Universidade Federal Fluminense

MARCIO TADAYUKI MINE

Um algoritmo heurístico híbrido para o problema de roteamento de veículos com coleta e entrega simultânea

NITERÓI-RJ

2009

Universidade Federal Fluminense

MARCIO TADAYUKI MINE

Um algoritmo heurístico híbrido para o problema de roteamento de veículos com coleta e entrega simultânea

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

Orientador:
Prof. Luiz Satoru Ochi, D.Sc.

Co-Orientador:
Prof. Marcone Jamilson Freitas Souza, D.Sc.

NITERÓI-RJ

2009

Um algoritmo heurístico híbrido para o problema de roteamento de veículos com coleta e entrega simultânea

Marcio Tadayuki Mine

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

Aprovada por:

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

Prof. Marcone Jamilson Freitas Souza, D.Sc. / DECOM-UFOP
(Co-Orientador)

Prof. Eduardo Uchoa Barboza, D.Sc. / EP-UFF

Prof. Virgílio José Martins Ferreira Filho, D.Sc. / PEP-UFRJ

Niterói, 10 de junho de 2009.

À minha família por todo apoio e carinho.

Agradecimentos

Aos meus pais Massaharu e Neusa, por todo o apoio, carinho, incentivo, educação e por estarem sempre presentes em toda minha vida.

Aos meus irmãos Flávio e Otávio, pelos fortes laços que representam nossa união, amizade, respeito, carinho.

À minha avó Yoshiko, por todos os momentos de felicidade.

À Adrielle, por todos os gestos de amor e carinho, pela compreensividade, pelo incentivo e por todos os momentos que passamos juntos.

Aos professores Satoru e Marcone, pela preciosa orientação, pelo esfoço e comprometimento na realização deste trabalho, além de seus incentivos e amizade.

Aos amigos Matheus e Silas pela amizade, incentivo, apoio e companherismo ao longo desta caminhada.

Aos companheiros André, Daniel (Xico), Rodrigo e Thiago, pela hospitalidade e amizade.

Aos amigos Thiago, Tomaz e Hadriel pela amizade, apoio e companherismo.

Aos irmãos e irmãs da República K-ZONA, pela amizade, união, companherismo e apoio.

Ao Gleidson pela amizade e pelas palavras de força e incentivo.

A todos os amigos da Gapso, pela amizade, pela ajuda no desenvolvimento deste trabalho, pelas festas, *happy-hours* e por todos os momentos de descontração.

Aos professores e funcionários do Instituto de Computação da Universidade Federal Fluminense, pelos ensinamentos e ajuda durante o mestrado.

Ao Anand, tanto pela sua disposição quanto pelas sugestões que contribuiram no desenvolvimento deste trabalho.

Ao Pucca, pelo auxílio com a biblioteca gráfica do LaTeX.

À CAPES, pela contribuição financeira parcial, a qual foi essencial para a conclusão dessa pós-graduação.

Enfim, agradeço à todos que, de alguma forma, acreditaram e torceram por mim, participaram de minha vida e na realização deste trabalho.

Resumo

Este trabalho trata do Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Este problema é de grande importância na área da logística reversa; possuindo diferentes aplicações que incluem o planejamento da distribuição da indústria de bebidas e a logística postal. O PRVCES pertence à classe NP-difícil, uma vez que ele pode ser reduzido ao Problema de Roteamento de Veículos clássico quando nenhum cliente necessita de serviço de coleta. Para resolvê-lo, propõe-se um algoritmo heurístico híbrido, denominado GENILS, baseado nas técnicas *Iterated Local Search*, Descida em Vizinhança Variável, Inserção Mais Barata e GENIUS. O algoritmo proposto foi testado em três conjuntos de problemas-teste consagrados da literatura e se mostrou competitivo com as melhores abordagens existentes. Dos 72 problemas-teste desse conjunto, em 49 o GENILS foi capaz de alcançar os melhores resultados existentes e, em 9, superou as melhores soluções da literatura.

Palavras-chave: Problema de Roteamento de Veículos com Coleta e Entrega Simultânea, *Iterated Local Search*, Descida em Vizinhança Variável, GENIUS, Inserção Mais Barata.

Abstract

This work addresses the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD). This problem is of great importance in the field of reverse logistics and it has several applications including the planning of the distribution of the drinks industry and the postal logistics. The VRPSPD is NP-hard, since it can be reduced to the classical Vehicle Routing Problem when no client needs the pickup service. To solve it, we propose a hybrid heuristic algorithm, called GENILS, based on Iterated Local Search, Variable Neighborhood Descent, Cheapest Insertion and GENIUS. The proposed algorithm was tested on three well-known sets of instances found in literature and it was competitive with the best existing approaches. For the 72 instances tested, the GENILS was able to reach 49 best results of literature and was able to improve 9 best known solutions.

Keywords: Vehicle Routing Problem with Simultaneous Pickup and Delivery, Iterated Local Search, Variable Neighborhood Descent, GENIUS, Cheapest Insertion.

Siglas e Abreviações

AG	:	Algoritmos Genéticos
G3-Opt	:	Procedimento baseado na heurística GENIUS e na busca local *3-optimal*
G4-Opt	:	Procedimento baseado na heurística GENIUS e na busca local *4-optimal*
GENI	:	*Generalized Insertion*
GENILS	:	Algoritmo heurístico híbrido proposto neste trabalho
ILS	:	*Iterated Local Search*
IMB-1R	:	Inserção Mais Barata com construção rota a rota
IMB-MR	:	Inserção Mais Barata com construção simultânea de múltiplas rotas
IT1	:	Inserção Tipo I da heurística GENIUS
IT2	:	Inserção Tipo II da heurística GENIUS
LNS	:	*Large Neighborhood Search*
PRC	:	Problema do Caixeiro Viajante
PRV	:	Problema de Roteamento de Veículos
PRVCES	:	Problema de Roteamento de Veículos com Coleta e Entrega Simultânea
RT1	:	Remoção Tipo I da heurística GENIUS
RT2	:	Remoção Tipo II da heurística GENIUS
US	:	*Unstringing and Stringing*
VND	:	*Variable Neighborhood Descent* (Descida em Vizinhança Variável)
VNS	:	*Variable Neighborhood Search*
VRGENI	:	Fase GENI aplicado ao PRVCES
VRP	:	*Vehicle Routing Problem*
VRPSPD	:	*Vehicle Routing Problem with Simultaneous Pickups and Deliveries*
VRUS	:	Fase US aplicado ao PRVCES

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

1.1 O Problema de Roteamento de Veículos

O Problema de Roteamento de Veículos (PRV), conhecido na literatura como *Vehicle Routing Problem* (VRP), foi originalmente proposto por [Dantzig and Ramser, 1959] e pode ser definido da seguinte forma: Dado um conjunto N de clientes, cada qual com uma demanda d_i e uma frota de veículos com capacidade Q, estabelecer os trajetos de custo mínimo a serem percorridos pelos veículos, de forma a atender completamente a demanda dos clientes. O PRV é amplamente estudado na literatura devido a sua dificuldade de resolução e alta aplicabilidade na área de logística. Diversas variações do PRV foram propostas, cujas características básicas são as seguintes:

- Tipo da frota: homogênea, caso todos os veículos possuam as mesmas características ou heterogênea, caso contrário;
- Tamanho da frota: um ou múltiplos veículos;
- Tempo: tempo de serviço em cada cliente e janelas de tempo (horário em que o cliente se encontra disponível para o serviço);
- Natureza da demanda: é determinística se a demanda dos clientes for conhecida ou estocástica se a demanda estiver relacionada a uma determinada distribuição de probabilidade;
- Periodicidade: o planejamento pode ser realizado em um determinado período de tempo;

- Operação: pode ser de coleta, entrega ou ambas;
- Número de depósitos: um ou vários depósitos.

Em 1989, [Min, 1989] propôs uma importante variante do PRV: o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES), em que os serviços de entrega e coleta devem ser realizadas simultaneamente. O PRVCES é um problema presente na área da logística reversa, a qual tem por objetivo o planejamento do transporte de produtos aos clientes, bem como o retorno de resíduos ou produtos utilizados por esses para a reciclagem ou depósitos especializados. A logística reversa pode ser observada, por exemplo, na logística postal, no planejamento da distribuição da indústria de bebidas e em processos que envolvem a reutilização, reciclagem e tratamento dos resíduos dos produtos.

O PRVCES pertence à classe de problemas NP-difíceis, uma vez que ele pode ser reduzido ao PRV clássico quando nenhum cliente necessita de serviço de coleta. Dessa forma, a abordagem mais comum na literatura é por meio de heurísticas baseadas em inserção e em clusterização e de metaheurísticas, tais como Descida em Vizinhança Variável [Mladenović and Hansen, 1997], Busca Tabu [Glover and Laguna, 1997], *Iterated Local Search* [Stützle and Hoos, 1999] e *Guided Local Search* [Voudouris and Tsang, 1996].

1.2 Objetivos do trabalho

Este trabalho tem como objetivo geral o desenvolvimento de um algoritmo eficiente de otimização, baseado na metaheurística *Iterated Local Search*, para resolver o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea.

1.2.1 Objetivos específicos

São os seguintes os objetivos específicos:

1. Fazer uma revisão de literatura do PRVCES e de suas técnicas de solução;
2. Pesquisar e estudar técnicas heurísticas, metaheurísticas e o algoritmo GENIUS [Gendreau *et al.*, 1992];
3. Desenvolver um algoritmo heurístico híbrido baseado na metaheurística *Iterated Local Search*, combinado com adaptações dos procedimentos Descida em Vizinhança Variável, Inserção Mais Barata e GENIUS, para resolver o PRVCES;

4. Avaliar o desempenho do algoritmo desenvolvido com os resultados da literatura;

1.3 Estrutura do trabalho

O presente trabalho está dividido em seis capítulos, incluindo esta introdução.

O Capítulo 2 apresenta a definição do problema abordado neste trabalho, o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES).

No Capítulo 3 é apresentada uma revisão bibliográfica sobre os diversos métodos utilizados na resolução do problema de roteamento de veículos, bem como a forma com que diversos autores tratam esse problema. Apresenta-se também a descrição das heurísticas construtivas Inserção Mais Barata e GENIUS e das metaheurísticas Descida em Vizinhança Variável (VND) e *Iterated Local Search* (ILS).

No Capítulo 4 é apresentado o algoritmo proposto, denominado GENILS, para resolver o PRVCES. Para detalhar esse algoritmo, o qual é baseado na metaheurística *Iterated Local Search*, é mostrada como uma solução é representada, como são geradas as soluções iniciais, as estruturas de vizinhança utilizadas, a função de avaliação, as adaptações das heurísticas Inserção Mais Barata e GENIUS ao PRVCES, bem como o procedimento VND usado como busca local do GENILS.

No Capítulo 5 são apresentados e analisados os resultados computacionais e no Capítulo 6 são apresentadas as conclusões e apontadas as sugestões para trabalhos futuros.

Capítulo 2

Descrição do Problema Abordado

O Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES), ou *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD), é uma variante do Problema de Roteamento de Veículos (PRV) clássico. Neste problema existe um depósito com uma frota ilimitada de veículos de capacidade Q e um conjunto N de clientes espalhados geograficamente. Cada cliente $i \in N$ está associado a duas quantidades d_i e p_i, que representam a demanda por um determinado produto e a coleta no cliente i, respectivamente. O objetivo do problema é definir as rotas necessárias para atender a todos os clientes, de forma a minimizar os custos referentes ao deslocamento dos veículos e satisfazer as seguintes restrições:

1. Cada rota deve iniciar e finalizar no depósito;
2. Todos os clientes devem ser visitados uma única vez e por um único veículo;
3. As demandas por coleta e entrega de cada cliente devem ser completamente atendidas;
4. A carga do veículo, em qualquer momento, não pode superar a capacidade do mesmo;

Em algumas variantes desse problema, considera-se também a necessidade de cada veículo não percorrer mais que um determinado limite de distância (tempo) D.

A Figura 2.1 ilustra um exemplo do PRVCES. Nesta Figura, os clientes são representados pelos números 1 a $|N|$ e o depósito é representado pelo número 0 (zero). Cada par $[d_i/p_i]$ denota a demanda e coleta em um cliente i, respectivamente.

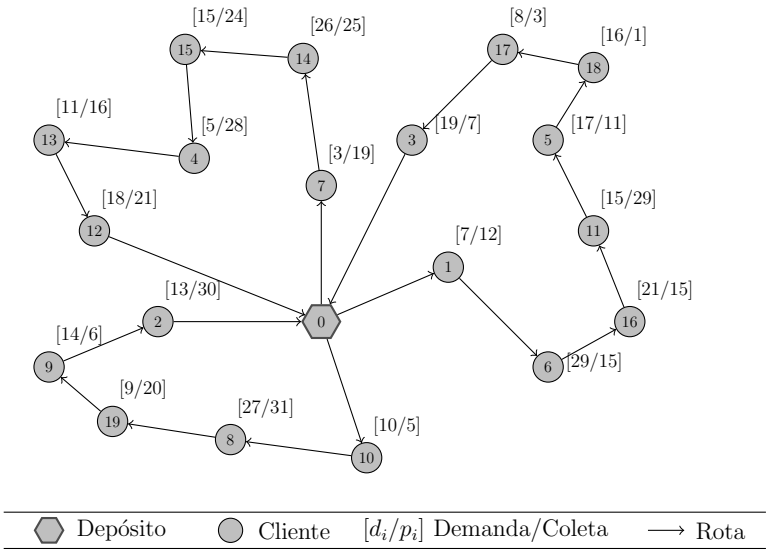

Figura 2.1: Exemplo do PRVCES.

Nessa figura, há três rotas a serem executadas por veículos de capacidade $Q = 150$. Em uma delas, o veículo sai do depósito e atende aos clientes 10, 8, 19, 9 e 2, retornando ao depósito no final. No primeiro cliente atendido nessa rota, é feita uma entrega de 10 unidades do produto e recolhida outras 5 unidades. A última visita do veículo ocorre no cliente 2, o qual demanda 13 unidades do produto e necessita que sejam coletadas 30 unidades.

Capítulo 3

Revisão Bibliográfica

3.1 Introdução

O PRVCES foi introduzido por [Min, 1989] e pertence à classe NP-difícil, uma vez que ele pode ser reduzido ao PRV clássico, quando a oferta de todos os clientes é nula. Em seu trabalho, o autor propôs um método de três fases para resolver o planejamento de distribuição de materiais para bibliotecas públicas. A primeira fase do método consiste em agrupar os clientes em *clusters* através do Método de Ligação por Médias (*Average Linkage Method* [Anderberg, 1973]). A segunda associa os veículos às respectivas rotas e a terceira consiste em resolver cada *cluster* por meio de uma heurística para o Problema do Caixeiro Viajante. Essa heurística atribui, iterativamente, uma penalidade aos arcos em que a carga do veículo foi excedida, procurando, dessa forma, gerar uma solução viável.

Na literatura, as abordagens mais comuns para o PRVCES são através de métodos de programação matemática, heurísticas baseadas em inserção e metaheurísticas.

[Halse, 1992] aborda o PRVCES por meio de uma heurística que consiste em, primeiramente, associar os clientes aos veículos e, em seguida, gerar as rotas através de um procedimento baseado no método *3-optimal*.

[Dethloff, 2001] desenvolveu uma adaptação do método da Inserção Mais Barata, em que os clientes são adicionados às rotas seguindo três critérios: (i) distância; (ii) capacidade residual e (iii) distância do cliente ao depósito. Nesse trabalho não foi aplicado nenhum método de refinamento da solução.

O PRVCES com janelas de tempo foi tratado em [Angelelli and Mansini, 2002]. Os

autores desenvolveram um algoritmo *branch-and-price*, utilizando a formulação do problema de recobrimento (*set covering*). Esse algoritmo resolveu problemas-teste com até 20 clientes e 5 veículos.

[Vural, 2003] desenvolveu duas versões do Algoritmo Genético (AG) [Goldberg, 1989]. A primeira faz a codificação dos indivíduos através de chaves aleatórias (*Random Keys* [Bean, 1994]) e a segunda foi implementada como uma heurística de refinamento, baseada na estrutura do AG desenvolvido por [Topcuoglu and Sevilmis, 2002].

