.

[61] Toth, P., Vigo, D., Eds. *The vehicle routing problem*. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[62] Wilck IV, J., Cavalier, T. A construction heuristic for the split delivery vehicle routing problem. *American Journal of Operations Research 2*, 2 (2012), 153–162.

[63] Wilck IV, J., Cavalier, T. A genetic algorithm for the split delivery vehicle routing problem. *American Journal of Operations Research 2*, 2 (2012), 207–216.

[64] Zanakis, S. H., Evans, J. R., Vazacopoulos, A. A. Heuristic methods and applications: A categorized survey. *European Journal of Operational Research 43*, 1 (1989), 88 – 110.