[Gökçe, 2004] trata o problema com Colônia de Formigas [Dorigo *et al.*, 1996] e utiliza a heurística *2-optimal* como um procedimento de pós-otimização.

[Nagy and Salhi, 2005] desenvolveram uma metodologia para o PRVCES considerando a restrição de limite de tempo para percorrer cada rota. Essa metodologia reúne diferentes heurísticas para resolver o PRV clássico, tais como, *2-optimal*, *3-optimal*, realocação, troca e, além disso, inclui procedimentos para viabilizar a solução.

[Dell'Amico *et al.*, 2006] utilizaram a técnica *branch-and-price* através de duas abordagens: programação dinâmica e relaxação do espaço de estados (*state space relaxation*). A formulação matemática básica apresentada nesse trabalho é descrita na Seção 3.2.1.

[Crispim and Brandão, 2005] propõem uma técnica híbrida, combinando as metaheurísticas Busca Tabu [Glover and Laguna, 1997] e Descida em Vizinhança Variável (VND) [Hansen and Mladenović, 2001]. Para gerar uma solução foi utilizado o método da varredura (*sweep method*) e, para refinar uma solução, foi aplicado um procedimento composto por movimentos de realocação e troca.

[Röpke and Pisinger, 2006] desenvolveram uma heurística baseada nas técnicas VNS [Mladenović and Hansen, 1997] e *Large Neighborhood Search* (LNS) [Shaw, 1998]. O LNS é uma busca local baseada em duas idéias para definir e explorar estruturas de vizinhança de alta complexidade. A primeira idéia é fixar uma parte da solução e assim definir o espaço de soluções. A segunda consiste em realizar a busca por meio de programação por restrições (*Constraint Programming*), programação inteira mista (*Mixed Integer Programming*), técnicas *branch-and-price*, *branch-and-cut*, geração de colunas, entre outros.

[Montané and Galvão, 2006] utilizaram a metaheurística Busca Tabu considerando quatro tipos de estruturas de vizinhança. Essas estruturas utilizam os movimentos de realocação, troca e *crossover*. Para gerar uma solução vizinha foram desenvolvidas duas estratégias, sendo que uma considera o primeiro movimento viável e a outra, o melhor movimento viável. A metaheurística foi testada em um conjunto de 87 problemas-teste

envolvendo 50 a 400 clientes.

[Chen, 2006] propôs uma técnica baseada nas metaheurísticas *Simulated Annealing* [Kirkpatrick *et al.*, 1983] e Busca Tabu, enquanto [Chen and Wu, 2006] desenvolveram uma metodologia baseada na heurística *record-to-record travel* [Dueck, 1993], a qual é uma variação do *Simulated Annealing*.

[Bianchessi and Righini, 2007] apresentam algoritmos construtivos, heurísticas de refinamento e técnicas baseadas na metaheurística Busca Tabu. Essas técnicas utilizam movimentos de troca de nós (*node-exchange-based*) e troca de arcos (*arc-exchange-based*).

[Wassan *et al.*, 2007] propõem uma versão reativa da metaheurística Busca Tabu. Para gerar uma solução inicial, foi utilizado o método da varredura (*sweep method*) e para explorar o espaço de soluções, foram utilizados os movimentos de realocação (*shift*), de troca (*swap*) e de inversão do sentido da rota (*reverse*).

[Zachariadis *et al.*, 2009] abordam o PRVCES com uma técnica híbrida, combinando as metaheurísticas Busca Tabu e *Guided Local Search* [Voudouris and Tsang, 1996].

[Subramanian *et al.*, 2008] desenvolveram uma metodologia híbrida, composta pelas metaheurísticas *Iterated Local Search* (ILS) e Descida em Vizinhança Variável (VND). Para gerar uma solução inicial foi utilizado uma adaptação da heurística de inserção de [Dethloff, 2001], porém sem considerar a capacidade residual do veículo. A busca local definida pelo ILS é feita pelo VND, que explora o espaço de soluções usando os movimentos descritos na Seção 4.2, a saber, *shift(1,0)*, *shift(2,0)*, *crossover*, *swap(1,1)*, *swap(2,1)*, *swap(2,2)*. O VND realiza, a cada melhora na solução corrente, uma intensificação nas rotas alteradas, por meio dos procedimentos de busca local *or-opt*, *2-opt*, *exchange* e *reverse*. O procedimento *or-opt* que foi implementado consiste em permutar um, dois ou três clientes consecutivos em uma rota. O *2-opt* e o *exchange* realizam a permutação de um par de arcos e dois clientes, respectivamente. O movimento *reverse* consiste em inverter o sentido da rota, caso haja redução na carga máxima do veículo nessa rota. Os mecanismos de perturbação aplicados no ILS foram o *ejection chain*, o *double swap(1,1)* e o *double bridge*. O *ejection chain* consiste em transferir um cliente de cada rota a outra adjacente. O *double swap(1,1)* consiste em realizar dois movimentos *swap(1,1)* e o *double bridge* consiste em remover quatro arcos e inserir quatro novos arcos. A metodologia foi testada em problemas-teste envolvendo 50 a 400 clientes. Uma descrição pormenorizada desse algoritmo, bem como uma nova formulação de programação matemática para o PRVCES (descrita na Seção 3.2.2) pode ser encontrada em [Subramanian, 2008].

Uma revisão bastante detalhada do PRVCES encontra-se em [Parragh *et al.*, 2008a, Parragh *et al.*, 2008b].

Com relação aos problemas-teste para o PRVCES há, na literatura, os seguintes conjuntos disponíveis:

- [Dethloff, 2001]: conjunto de 40 problemas-teste envolvendo 50 clientes cada.
- [Salhi and Nagy, 1999]: conjunto de 28 problemas-teste com 50 a 199 clientes, sendo que a metade desses possuem restrições de limite de tempo;
- [Montané and Galvão, 2006]: conjunto de 18 problemas-teste com 100, 200 e 400 clientes.

Até o momento, os melhores resultados encontrados na literatura para esses problemas-teste pertencem a:

- [Chen and Wu, 2006]: um problema-teste de [Salhi and Nagy, 1999];
- [Röpke and Pisinger, 2006]: 26 problemas-teste de [Dethloff, 2001];
- [Wassan *et al.*, 2007]: 6 problemas-teste de [Salhi and Nagy, 1999];
- [Zachariadis *et al.*, 2009]: 6 problemas-teste de [Salhi and Nagy, 1999] e 27 de [Dethloff, 2001];
- [Subramanian, 2008] e [Subramanian *et al.*, 2008]: todos os problemas-teste propostos por [Dethloff, 2001] e [Montané and Galvão, 2006] e 17 de [Salhi and Nagy, 1999].

3.2 Formulação Matemática

Descrevem-se, a seguir, as formulações matemáticas propostas por [Dell'Amico *et al.*, 2006] e [Subramanian, 2008] para resolver o PRVCES.

3.2.1 Formulação de Dell'Amico *et al.* (2006)

A formulação matemática apresentada no trabalho de [Dell'Amico *et al.*, 2006] é descrita a seguir. Nesta formulação, são consideradas as seguintes notações:

N: conjunto dos clientes;

N^+: conjunto dos clientes incluindo o depósito (representado por 0);

A: conjunto dos arcos (i, j) com $i, j \in N^+$;

c_{ij}: distância entre o cliente i e j;

D_i: quantidade de produtos a ser entregue no cliente i;

P_i: quantidade de produtos a ser coletado no cliente i;

K: número de veículos disponíveis;

Q: capacidade do veículo.

As variáveis de decisão envolvidas neste modelo são:

x_{ij}: indica se o arco (i, j) está presente na solução ($x_{ij} = 1$) ou não ($x_{ij} = 0$)

d_{ij}: quantidade de produtos a serem entregues a clientes e escoados no arco (i, j)

p_{ij}: quantidade de produtos coletados de clientes e escoados no arco (i, j)

O modelo de programação linear inteira mista descrito pelos autores é definido pelas equações (3.1) a (3.9):

$$\text{(P1)} \qquad \text{Minimizar} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{3.1}$$

$$\text{s.a:} \sum_{j\in N^+} x_{ij} = 1 \qquad (i \in N) \tag{3.2}$$

$$\sum_{j\in N} x_{0j} \leq K \tag{3.3}$$

$$\sum_{j\in N^+} x_{ij} = \sum_{j\in N^+} x_{ji} \qquad (i \in N^+) \tag{3.4}$$

$$\sum_{j\in N^+} p_{ij} - \sum_{j\in N^+} p_{ji} = P_i \qquad (i \in N) \tag{3.5}$$

$$\sum_{j\in N^+} d_{ji} - \sum_{j\in N^+} d_{ij} = D_i \qquad (i \in N) \tag{3.6}$$

$$p_{ij} + d_{ij} \leq Q\, x_{ij} \qquad ((i, j) \in A) \tag{3.7}$$

$$p_{ij}, d_{ij} \geq 0 \qquad ((i, j) \in A) \tag{3.8}$$

$$x_{ij} \in \{0, 1\} \qquad ((i, j) \in A) \tag{3.9}$$

Neste modelo, a função objetivo (3.1) visa minimizar a distância total percorrida pelos veículos; as restrições (3.2) e (3.4) garantem que todo cliente só será visitado uma única vez; a restrição (3.3) limita a quantidade de veículos utilizados; (3.4), (3.5) e (3.6) são restrições de conservação de fluxo do número de veículos e da coleta e entrega realizada; (3.7) assegura que a capacidade do veículo não será excedida; (3.8) indicam que as variáveis

p_{ij} e d_{ij} são contínuas e não-negativas e (3.9) definem as variáveis x_{ij} como binárias.

3.2.2 Formulação de Subramanian (2008)

A formulação matemática apresentada no trabalho de [Subramanian, 2008] é descrita a seguir. Nessa formulação, são consideradas as seguintes notações:

- N: conjunto dos clientes;
- N^+: conjunto dos clientes incluindo o depósito (0);
- N^*: conjunto dos clientes incluindo o depósito e uma cópia do depósito ($\bar{0}$);
- A: conjunto dos arcos (i, j) com $i, j \in N^*$;
- c_{ij}: distância entre o cliente i e j;
- D_i: quantidade de produtos a ser entregue no cliente i;
- P_i: quantidade de produtos a ser coletado no cliente i;
- K: número de rotas ou veículos disponíveis;
- Q: capacidade do veículo.

As variáveis de decisão envolvidas neste modelo são:

- x_{ij}: indica se o arco (i, j) está presente na solução ($x_{ij} = 1$) ou não ($x_{ij} = 0$)
- d_{ij}: quantidade de produtos a serem entregues a clientes e escoados no arco (i, j)
- p_{ij}: quantidade de produtos coletados de clientes e escoados no arco (i, j)
- w_{ij}: carga do veículo no arco $(i, j) \in A$, referente à entrega e coleta

O modelo de programação linear inteira mista dos autores é apresentado pelas equações (3.10) a (3.29).

$$\text{(P2)} \qquad \text{Minimizar} \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{3.10}$$

$$\text{s.a:} \qquad \sum_{j\in N^*} (d_{ji} - d_{ij}) = 2\, D_i \qquad (i \in N) \tag{3.11}$$

$$\sum_{j\in N} d_{0j} = \sum_{i\in N} D_i \tag{3.12}$$

$$\sum_{j\in N} d_{j0} = K\, Q - \sum_{i\in N} D_i \tag{3.13}$$

$$\sum_{j\in N^*} (p_{ij} - p_{ji}) = 2\, P_i \qquad (i \in N) \tag{3.14}$$

$$\sum_{j\in N} p_{j\bar{0}} = \sum_{i\in N} P_i \tag{3.15}$$

$$\sum_{j \in N} p_{\bar{0}j} = K\,Q - \sum_{i \in N} P_i \tag{3.16}$$

$$\sum_{j \in N^*} (w_{ji} - w_{ij}) = 2\,(D_i - P_i) \qquad (i \in N) \tag{3.17}$$

$$w_{0j} = d_{0j} \qquad (j \in N) \tag{3.18}$$

$$w_{j0} = d_{j0} \qquad (j \in N) \tag{3.19}$$

$$w_{j\bar{0}} = p_{j\bar{0}} \qquad (j \in N) \tag{3.20}$$

$$w_{\bar{0}j} = p_{\bar{0}j} \qquad (j \in N) \tag{3.21}$$

$$d_{ij} + d_{ji} = Q\,x_{ij} \qquad ((i,j) \in A) \tag{3.22}$$

$$p_{ij} + p_{ji} = Q\,x_{ij} \qquad ((i,j) \in A) \tag{3.23}$$

$$w_{ij} + w_{ji} = Q\,x_{ij} \qquad ((i,j) \in A) \tag{3.24}$$

$$\sum_{i \in N^*,\, i<k} x_{ik} + \sum_{j \in N^*,\, j>k} x_{kj} = 2 \qquad (k \in N) \tag{3.25}$$

$$\sum_{j \in N} x_{0j} \leq K \tag{3.26}$$

$$\sum_{j \in N} x_{j\bar{0}} \leq K \tag{3.27}$$

$$x_{ij} \in \{0,1\} \qquad ((i,j) \in A) \tag{3.28}$$

$$p_{ij}, d_{ij}, w_{ij} \geq 0 \qquad ((i,j) \in A) \tag{3.29}$$

Neste modelo, (3.10) representa a função objetivo, que consiste em minimizar o total das distâncias percorridas por todos os veículos; as restrições (3.11) garantem que todas as demandas por entrega sejam satisfeitas; a restrição (3.12) estabelece que a carga do veículo que sai do depósito seja igual ao somatório das demandas (entrega) dos clientes; a restrição (3.13) estabelece que a carga dos veículos que chegam no depósito seja igual à carga residual dos veículos quando saem do depósito; as restrições (3.14) a (3.16) são análogas às restrições (3.11) a (3.13), porém relativas a demanda por coleta; as restrições (3.17) asseguram que as entregas e coletas sejam realizadas simultaneamente; as restrições (3.18) a (3.24) estabelecem que a capacidade do veículo não seja excedida; as restrições (3.25) definem que o número de arcos incidentes em cada cliente seja igual a dois; as restrições (3.26) e (3.27) limitam a quantidade máxima de veículos utilizados; as restrições (3.29) indicam que as variáveis p_{ij}, d_{ij} e w_{ij} são contínuas e não-negativas e as restrições (3.28) definem as variáveis x_{ij} como binárias.

3.3 Heurísticas

As heurísticas são técnicas que visam a obtenção de soluções de boa qualidade em um tempo computacional aceitável. Essas técnicas, no entanto, não garantem a obtenção da solução ótima para o problema nem são capazes de garantir o quão próximo a solução obtida está da ótima.

As heurísticas podem ser construtivas ou de refinamento. As construtivas têm por objetivo construir uma solução, usualmente, elemento a elemento. A escolha de cada elemento está, geralmente, relacionada a uma determinada função que o avalia de acordo com sua contribuição para a solução. Tal função é bastante relativa, pois varia conforme o tipo de problema abordado.

As heurísticas de refinamento, também chamadas de mecanismos de busca local, são técnicas baseadas na noção de vizinhança. Para definirmos o que é uma vizinhança, seja S o espaço de busca de um problema de otimização e f a função objetivo a minimizar. O conjunto $N(s) \subseteq S$, o qual depende da estrutura do problema tratado, reúne um número determinado de soluções s', denominado vizinhança de s. Cada solução $s' \in N(s)$ é chamada de vizinho de s e é obtida a partir de uma operação chamada de movimento.

Em linhas gerais, esses métodos partem de uma solução inicial s_0, percorrem o espaço de busca por meio de movimentos, passando de uma solução para outra que seja sua vizinha.

A Subseção 3.3.1 apresenta a heurística GENIUS.

3.3.1 GENIUS

A heurística GENIUS foi desenvolvida por [Gendreau *et al.*, 1992] para resolver o Problema do Caixeiro Viajante (PCV), conhecido na literatura inglesa como *Traveling Salesman Problem*. Essa heurística é composta de duas fases, uma construtiva (GENI - *Generalized Insertion*) e a outra de refinamento (US - *Unstringing and Stringing*). A fase construtiva GENI baseia-se nos métodos de inserção e sua principal característica é que a inserção de um vértice v não é realizada necessariamente entre dois vértices consecutivos v_i e v_j. No entanto, esses dois vértices tornam-se adjacentes a v após a inserção. Para descrever a heurística GENIUS, considere as seguintes definições:

- V: conjunto dos vértices;

- V^+: conjunto dos vértices que estão na rota;
- V^-: conjunto dos vértices que não estão na rota;
- v: vértice, pertencente à V^-, a ser inserido entre os vértices v_i e $v_j \in V$;
- $\bar{v}_i$: vértice, pertencente à V^+ e adjacente à $\bar{v}_{i-1}$ e $\bar{v}_{i+1}$, a ser removido;
- v_k: vértice pertencente ao caminho de v_j a v_i;
- v_l: vértice pertencente ao caminho de v_i a v_j;
- v_{h+1}, v_{h-1}: vértices, pertencentes à V^+, sucessor e antecessor ao vértice $v_h \in V^+$, respectivamente;
- $N_p(v)$: vizinhança do vértice v, composta dos p vértices ($v_h \in V^+$) mais próximos à v;
- s: solução (rota) parcial ou completa.
- $\bar{s}$: solução parcial ou completa no sentido inverso.

O Algoritmo 1 apresenta o pseudocódigo básico da heurística GENIUS. Nesse algoritmo, a *FaseGENI* é a descrita na Seção 3.3.1.1 e a *FaseUS* está descrita na Subseção 3.3.1.2.

Algoritmo 1 GENIUS

1: $s_0 \leftarrow$ *GENI*(V);
2: $s \leftarrow$ *US*(s_0);
3: Retorne s;

3.3.1.1 Fase GENI

A fase construtiva GENI (*Generalized Insertion*) consiste em inserir, a cada iteração, um vértice $v \in V^-$ na rota por meio de dois tipos de inserção descritos a seguir.

A Inserção Tipo I (IT1) adiciona um vértice $v \in V^-$ na rota removendo os arcos (v_i, v_{i+1}), (v_j, v_{j+1}) e (v_k, v_{k+1}) e inserindo os arcos (v_i, v), (v, v_j), (v_{i+1}, v_k) e (v_{j+1}, v_{k+1}). Nessa inserção, considera-se que $v_k \neq v_i$ e $v_k \neq v_j$. O pseudocódigo dessa inserção é apresentado no Algoritmo 2 e a Figura 3.1 mostra a inserção do vértice $v = 8$ na rota entre os vértices $v_i = 6$ e $v_j = 7$, considerando $v_k = 11$. Observe que os caminhos $(v_{i+1}, \ldots, v_j)$ e $(v_{j+1}, \ldots, v_k)$ são invertidos após a inserção.

Algoritmo 2 GENI - Inserção Tipo I

1: Dados os vértices v, v_i, v_j e v_k e uma solução parcial s:
2: $s' \leftarrow s$;
3: **se** ($v_k \neq v_i \wedge v_k \neq v_j$) **então**
4: Remova de s' os arcos (v_i, v_{i+1}), (v_j, v_{j+1}) e (v_k, v_{k+1});
5: Insira em s' os arcos (v_i, v), (v, v_j), (v_{i+1}, v_k) e (v_{j+1}, v_{k+1});
6: Inverta o sentido dos caminhos $(v_{i+1}, \ldots, v_j)$ e $(v_{j+1}, \ldots, v_k)$;
7: **fim se**
8: Retorne s';

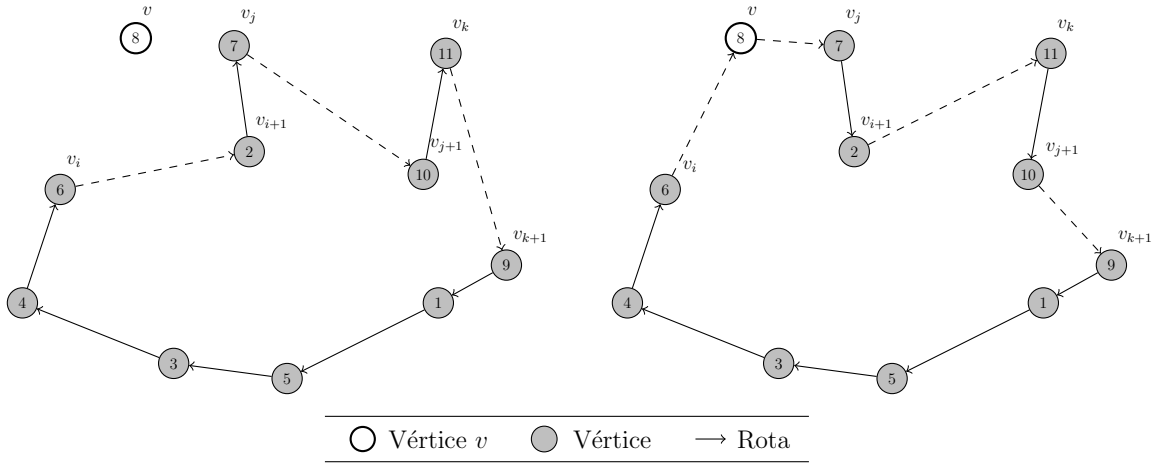

Figura 3.1: Exemplo da Inserção Tipo I da fase GENI.

A Inserção Tipo II (IT2) consiste em inserir um vértice $v \in V^-$ na rota removendo os arcos (v_i, v_{i+1}), (v_j, v_{j+1}), (v_{k-1}, v_k) e (v_{l-1}, v_l) e inserindo os arcos (v_i, v), (v, v_j), (v_l, v_{j+1}), (v_{k-1}, v_{l-1}) e (v_{i+1}, v_k). Nessa inserção, considera-se que $v_k \neq v_j$, $v_k \neq v_{j+1}$, $v_l \neq v_i$ e $v_l \neq v_{i+1}$. O Algoritmo 3 apresenta o pseudocódigo dessa inserção, enquanto a Figura 3.2 ilustra a inserção do vértice $v = 8$ na rota entre os vértices $v_i = 7$ e $v_j = 11$, considerando $v_k = 5$ e $v_l = 10$. Observe que os caminhos $(v_{i+1}, \ldots, v_{l-1})$ e $(v_l, \ldots, v_j)$ são invertidos após a inserção.

Algoritmo 3 GENI - Inserção Tipo II

1: Dados os vértices v, v_i, v_j, v_k e v_l e uma solução parcial s:
2: $s' \leftarrow s$;
3: **se** ($v_k \neq v_j \wedge v_k \neq v_{j+1} \wedge v_l \neq v_i \wedge v_l \neq v_{i+1}$) **então**
4: Remova de s' os arcos (v_i, v_{i+1}), (v_j, v_{j+1}), (v_{k-1}, v_k) e (v_{l-1}, v_l);
5: Insira em s' os arcos (v_i, v), (v, v_j), (v_l, v_{j+1}), (v_{k-1}, v_{l-1}) e (v_{i+1}, v_k);
6: Inverta o sentido dos caminhos $(v_{i+1}, \ldots, v_{l-1})$ e $(v_l, \ldots, v_j)$;
7: **fim se**
8: Retorne s';

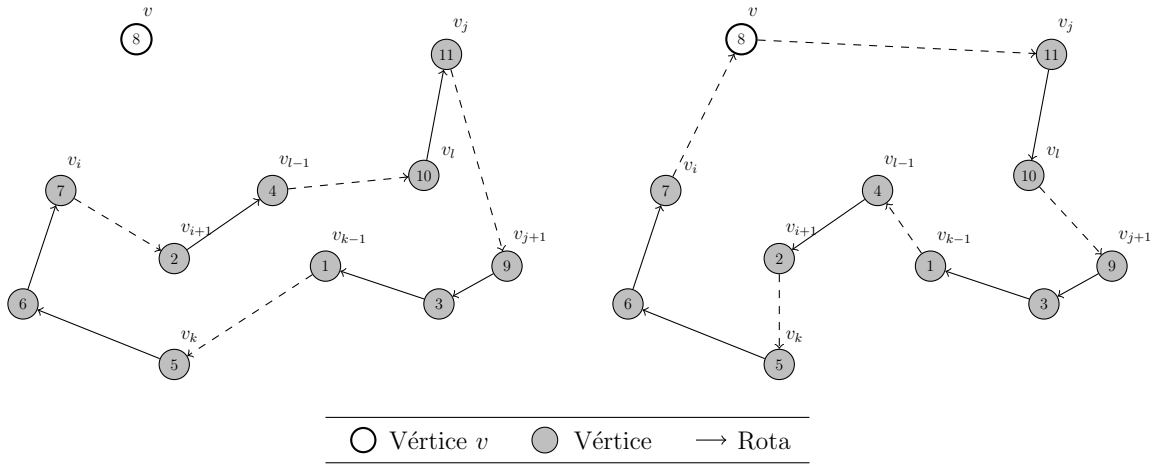

Figura 3.2: Exemplo da Inserção Tipo II da fase GENI.

A heurística GENI inicia-se construindo uma subrota s contendo, aleatoriamente, três vértices. A cada iteração, calcula-se o custo de inserção de um vértice arbitrário v na solução s, considerando os dois tipos de inserção IT1 e IT2 e as duas orientações possíveis da rota. Nesse cálculo, considera-se todas as combinações de v_i e $v_j \in N_p(v)$, $v_k \in N_p(v_{i+1})$ e $v_l \in N_p(v_{j+1})$. Realizado o cálculo, v é inserido considerando os vértices v_i, v_j, v_k e v_l que levam ao menor custo de inserção. Esse procedimento é repetido até que todos os vértices sejam inseridos na rota, ou seja, quando $V^+ = V$ e $V^- = \emptyset$.

O pseudocódigo da heurística GENI é apresentado pelo Algoritmo 4, o qual faz uso do procedimento de inserção mostrada no Algoritmo 5. Nesse último algoritmo, considera-se que $f(s)$ é o custo (distância total) da solução s.

Algoritmo 4 Fase construtiva GENI

1: $s \leftarrow \emptyset \Rightarrow V^- = V$;
2: **enquanto** ($|V^-| > 0$) **faça**
3: Selecione, aleatoriamente, um vértice $v \in V^-$;
4: $s \leftarrow$ *InserçãoGENI(v,s)*; { Algoritmo 5 }
5: $V^- = V^- \setminus \{v\}$;
6: **fim enquanto**
7: Retorne s;

Algoritmo 5 Inserção GENI

```
1: Dado um vértice v e uma solução s:
2: s* ← s;
3: para ( s' ∈ { s, s̄ } ) faça
4:    para ( v_i, v_j ∈ N_p(v) ) faça
5:       para ( v_k ∈ N_p(v_{i+1}) ) faça
6:          s'' ← InserçãoTipoI(s', v, v_i, v_j, v_k);   { Algoritmo 2 }
7:          se ( f(s'') < f(s*) ) então
8:             s* ← s'';
9:          fim se
10:         para ( v_l ∈ N_p(v_{j+1}) ) faça
11:            s'' ← InserçãoTipoII(s', v, v_i, v_j, v_k, v_l);   { Algoritmo 3 }
12:            se ( f(s'') < f(s*) ) então
13:               s* ← s'';
14:            fim se
15:         fim para
16:      fim para
17:   fim para
18: fim para
19: Retorne s*;
```

É importante destacar que a IT1 e a IT2 analisam um espaço reduzido da vizinhança explorada pelos procedimentos *3-optimal* [Steiglitz and Weiner, 1968] e *4-optimal*, respectivamente. A eficiência do método encontra-se no fato de que o espaço analisado é restrito ao número de vizinhos de cada vértice, sendo, no máximo, igual a p.

Para melhor entender essa heurística, considere o exemplo mostrado na Figura 3.3.

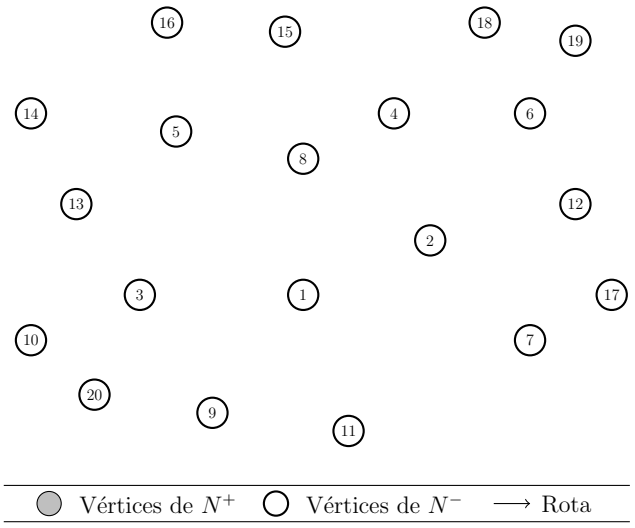

Figura 3.3: Exemplo de um problema do PCV, considerando 20 clientes.

Inicialmente, a heurística seleciona três vértices de forma arbitrária (no caso, 7, 4 e 20), conforme mostrado na Figura 3.4.

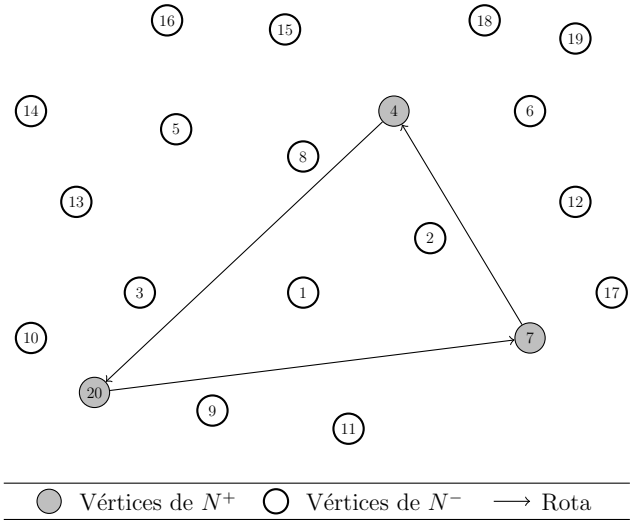

Figura 3.4: Subrota inicial do GENI contendo os vértices 7, 4 e 20.

Em seguida, seleciona-se, aleatoriamente, um vértice que ainda não está na rota, por exemplo, o vértice 14. O vértice 14 será adicionado na posição e com o tipo de inserção cujo custo seja mínimo. As Figuras 3.5 e 3.6 ilustram a inserção dos vértices 14 e 18, respectivamente.

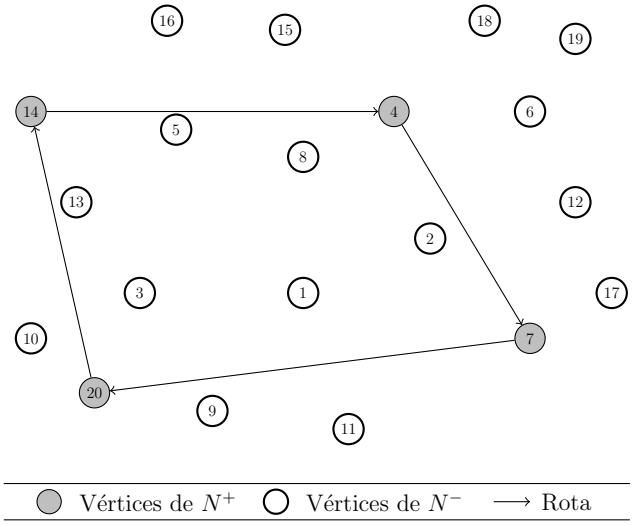

Figura 3.5: Inserção do vértice 14 com a heurística GENI.

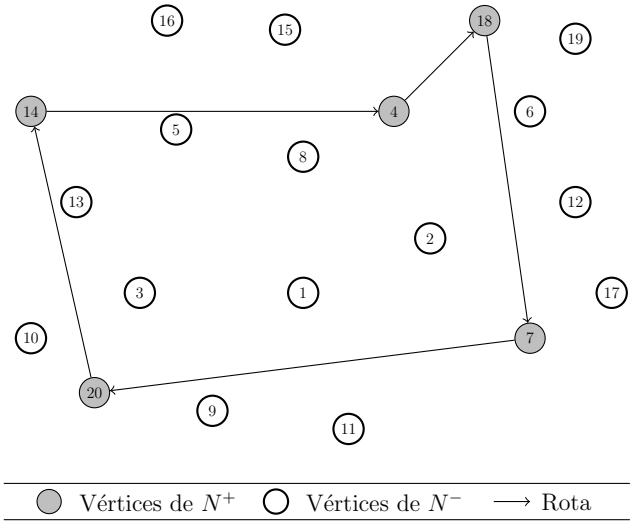

Figura 3.6: Inserção do vértice 18 com a heurística GENI.

A Figura 3.7 mostra a inclusão do vértice 15 na rota, considerando a Inserção Tipo I com $v_i = 14$, $v_j = 18$ e $v_k = 7$. Observe que, após a inserção, o vértice 15 torna-se adjacente aos vértices não consecutivos 14 e 18.

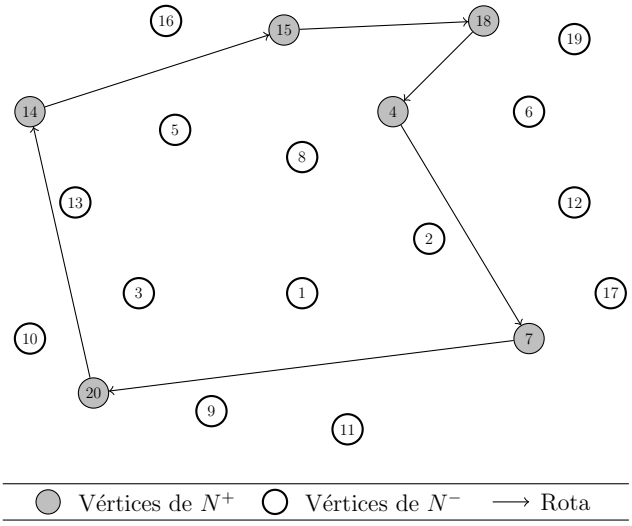

Figura 3.7: Inserção do vértice 15 com a heurística GENI.

O método é interrompido quando todos os vértices forem adicionados à rota. A solução gerada pela heurística GENI é apresentada na Figura 3.8.

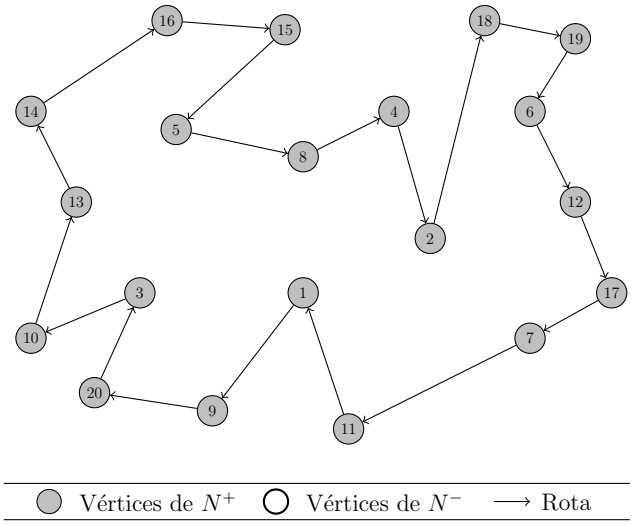

Figura 3.8: Solução inicial gerada pela heurística GENI.

3.3.1.2 Fase US

A fase de refinamento US (*Unstringing and Stringing*) consiste em, a cada iteração, remover um vértice da rota e reinserí-lo em outra posição na rota. A exclusão é realizada por meio de dois tipos de remoção, os quais serão descritos a seguir e a reinserção é feita pelas duas inserções da fase GENI.

A Remoção Tipo I (RT1) remove um vértice $\bar{v}_i \in V^+$ da rota excluindo os arcos $(\bar{v}_{i-1}, \bar{v}_i)$, $(\bar{v}_i, \bar{v}_{i+1})$, $(v_j, v_{j+1}$ e (v_l, v_{l+1}) e adicionando os arcos $(\bar{v}_{i-1}, v_l)$, $(\bar{v}_{i+1}, v_j)$ e (v_{l+1}, v_{j+1}). Nessa remoção, os caminhos $(\bar{v}_{i+1}, \ldots, v_l)$ e $(v_{l+1}, \ldots, v_j)$ são invertidos após a remoção. O pseudocódigo dessa remoção é apresentada no Algoritmo 6 e a Figura 3.9 mostra a exclusão do vértice $\bar{v}_i = 8$, considerando $v_l = 2$ e $v_j = 10$.

Algoritmo 6 US - Remoção Tipo I

1: Dados os vértices $\bar{v}_i$, v_j e v_l e uma solução s:
2: $s' \leftarrow s$;
3: Remova de s' os arcos $(\bar{v}_{i-1}, \bar{v}_i)$, $(\bar{v}_i, \bar{v}_{i+1})$, (v_j, v_{j+1}) e (v_l, v_{l+1});
4: Insira em s' os arcos $(\bar{v}_{i-1}, v_l)$, $(\bar{v}_{i+1}, v_j)$ e (v_{l+1}, v_{j+1});
5: Inverta o sentido dos caminhos $(\bar{v}_{i+1}, \ldots, v_l)$ e $(v_{l+1}, \ldots, v_j)$;
6: Retorne s';

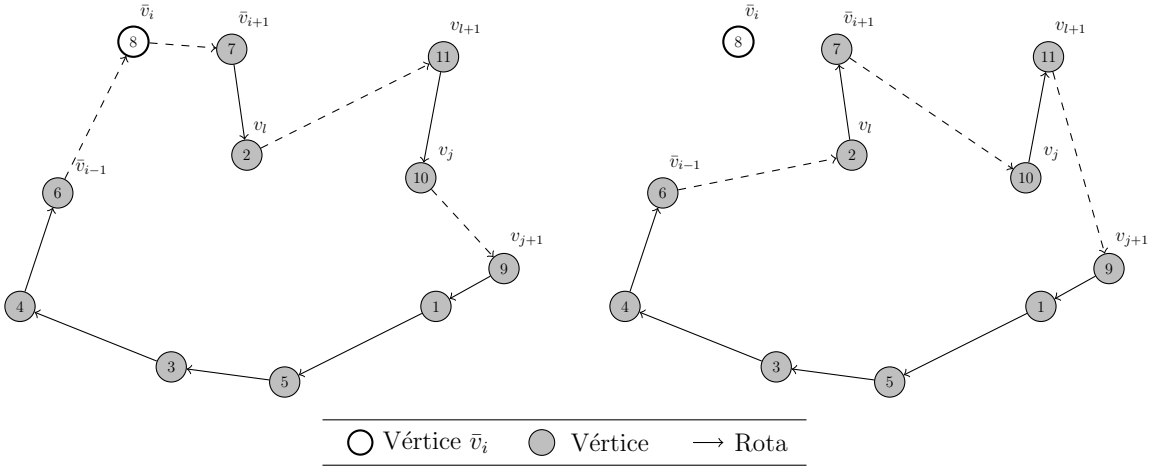

Figura 3.9: Exemplo da Remoção Tipo I da fase US.

A Remoção Tipo II (RT2) consiste em remover um vértice $\bar{v}_i \in V^+$, excluindo os arcos $(\bar{v}_{i-1}, \bar{v}_i)$, $(\bar{v}_i, \bar{v}_{i+1})$, (v_j, v_{j+1}), (v_{l-1}, v_l) e (v_k, v_{k+1}) e inserindo os arcos $(\bar{v}_{i-1}, v_k)$, (v_{j+1}, v_{l-1}), $(\bar{v}_{i+1}, v_l)$ e (v_j, v_{k+1}). Os caminhos $(v_j, \ldots, v_k)$ e $(\bar{v}_{i+1}, \ldots, v_l)$ são invertidos após a remoção. O Algoritmo 7 apresenta o pseudocódigo dessa remoção e a Figura 3.10 ilustra a exclusão do vértice $\bar{v}_i = 8$, considerando $v_j = 1$, $v_k = 2$ e $v_l = 9$.

Algoritmo 7 US - Remoção Tipo II

1: Dados os vértices $\bar{v}_i$, v_j, v_k e v_l e uma solução s:
2: $s' \leftarrow s$;
3: Remova de s' os arcos $(\bar{v}_{i-1}, \bar{v}_i)$, $(\bar{v}_i, \bar{v}_{i+1})$, (v_{j-1}, v_j), (v_l, v_{l+1}) e (v_k, v_{k+1});
4: Insira em s' os arcos $(\bar{v}_{i-1}, v_k)$, (v_j, v_l), $(\bar{v}_{i+1}, v_{l+1})$ e (v_{j-1}, v_{k+1});
5: Inverta o sentido dos caminhos $(v_j, \ldots, v_k)$ e $(\bar{v}_{i+1}, \ldots, v_l)$;
6: Retorne s';

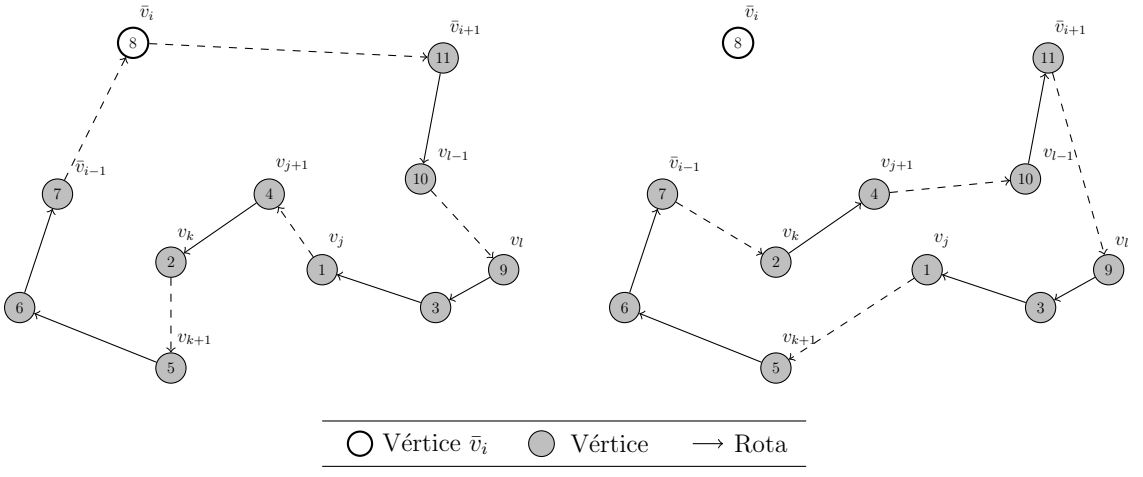

Figura 3.10: Exemplo da Remoção Tipo II da fase US.

O pseudocódigo da fase US é apresentada no Algoritmo 8. Nesse algoritmo, considere que p_z é o z-ésimo vértice a ser visitado. Dessa forma, p_1 e p_n representam, respectivamente, as posições do primeiro e do último vértice a ser visitado. Além disso, considere que $v(p_z, s)$ é o vértice que se encontra na posição p_z da solução s.

Algoritmo 8 Fase de refinamento US

```
1: Considere s como sendo uma solução inicial:
2: s* ← s;
3: p_z ← p_1;
4: enquanto ( p_z ≤ p_n ) faça
5:    v ← v(p_z, s);
6:    s' ← RemoçãoUS(v,s);  { Algoritmo 9 }
7:    s'' ← InserçãoGENI(v,s');  { Algoritmo 5 }
8:    se ( f(s'') < f(s*) ) então
9:       s* ← s'';
10:      p_z ← p_1;
11:   senão
12:      p_z ← p_{z+1};
13:   fim se
14:   s ← s'';
15: fim enquanto
16: s ← s*;
17: Retorne s;
```

Algoritmo 9 Remoção US

```
1: Dado um vértice v̄_i e uma solução s:
2: s* ← s;
3: para ( s' ∈ { s, s̄ } ) faça
4:    para ( v_k ∈ N_p(v̄_{i-1}) ) faça
5:       para ( v_j ∈ N_p(v_{k+1}) ) faça
6:          s'' ← RemoçãoTipoI(s', v̄_i, v_j, v_k);  { Algoritmo 6 }
7:          se ( f(s'') < f(s*) ) então
8:             s* ← s'';
9:          fim se
10:         para ( v_l ∈ N_p(v̄_{i+1}) ) faça
11:            s'' ← RemoçãoTipoII(s', v̄_i, v_j, v_k, v_l);  { Algoritmo 7 }
12:            se ( f(s'') < f(s*) ) então
13:               s* ← s'';
14:            fim se
15:         fim para
16:      fim para
17:   fim para
18: fim para
19: Retorne s*;
```

O algoritmo US realiza, a cada iteração, a remoção de um vértice $\bar{v}_i$ da solução s, que se encontra na posição p_z, por meio das duas remoções RT1 e RT2. Em seguida, o vértice $\bar{v}_i$ é adicionado com a configuração das inserções IT1 e IT2 da fase GENI a qual resulta no melhor custo de inserção. Se a solução gerada for melhor que a melhor solução encontrada (s^*), então o próximo vértice a ser considerado será o da primeira posição p_1. Caso contrário, o algoritmo avança para o vértice da próxima posição p_{z+1}. Esse procedimento é realizado até que todos os vértices sejam analisados.

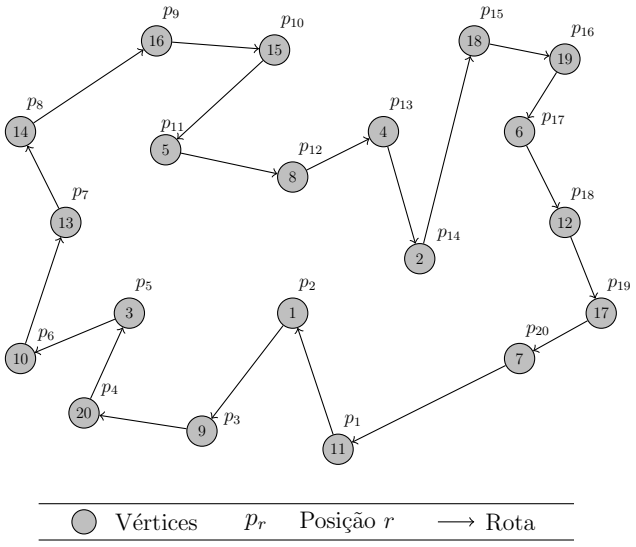

Figura 3.11: Exemplo da fase US.

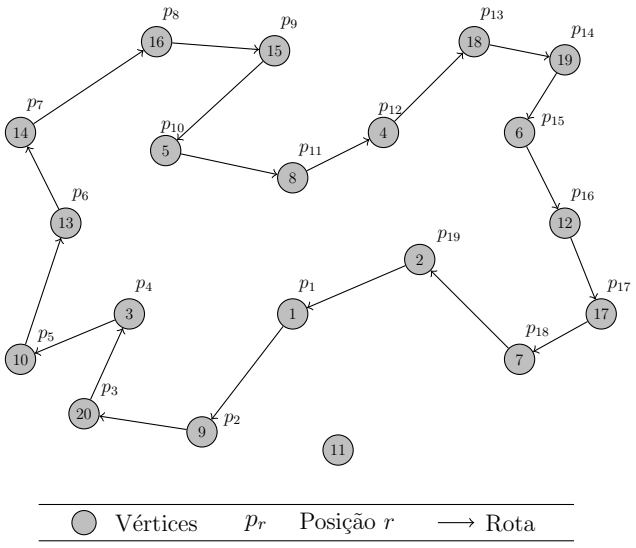

Figura 3.12: Remoção do vértice 11 localizado na posição p_1.

As Figuras 3.12 e 3.13 ilustram a primeira iteração da fase US, considerando o exemplo

apresentado na Figura 3.11. Esse exemplo corresponde à solução inicial gerada pela fase GENI (Figura 3.8). Nessas figuras, pode-se observar a retirada do vértice $\bar{v}_i = 11$, que se encontra na posição p_1 e a sua reinserção utilizando o Algoritmo 5 (*InserçãoGENI*).

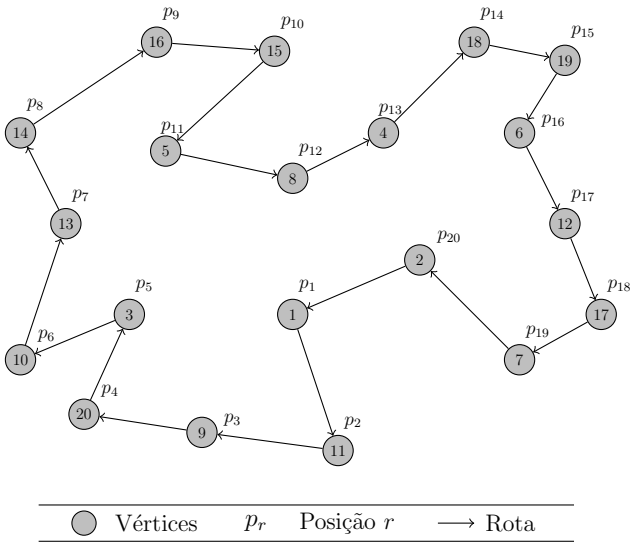

Figura 3.13: Reinserção do vértice 11 com o Algoritmo 5.

A Figura 3.14 mostra a solução final gerada pela fase US. Como a solução inicial foi gerada pela fase GENI, então a solução dessa figura também corresponde à gerada pela heurística GENIUS.

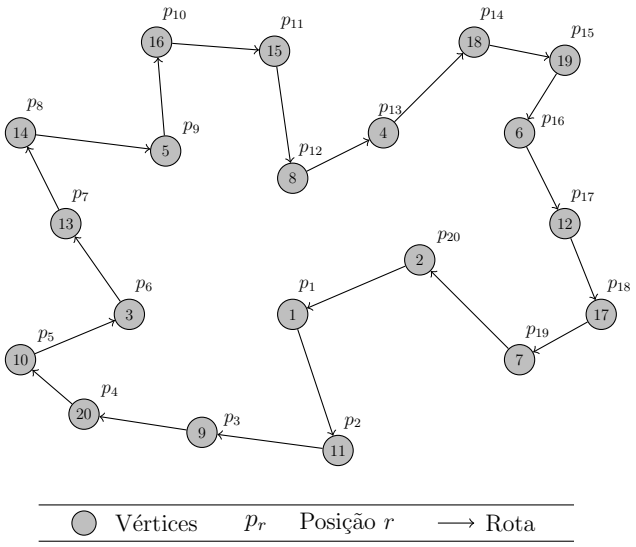

Figura 3.14: Solução gerada pela fase US e pela heurística GENIUS.

3.4 Metaheurísticas

Metaheurísticas são procedimentos de busca local destinados a resolver aproximadamente um problema de otimização, tendo a capacidade de escapar das armadilhas dos ótimos locais, ainda distantes de um ótimo global. Elas podem ser de busca local ou populacional. Na primeira, a exploração do espaço de soluções é feita por meio de movimentos, os quais são aplicados a cada passo sobre a solução corrente, gerando outra solução promissora em sua vizinhança. Já na segunda, trabalha-se com um conjunto de soluções, recombinando-as com o intuito de aprimorá-las.

Neste trabalho, foram utilizadas as metaheurísticas Descida em Vizinhança Variável (VND) e *Iterated Local Search* (ILS), as quais são descritas nas Seções 3.4.1 e 3.4.2, respectivamente.

3.4.1 Descida em Vizinhança Variável

A Descida em Vizinhança Variável [Hansen and Mladenović, 2001], conhecida na literatura inglesa como *Variable Neighborhood Descent* - VND, é uma metaheurística que consiste em explorar o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança.

Seja s a solução corrente e N a vizinhança de uma solução estruturada em r vizinhanças distintas, isto é, $N = N^{(1)} \cup N^{(2)} \cup \cdots \cup N^{(r)}$. O VND inicia-se analisando a primeira estrutura de vizinhança $N^{(1)}$. A cada iteração, o método gera o melhor vizinho s' da solução corrente s na vizinhança $N^{(k)}$. Caso s' seja melhor que s, então s' passa a ser a nova solução corrente e retorna-se à vizinhança $N^{(1)}$. Caso contrário, passa-se para a próxima estrutura de vizinhança $N^{(k+1)}$. O método termina quando não é possível encontrar uma solução $s' \in N^{(r)}$ melhor que a solução corrente.

O Algoritmo 10 mostra o pseudocódigo do VND.

Algoritmo 10 Descida em Vizinhança Variável

```
1: Seja r o número de estruturas de vizinhança distintas;
2: k ← 1;
3: enquanto ( k ≤ r ) faça
4:    Encontre o melhor vizinho s' ∈ N^(k)(s);
5:    se ( f(s') < f(s) ) então
6:       s ← s';
7:       k ← 1;
8:    senão
9:       k ← k + 1;
10:   fim se
11: fim enquanto
12: Retorne s;
```

3.4.2 *Iterated Local Search*

O *Iterated Local Search* (ILS) [Stützle and Hoos, 1999] é uma metaheurística que possui quatro componentes básicos, a saber, a geração de uma solução inicial, o mecanismo de perturbação, o método de busca local e o critério de aceitação. Primeiramente, gera-se uma solução inicial e aplica-se uma busca local. Para escapar do ótimo local s gerado, é feita uma perturbação, gerando uma nova solução s'. Em seguida, essa solução perturbada é refinada pelo método de busca local, obtendo-se um novo ótimo local s''. Esta solução tornar-se a nova solução corrente caso s'' seja aprovada por um critério de aceitação; caso contrário, ela é descartada e nova perturbação é feita a partir da solução s. Esse procedimento é repetido até que um determinado critério de parada seja satisfeito, como, por exemplo, um número máximo de iterações sem melhora na solução corrente ou um tempo máximo de processamento.

Vale ressaltar que a intensidade das perturbações não pode ser nem tão pequena e nem tão grande. Se a intensidade for muito pequena, s' poderá voltar à região de atração de s e com isso a probabilidade de encontrar novas soluções é reduzida. Se a intensidade da perturbação for muito elevada, s' será uma solução aleatória e o método funcionaria como um algoritmo de reinício aleatório.

Uma análise mais detalhada sobre a metaheurística ILS pode ser encontrada em [Lourenço *et al.*, 2003].

O pseudocódigo do ILS básico é apresentado no Algoritmo 11.

Algoritmo 11 *Iterated Local Search*

1: Seja s_0 uma solução inicial;
2: $s \leftarrow BuscaLocal(s_0)$;
3: **enquanto** (*critério de parada não satisfeito*) **faça**
4: $\quad s' \leftarrow Perturbação(s)$;
5: $\quad s'' \leftarrow BuscaLocal(s')$;
6: $\quad s \leftarrow CritérioAceitação(s, s'')$;
7: **fim enquanto**
8: Retorne s;

Capítulo 4

Metodologia

Neste capítulo é apresentada a metodologia proposta para resolver o PRVCES. Na Seção 4.1 mostra-se como uma solução do problema é representada, enquanto na Seção 4.2 são apresentados os movimentos utilizados para explorar o espaço de soluções do problema. Na Seção 4.3 mostra-se como uma solução é avaliada. Os métodos de geração de uma solução inicial são apresentados na Seção 4.4. Na Seção 4.5 é descrito o algoritmo proposto para resolver o PRVCES, o qual faz uso de um método de busca local descrito na Seção 4.6.

4.1 Representação de uma solução

Uma solução do PRVCES é representada como uma permutação de clientes, numerados de 1 a n e separadas em k partições, sendo k o número de rotas ou veículos utilizados. O elemento separador é indicado pelo valor zero (0), representando o depósito. Por exemplo, se existem 19 clientes a serem atendidos e 3 veículos disponíveis, então uma possível solução é:

$$s = [\ 0\ 7\ 14\ 15\ 4\ 13\ 12\ 0\ 1\ 6\ 16\ 11\ 5\ 18\ 17\ 3\ 0\ 10\ 8\ 19\ 9\ 2\ 0\ ]$$

em que [0 7 14 15 4 13 12 0], [0 1 6 16 11 5 18 17 3 0] e [0 10 8 19 9 2 0] são as rotas desta solução. A rota [0 10 8 19 9 2 0] indica que o veículo sai do depósito, visita os clientes 10, 8, 19, 9 e 2 nesta ordem e retorna ao depósito.

Para facilitar a visualização e o entendimento do trabalho, as soluções neste trabalho são representadas graficamente, conforme exemplificada na Figura 4.1.

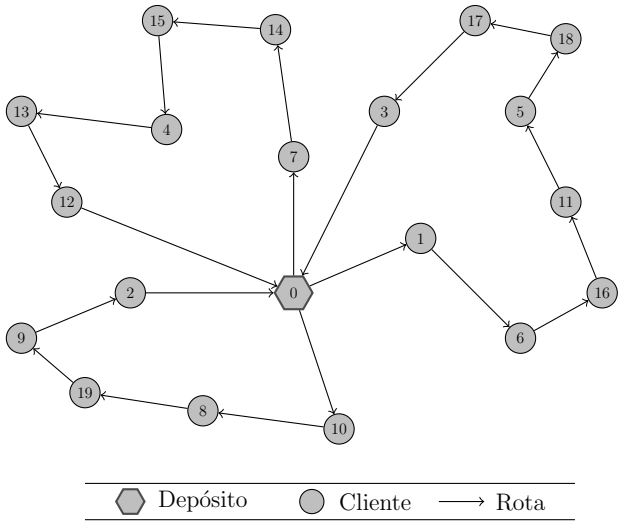

Figura 4.1: Exemplo de uma solução do PRVCES.

4.2 Estruturas de vizinhança

Para explorar o espaço de soluções do problema, aplicam-se, neste trabalho, sete tipos diferentes de movimentos, os quais são apresentados a seguir. É importante destacar que não são permitidos movimentos que conduzam a soluções inviáveis.

4.2.1 Movimento *Shift*

O *Shift* é um movimento de realocação que consiste em transferir um cliente i de uma rota para outra. A Figura 4.2 ilustra um exemplo em que o cliente 6 é transferido da Rota 2 à Rota 3.

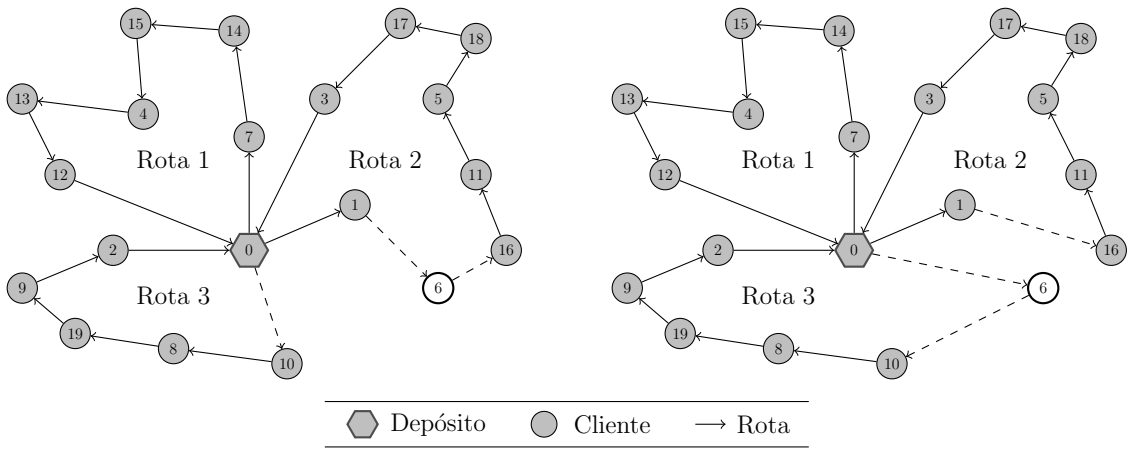

Figura 4.2: Exemplo do movimento *Shift*.

4.2.2 Movimento *Shift*(2,0)

O *Shift*(2,0) é um movimento semelhante ao *Shift*, porém realocando dois clientes consecutivos de uma rota para outra. A Figura 4.3 exemplifica a realocação dos clientes 1 e 6 da Rota 2 para a Rota 3.

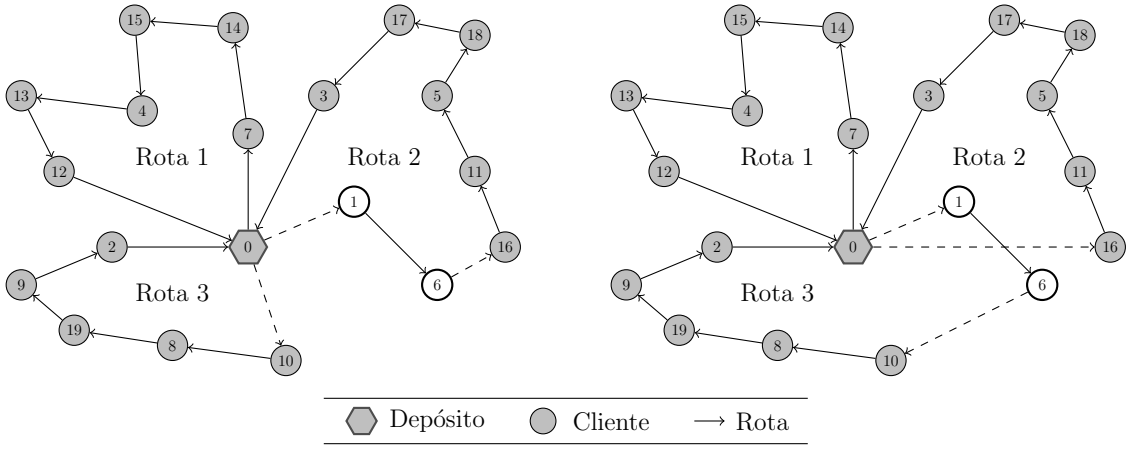

Figura 4.3: Exemplo do movimento *Shift*(2,0).

4.2.3 Movimento *Swap*

O movimento *Swap* consiste em trocar um cliente i de uma rota r_p com um outro cliente j de uma rota r_q. A Figura 4.4 mostra a aplicação do movimento *Swap* para os clientes 1 e 10 das rotas 2 e 3, respectivamente.

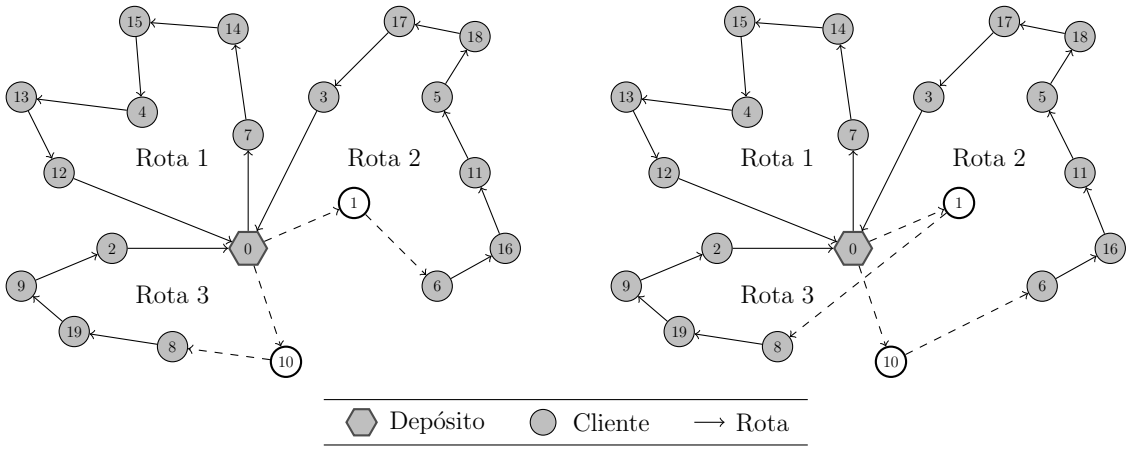

Figura 4.4: Exemplo do movimento *Swap*.

4.2.4 Movimento *Swap*(2,1)

O movimento *Swap*(2,1) é análogo ao *Swap*, porém trocando dois clientes consecutivos de uma rota com um cliente de outra rota. A Figura 4.5 mostra a aplicação do movimento *Swap*(2,1) considerando os clientes 9 e 2 da Rota 3 e o cliente 12 da Rota 1.

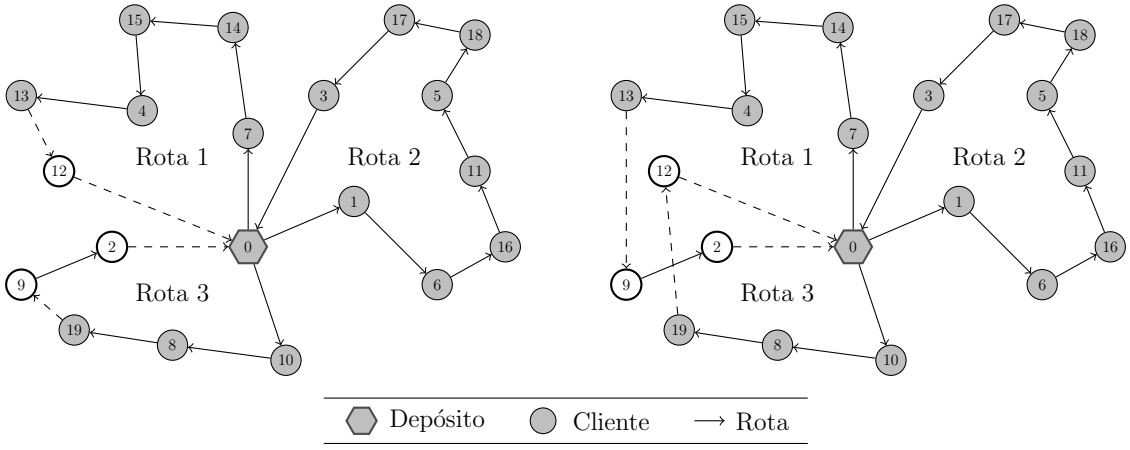

Figura 4.5: Exemplo do movimento *Swap*(2,1).

4.2.5 Movimento *Swap*(2,2)

O movimento *Swap*(2,2) é análogo ao *Swap*, porém trocando dois clientes consecutivos de uma rota com dois clientes de outra rota. A aplicação do movimento *Swap*(2,2) para os clientes 13 e 12 da Rota 1 e 9 e 2 da Rota 3 é ilustrado na Figura 4.6.

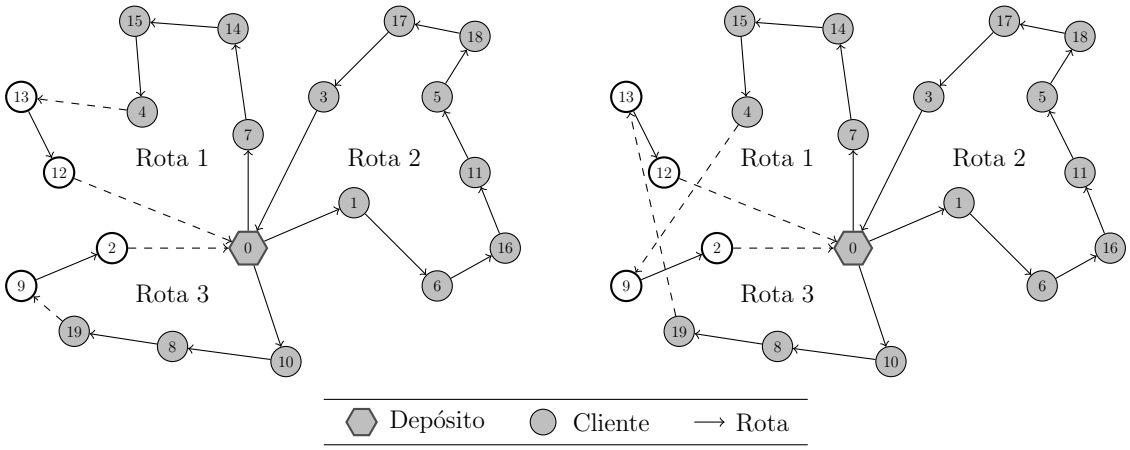

Figura 4.6: Exemplo do movimento *Swap*(2,2).

4.2.6 Movimento *2-Opt*

O *2-Opt* consiste em remover dois arcos e inserir dois novos arcos. A Figura 4.7 mostra a remoção dos arcos $(1, 6)$ e $(0, 10)$ e a inserção dos arcos $(1, 10)$ e $(0, 6)$.

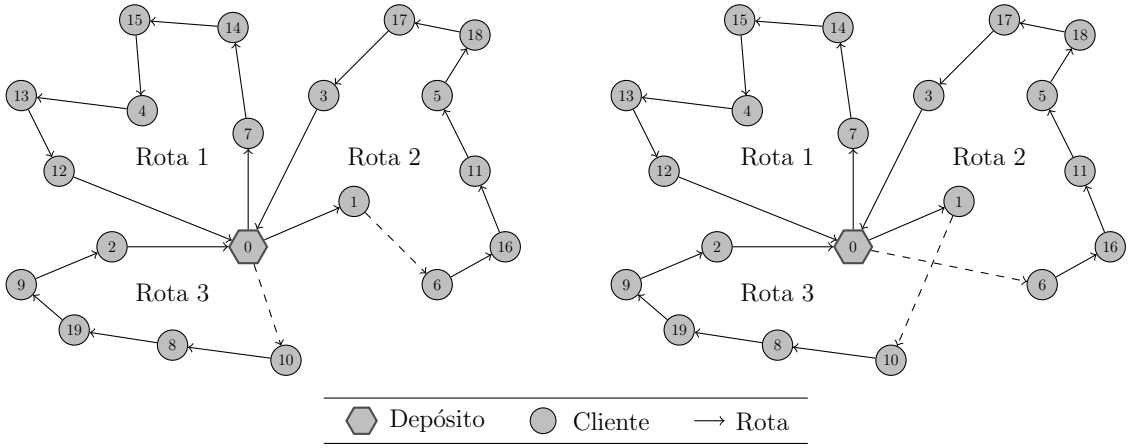

Figura 4.7: Exemplo do movimento *2-Opt*.

4.2.7 Movimento *kOr-Opt*

O movimento *kOr-Opt* consiste em remover k clientes consecutivos de uma rota r e, em seguida, reinserí-los em uma outra posição nessa mesma rota. O valor de k é um parâmetro do movimento. Esse movimento é uma generalização do movimento *Or-Opt* proposto por [Or, 1976], em que é realizado a remoção de no máximo três clientes consecutivos. A Figura 4.8 ilusta a aplicação do movimento *kOr-Opt* ($k = 3$), considerando a reinserção dos clientes 18, 17 e 3 da Rota 1 na posição do arco $(11, 5)$.

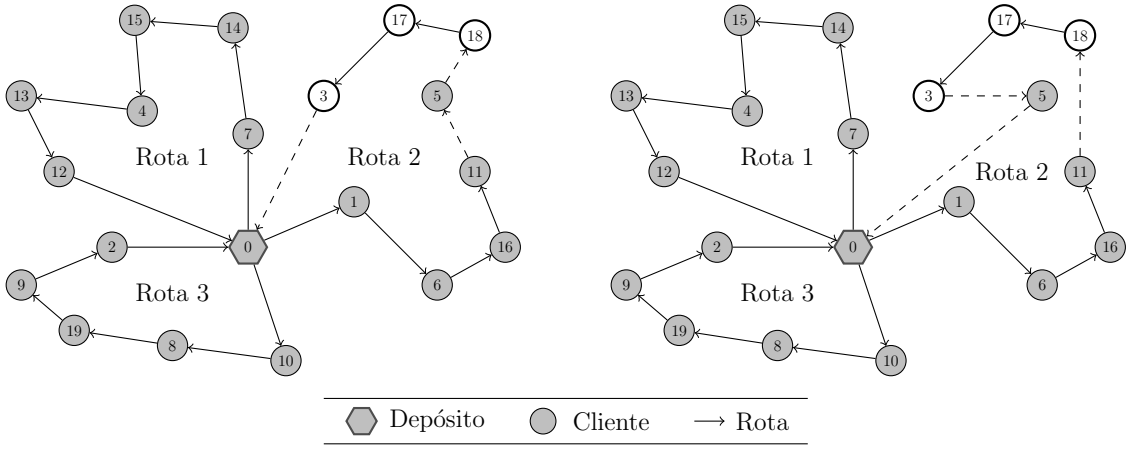

Figura 4.8: Exemplo do movimento *kOrOpt*.

4.3 Função de avaliação

Uma solução s é avaliada pela função f apresentada pela Equação 4.1, que determina os custos totais de deslocamento.

$$f(s) = \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{4.1}$$

em que:

- N^+ : conjunto dos clientes, incluindo o depósito;
- A : conjunto dos arcos (i, j), com $i, j \in N^+$;
- c_{ij} : custo de deslocamento ou distância de um cliente i a um cliente j $(i, j \in N^+)$;
- x_{ij} : indica se o arco $(i, j) \in A$ é utilizado $(x_{ij} = 1)$ ou não $(x_{ij} = 0)$ na solução.

4.4 Geração de uma solução inicial

Nesta seção são apresentados três procedimentos heurísticos para a geração de uma solução inicial para o PRVCES. Nas Subseções 4.4.1 e 4.4.2 são descritas, respectivamente, as heurísticas de Inserção Mais Barata considerando a construção rota a rota (IMB-1R) e com múltiplas rotas simultaneamente (IMB-MR). Na Subseção 4.4.3 é proposta uma adaptação da heurística GENIUS [Gendreau *et al.*, 1992], descrita na Seção 3.3.1 e denominada VRGENIUS, para o PRVCES.

4.4.1 Inserção Mais Barata com construção rota a rota

Este método, denominado IMB-1R, consiste em gerar uma solução inicial s, construindo uma rota a cada passo. Inicialmente, gera-se uma rota r contendo um cliente escolhido aleatoriamente. Em seguida, calcula-se o custo de inserção e^k_{ij}, expresso pela equação 4.2, de cada cliente k, que ainda não está presente na solução, entre cada par de clientes i e j da rota r.

$$e^k_{ij} = (c_{ik} + c_{kj} - c_{ij}) - \gamma(c_{0k} + c_{k0}) \tag{4.2}$$

Nessa função, a primeira parcela refere-se ao custo de inserção de um cliente k entre os clientes i e j e a segunda parcela corresponde a uma bonificação dada a um cliente que situa-se distante do depósito. Essa bonificação é controlada por um fator $\gamma \in [0, 1]$ e favorece a inserção de um cliente, de forma a não adicioná-lo tardiamente à rota. Detalhes sobre a influência do parâmetro γ podem ser encontrados em [Subramanian, 2008].

O cliente que tiver o menor custo de inserção é adicionado à rota, desde que a inserção deste não viole a restrição de capacidade do veículo. Caso a inserção de qualquer cliente implique na sobrecarga do veículo, então a rota corrente é finalizada e inicia-se a construção de uma nova rota. Esse procedimento é repetido até que todos os clientes sejam adicionados à solução.

Para melhor entendimento dessa heurística, considere o exemplo apresentado na Figura 4.9. Neste exemplo existem 19 clientes e uma frota ilimitada de veículos de capacidade $Q = 120$.

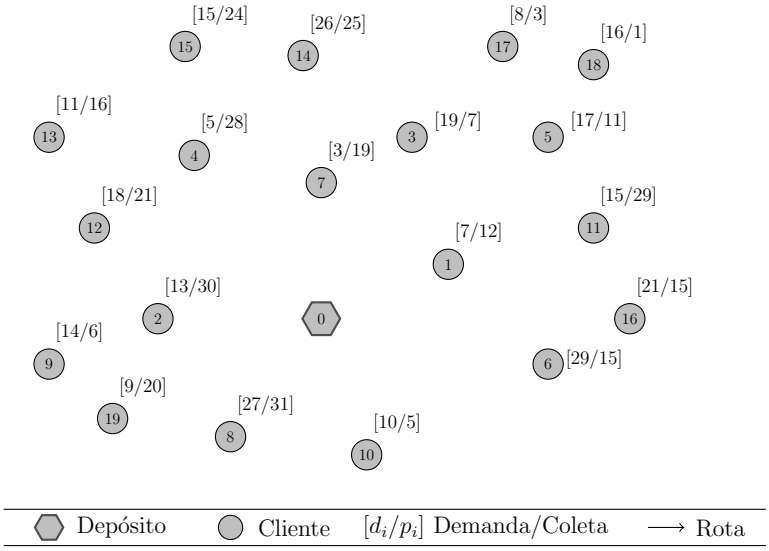

Figura 4.9: Exemplo de um problema envolvendo 19 clientes.

Inicialmente, deve-se gerar uma rota contendo um cliente escolhido aleatoriamente, no caso, o cliente 1 conforme apresentado na Figura 4.10.

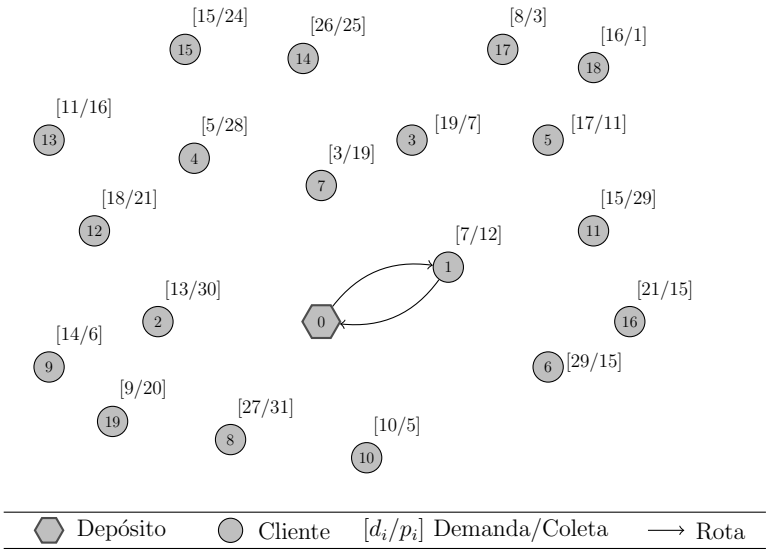

Figura 4.10: Construção de uma rota com o cliente 1.

Em seguida, calcula-se o custo de inserção dos demais clientes em todas as posições da rota. Por exemplo, o custo de inserção do cliente 7 entre o depósito e o cliente 1 é de $e_{01}^{7} = (c_{07} + c_{71} - c_{07}) - \gamma(c_{07} + c_{70})$. Como, neste exemplo, esse custo é o menor, então deve-se inserir o cliente 7 entre os clientes 0 e 1. A Figura 4.11 mostra essa inserção.

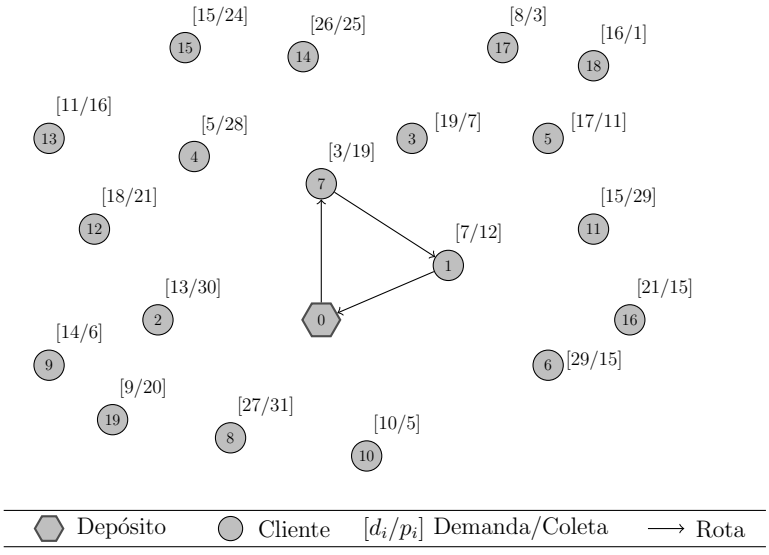

Figura 4.11: Inserção do cliente 7 na rota.

Continuando com a inserção dos clientes, chega-se à situação apresentada na Figura 4.12.

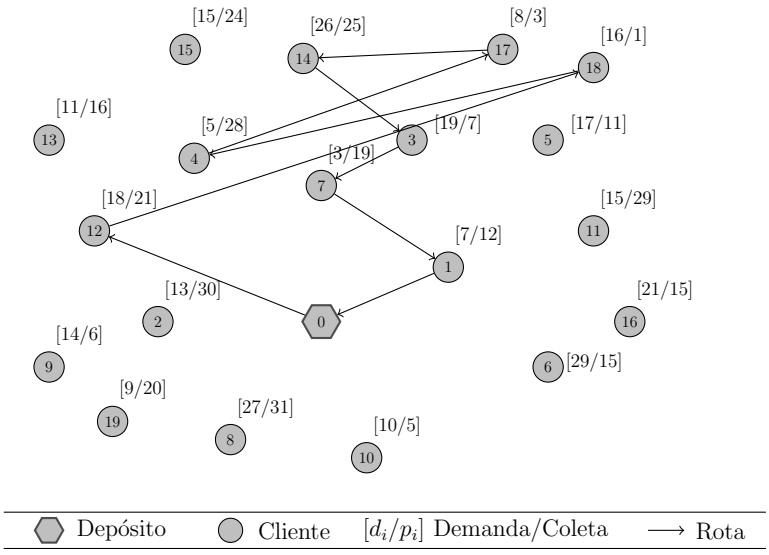

Figura 4.12: Construção completa de uma rota.

Na Figura 4.12 observa-se que a inserção de qualquer cliente na rota resultará na sobrecarga do veículo. Dessa forma, essa rota é finalizada e inicia-se a construção de uma nova rota. O procedimento é repetido até que todos os clientes sejam adicionados à solução. A Figura 4.13 mostra a solução final obtida pelo método.

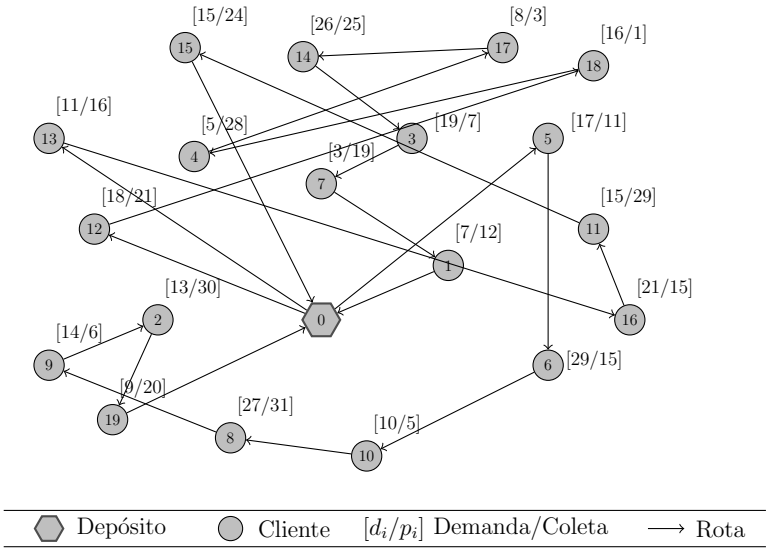

Figura 4.13: Solução gerada pela heurística de inserção mais barata rota a rota.

4.4.2 Inserção Mais Barata com construção simultânea de múltiplas rotas

Esse procedimento construtivo, nomeado IMB-MR, foi proposto por [Subramanian, 2008] e baseia-se em uma adaptação do método de inserção proposto por [Dethloff, 2001], porém sem considerar a capacidade residual do veículo.

Seja m um parâmetro do método. Inicialmente, são construídas m rotas com um único cliente escolhido de forma aleatótia. Em seguida, adiciona-se um cliente k entre os clientes i e j de uma rota r, tal que o custo de inserção e_{ij}^k, expresso pela Função 4.2 da Subseção 4.4.1, seja o menor possível. Esse procedimento é executado iterativamente até que não haja mais clientes a serem inseridos. É importante destacar que a inserção de um cliente só é realizada se não houver a sobrecarga do veículo na rota considerada.

Para facilitar o entendimento dessa heurística, considere o problema apresentado na Figura 4.9. Inicialmente, deve-se construir k rotas contendo um cliente aleatório. A Figura 4.14 mostra a etapa inicial do método considerando três rotas, envolvendo os clientes 8, 11 e 13. Neste exemplo, considera-se que o valor de γ seja igual a $0,10$.

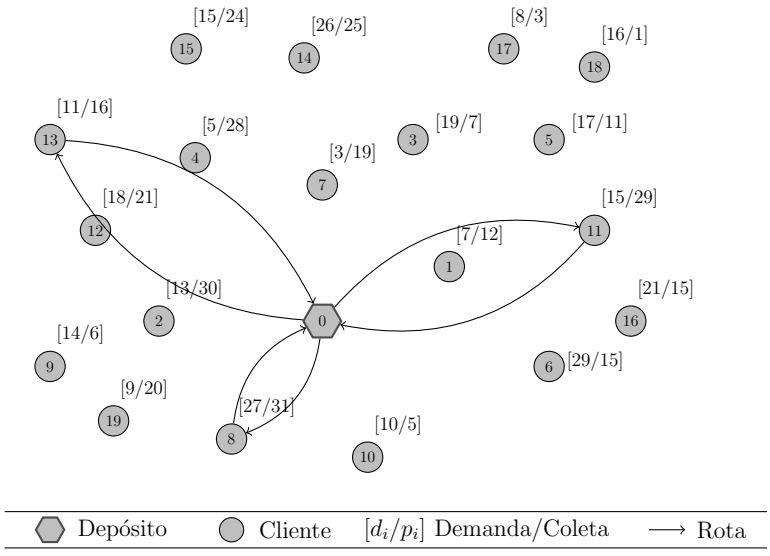

Figura 4.14: Etapa inicial do método, considerando três rotas.

Em seguida, deve-se calcular o custo de inserção de cada cliente que ainda não esteja presente na solução, em todas as posições de todas as rotas. Nesse exemplo, a inserção de menor custo é a do cliente 12 entre o depósito e o cliente 13. A Figura 4.15 ilustra essa inserção.

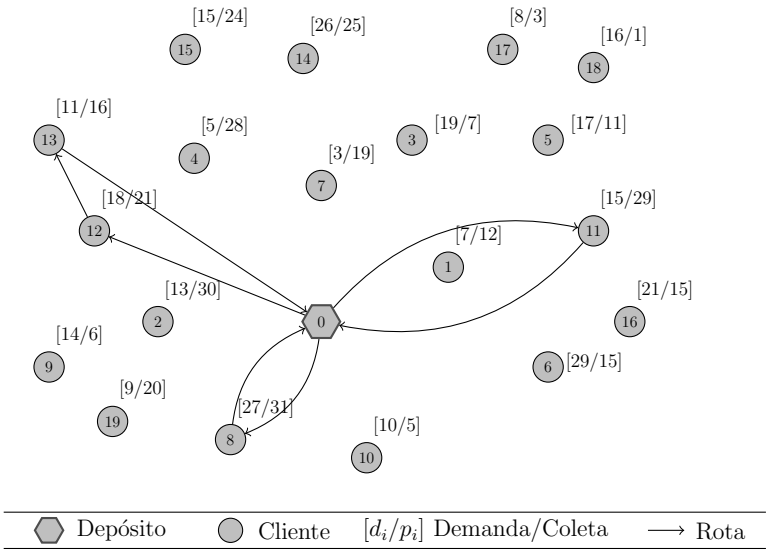

Figura 4.15: Inserção do cliente 12 entre o depósito e o cliente 13.

Após a inserção do cliente 12, deve-se novamente calcular os custos de inserção de cada cliente. Na Figura 4.16 observa-se a inserção do cliente 1 entre o depósito e o cliente 11.

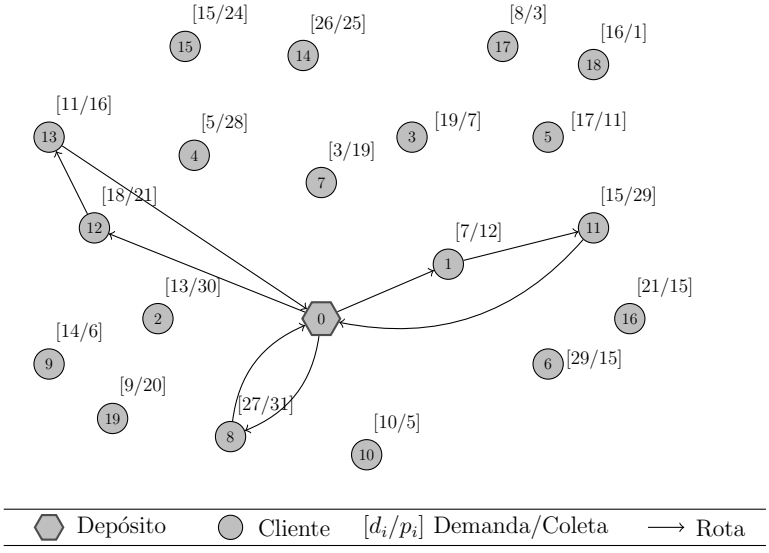

Figura 4.16: Inserção do cliente 1 entre o depósito e o cliente 11.

Continuando com esse procedimento, obtém-se a solução apresentada na Figura 4.17.

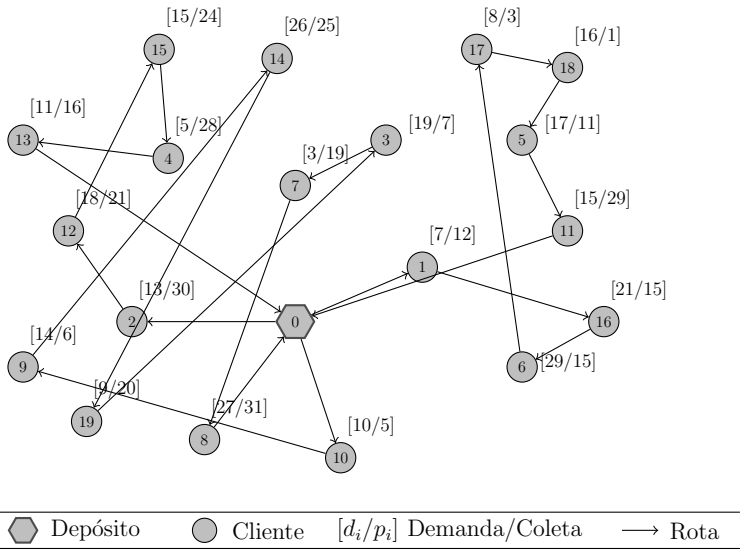

Figura 4.17: Solução gerada pelo método de inserção mais barata com múltiplas rotas.

Um observação importante é que esta heurística pode gerar uma solução incompleta e, portanto, inviável. Isso é devido à dependência do parâmetro k, que indica quantas rotas deve-se construir no início do procedimento. Se o número de rotas não for suficiente, então não será possível gerar uma solução viável. Um exemplo dessa situação pode ser observada na Figura 4.18, no qual foram utilizadas apenas duas rotas, considerando o problema apresentado na Figura 4.9.

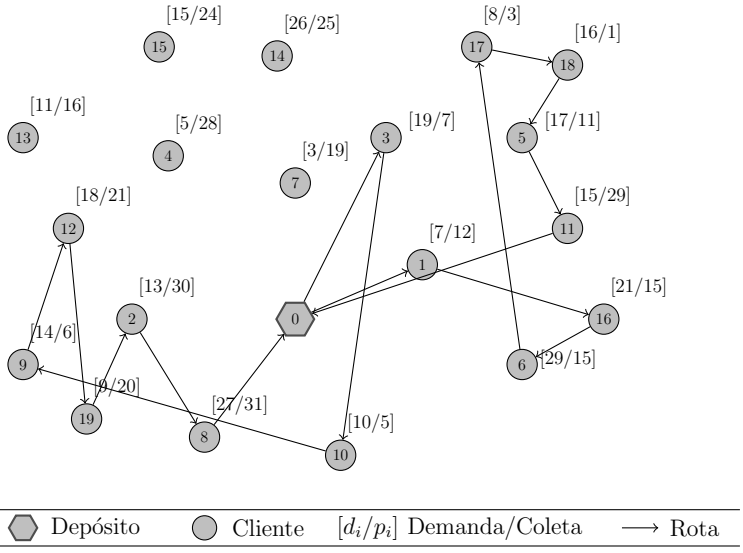

Figura 4.18: Exemplo da geração de uma solução incompleta.

Para contornar esse problema, o método IMB-MR foi adaptado, combinando-o com a heurística IMB-1R, apresentada na Subseção 4.4.1. Inicialmente, gera-se uma solução,

eventualmente parcial, com o método IMB-MR. Caso essa solução seja incompleta, então continua-se a construção rota a rota por meio da heurística IMB-1R. A Figura 4.19 mostra a solução viável gerada pela aplicação desta adaptação à solução incompleta da Figura 4.18.

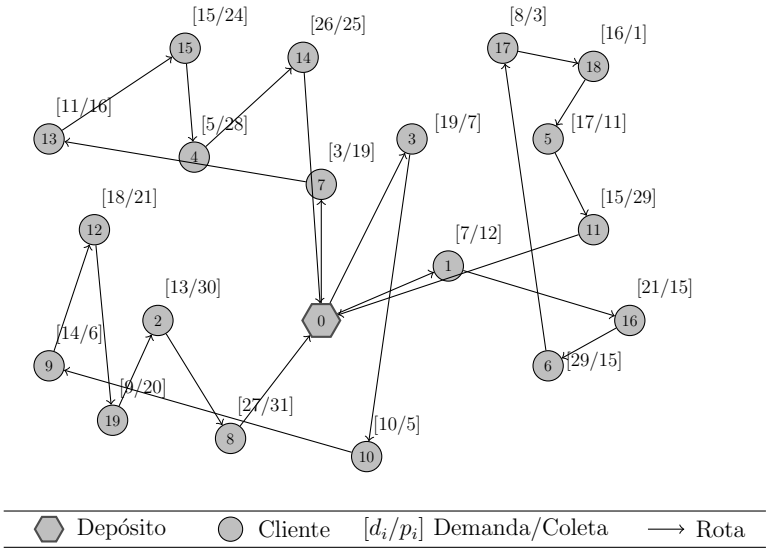

Figura 4.19: Solução gerada pela adaptação do método IMB-MR.

4.4.3 Procedimento construtivo VRGENIUS

O terceiro procedimento utilizado para construir uma solução inicial para o PRVCES é o VRGENIUS. Este procedimento, desenvolvido neste trabalho, é uma adaptação da heurística GENIUS, descrita na Subseção 3.3.1. As Subseções 4.4.3.1 e 4.4.3.2 descrevem, respectivamente, as adaptações das fases GENI e US da heurística GENIUS.

4.4.3.1 Fase GENI adaptada ao PRVCES

Inicialmente, são construídas m rotas, onde m é um parâmetro do método, contendo duas cidades além do depósito. Essas cidades são escolhidas de forma aleatória. A cada iteração, seleciona-se um vértice $v \in V^-$ de forma arbitrária, onde V^- representa o conjunto de cidades que não estão na solução. Em seguida, calcula-se o custo de inserção de v em cada posição de cada rota, por meio dos métodos IT1 e IT2 descritos na Subseção 3.3.1.1. O vértice v será inserido na posição, cujo custo de adição seja mínimo. Vale ressaltar que v é inserido somente em uma posição que não viole a restrição de capacidade do veículo. O método pára quando todos os vértices estiverem presentes na solução. Se o número de rotas não for suficiente para gerar uma solução viável, então esse número é incrementado

em uma unidade e a fase GENI é executada novamente.

O Algoritmo 12 apresenta o pseudocódigo da Fase GENI adaptada para o PRVCES, denominada VRGENI.

Algoritmo 12 Fase construtiva VRGENI

```
1: Seja m_0 o número inicial de rotas:
2: m ← m_0;
3: V⁻ ← V, onde V é o conjunto de cidades;
4: enquanto ( |V⁻| > 0 ) faça
5:    s ← ∅;
6:    m' ← 1;
7:    enquanto ( m' ≤ m ) faça
8:       Selecione, aleatoriamente, duas cidades v', v'' ∈ V⁻;
9:       r ← rota contendo o depósito e as cidades v' e v'';
10:      s ← s ∪ {r};
11:      V⁻ = V⁻ \ {v'};
12:      V⁻ = V⁻ \ {v''};
13:      m' ← m' + 1;
14:   fim enquanto
15:   V' = ∅;
16:   enquanto ( |V⁻| > 0 ) faça
17:      Selecione, aleatoriamente, um vértice v ∈ V⁻;
18:      s' ← InserçãoGENI(v,s);  { Algoritmo 5 }
19:      se ( existe um arco ∈ s' tal que sua carga exceda a capacidade do veículo ) então
20:         V' = V' ∪ {v};
21:      senão
22:         s ← s';
23:         V⁻ = V⁻ ∪ V';
24:         V' = ∅;
25:      fim se
26:      V⁻ = V⁻ \ {v};
27:   fim enquanto
28:   se ( |V'| > 0 ) então
29:      s ← ∅;
30:      V⁻ ← V;
31:      m ← m + 1;
32:   fim se
33: fim enquanto
34: Retorne s;
```

Basicamente, a fase GENI adaptada ao PRVCES segue a mesma idéia da construção da IMB-MR. A principal diferença é que a VRGENI pode inserir uma cidade entre duas outras não consecutivas, por meio das inserções IT1 e IT2.

A Figura 4.20 apresenta a solução obtida pela fase VRGENI.

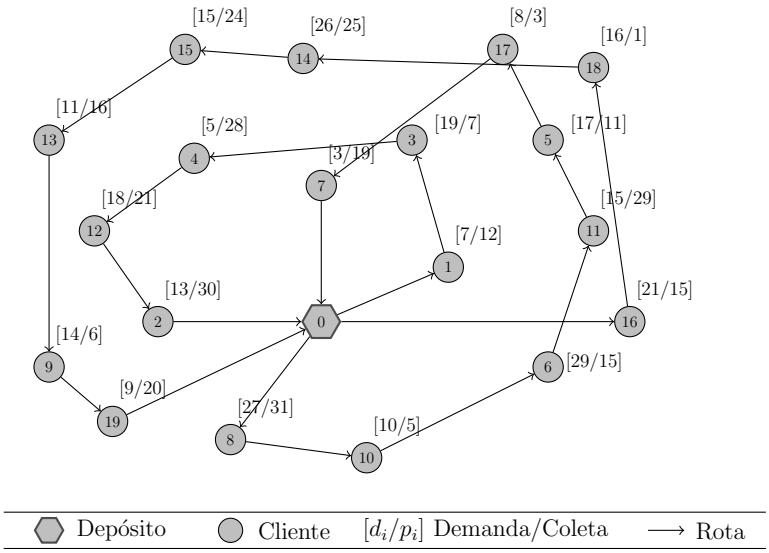

Figura 4.20: Solução gerada pela fase VRGENI.

4.4.3.2 Fase US adaptada ao PRVCES

A Fase US adaptada ao PRVCES, denominada VRUS, consiste em, a cada iteração, remover uma cidade e reinserí-la em uma outra posição na mesma rota ou em outra rota. A exclusão é realizada por meio das remoções UST1 e UST2, descritas na Subseção 3.3.1.2 e a adição é feita por meio das inserções IT1 e IT2 descritas na Subseção 3.3.1.1.

O pseudocódigo da fase US adaptada ao PRVCES é apresentado no Algoritmo 13. Nesse algoritmo, considere que p_z^r é o z-ésimo vértice da rota $r \in R(s)$ a ser visitado, onde $R(s)$ é o conjunto de rotas da solução s. Dessa forma, p_1^r e p_n^r representam, respectivamente, as posições do primeiro e do último vértice da rota r a ser visitado. Além disso, considere que $v(p_z^r, s)$ é o vértice que se encontra na posição z da rota r na solução s.

O algoritmo VRUS realiza, a cada iteração, a remoção de um vértice $\bar{v}_i$ da rota r na solução s, que se encontra na posição p_z^r, por meio das duas remoções RT1 e RT2. Em seguida, o vértice $\bar{v}_i$ é adicionado com a configuração das inserções IT1 e IT2 da fase GENI a qual resulta no melhor custo de inserção, considerando todas as rotas da solução s. Se a solução gerada for melhor que a melhor solução encontrada (s^*), então o próximo vértice a ser considerado será o da primeira posição da primeira rota de s. Caso contrário, o algoritmo avança para o vértice da próxima posição p_{z+1}^r. Esse procedimento é realizado até que todos os vértices sejam analisados.

Algoritmo 13 Fase de refinamento VRUS

```
1: Seja s uma solução inicial:
2: s* ← s;
3: para ( r ∈ R(s) ) faça
4:    p_z^r ← p_1^r;
5:    enquanto ( p_z^r ≤ p_n^r ) faça
6:       v ← v(p_z^r, s);
7:       s' ← RemoçãoUS(v,r);  { Algoritmo 9 }
8:       s'' ← InserçãoGENI(v,s');  { Algoritmo 5 }
9:       se ( f(s'') < f(s*) ) então
10:         s* ← s'';
11:         r ← primeira rota de R(s);
12:         p_z^r ← p_1^r;
13:      senão
14:         p_z^r ← p_{z+1}^r;
15:      fim se
16:      s ← s'';
17:   fim enquanto
18: fim para
19: s ← s*;
20: Retorne s;
```

A Figura 4.21 mostra a solução gerada pela fase US adapatada ao PRVCES, considerando o exemplo da solução inicial gerada pela fase VRGENI (Figura 4.20). Como a solução inicial foi gerada pela fase VRGENI, então esta figura representa a solução final obtida pela heurística GENIUS adapatada ao PRVCES.

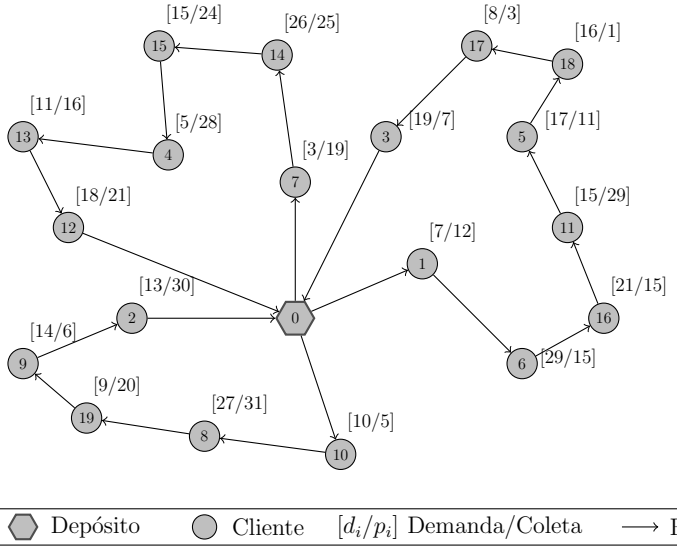

Figura 4.21: Solução gerada pela fase VRUS e pela heurística VRGENIUS.

4.5 *Iterated Local Search* aplicado ao PRVCES

Nesta seção mostra-se a adaptação da metaheurística *Iterated Local Search* ao PRVCES. O algoritmo proposto, denominado GENILS, segue a mesma idéia do ILS básico, descrito na Subseção 3.4.2. O Algoritmo 14 apresenta o seu pseudocódigo.

Algoritmo 14 *GENILS*

1: $\gamma \leftarrow$ valor aleatório do intervalo $[0.0; 0.7]$;
2: $s^A \leftarrow$ *InserçãoMaisBarataRotaARota*(γ); {método descrito na Subseção 4.4.1}
3: $s^A \leftarrow$ *VND*(s^A);
4: *nRotas* $\leftarrow$ número de rotas da solução s^A;
5: $s^B \leftarrow$ *InserçãoMaisBarataMRotas*(γ, *nRotas*); {método descrito na Subseção 4.4.2}
6: $s^B \leftarrow$ *VND*(s^B);
7: $s^C \leftarrow$ *VRGENIUS*(*nRotas*); {método descrito na Subseção 4.4.3}
8: $s^C \leftarrow$ *VND*(s^C);
9: $s \leftarrow s_0 \mid f(s_0) = min(f(s^A), f(s^B), f(s^C))$;
10: $iter \leftarrow 1$;
11: **enquanto** ($iter \leq maxIter$) **faça**
12: $s' \leftarrow$ *Perturbação*(s);
13: $s'' \leftarrow$ *VND*(s');
14: **se** ($f(s'') < f(s)$) **então**
15: $s \leftarrow s''$;
16: $iter \leftarrow 1$;
17: **senão**
18: $iter \leftarrow iter + 1$;
19: **fim se**
20: **fim enquanto**
21: Retorne s;

O GENILS inicia-se gerando três soluções iniciais s^A, s^B e s^C, cada qual por uma heurística construtiva distinta. Essas heurísticas são as descritas na Seção 4.4. O número de rotas necessárias para as heurísticas de Inserção Mais Barata com múltiplas rotas (IMB-MR) e VRGENIUS é determinado pelo método de Inserção Mais Barata rota a rota (IMB-1R). O parâmetro γ é escolhido aletoriamente no intervalo $[0,0; 0,7]$. Esse intervalo foi determinado com base em uma bateria preliminar de testes. As perturbações são realizadas por meio dos mecanismos descritos na Seção 4.5.1 e, para refinar uma solução, utiliza-se o método VND, descrito na Seção 4.6. O método é interrompido quando o número máximo de iterações sem melhora na solução corrente ($maxIter$) é atingido.

4.5.1 Mecanismos de perturbação

Para realizar uma perturbação na solução corrente, o GENILS seleciona um dos três mecanismos abaixo de forma aleatória:

- Múltiplos *Shift*: Consiste em realizar k movimentos *Shift* (descrito na Subseção 4.2.1) sucessivamente a cada chamada desse mecanismo. O valor de k é definido aleatoriamente entre 1, 2 ou 3;
- Múltiplos *Swap*: Segue a mesma idéia da perturbação com múltiplos *Shift*, porém utilizando movimentos *Swap* (descrito na Subseção 4.2.3);
- *Ejection chain*: Essa perturbação foi proposta por [Rego and Roucairol, 1996]. Inicialmente, seleciona-se um subconjunto de m rotas $R = \{r_1, r_2, \ldots, r_m\}$ de forma arbitrária. Em seguida, transfere-se um cliente da rota r_1 para a rota r_2, logo após, um cliente de r_2 para r_3 e assim sucessivamente até que um cliente seja tranferido da rota r_m para a primeira rota r_1. Nesse movimento, os clientes são escolhidos de forma aleatória.

4.6 Busca Local do GENILS

Uma solução do GENILS é refinada pelo método VND, conforme descrito na Seção 3.4.1, com duas estratégias adicionais. A primeira consiste em definir, aleatoriamente, uma ordem das vizinhanças a serem exploradas. O conjunto N de vizinhanças utilizadas são as descritas na Seção 4.2. A outra estratégia é a de intensificação da busca nas rotas modificadas em cada iteração do método. Essa intensificação é realizada por meio de procedimentos de busca local baseados nos movimentos *Shift*, *Shift20*, *Swap*, *2-opt*, *Swap21*, *Swap22* e *kOr-Opt* com $k = 3, 4, 5$, apresentados na Seção 4.2. Além dessas buscas, a intensificação é realizada também pelos procedimentos *G3-Opt* e *G4-Opt*, descritos na Subseção 4.6.1, e *Reverse*, descrito na Subseção 4.6.2. O Algoritmo 15 mostra o pseudocódigo do VND, utilizado como método de busca local do GENILS.

Algoritmo 15 Descida em Vizinhança Variável

Seja r o número de estruturas de vizinhança distintas;
$RN \leftarrow$ conjunto das vizinhanças N, descritas na Seção 4.2, em ordem aleatória;
$k \leftarrow 1$;
enquanto ($k \leq r$) **faça**
  Encontre o melhor vizinho $s' \in RN^{(k)}(s)$;
  se ($f(s') < f(s)$) **então**
    $s \leftarrow s'$;
    $k \leftarrow 1$;
    {Intensificação nas rotas alteradas}
    $s \leftarrow$ *BuscaLocalShift*(s);
    $s \leftarrow$ *BuscaLocalShift20*(s);
    $s \leftarrow$ *BuscaLocalSwap*(s);
    $s \leftarrow$ *BuscaLocal2-Opt*(s);
    $s \leftarrow$ *BuscaLocalSwap21*(s);
    $s \leftarrow$ *BuscaLocalSwap22*(s);
    $s \leftarrow$ *BuscaLocalG3-Opt*(s);
    $s \leftarrow$ *BuscaLocalG4-Opt*(s);
    $s \leftarrow$ *BuscaLocalkOr-Opt*(s) com $k = 3, 4, 5$;
    $s \leftarrow$ *Reverse*(s);
  senão
    $k \leftarrow k + 1$;
  fim se
fim enquanto
Retorne s;

4.6.1 Procedimentos *G3-Opt* e *G4-Opt*

Os procedimentos *G3-Opt* e *G4-Opt* são adaptações das buscas locais *3-optimal* e *4-optimal*, respectivamente. Eles exploram um espaço reduzido de soluções e seguem a idéia da heurística GENIUS, descrita na Subseção 3.3.1. Essa idéia consiste em analisar a inserção de um arco (v_a, v_b) somente se os clientes v_a e v_b estiverem relativamente próximos. Para isso, define-se $N_p(v)$ como o conjunto dos p vizinhos mais próximos ao cliente v em uma rota r da solução s, sendo p um parâmetro. Além disso, considere as seguintes definições:

- N^r: conjunto dos clientes pertencentes à rota r;
- v_i: cliente $v_i \in N^r$;
- v_{h+1}, v_{h-1}: clientes, pertencentes à rota r, sucessor e antecessor ao cliente $v_h \in N^r$, respectivamente;
- v_j: cliente $j \in N_p(v_i)$;

- v_k: cliente $k \in N_p(v_{i+1})$ no caminho de v_j para v_i;
- v_l: cliente $l \in N_p(v_{j+1})$ no caminho de v_i para v_j;
- $\bar{r}$: rota r no sentido inverso;

O procedimento *G3-Opt* funciona da seguinte forma: a cada passo, é feita a remoção dos arcos (v_i, v_{i+1}), (v_j, v_{j+1}) e (v_k, v_{k+1}) e a inserção os arcos (v_i, v_j), (v_{i+1}, v_k) e (v_{j+1}, v_{k+1}) na rota r, de forma a melhorar a solução s e tal que o custo seja o menor possível. Ressalta-se que ambos os sentidos da rota r são analisados. Este procedimento é repetido até que não seja possível melhorar a solução s. O pseudocódigo do *G3-Opt* é mostrado no Algoritmo 16.

Algoritmo 16 *G3-Opt*

Seja r uma rota da solução s;
$r' \leftarrow \emptyset \Rightarrow f(r') \leftarrow \infty$;
enquanto ($f(r) \leq f(r')$) **faça**
  $r' \leftarrow r$;
  para ($r'' \in \{ r', \bar{r'} \}$) **faça**
    para ($v_i \in N^{r'}$) **faça**
      para ($v_j \in N_p(v_i), v_j \neq v_i$) **faça**
        para ($v_k \in N_p(v_{i+1}), v_k \neq v_i, v_j$) **faça**
          $r''' \leftarrow r'' \setminus \{(v_i, v_{i+1}), (v_j, v_{j+1}), (v_k, v_{k+1})\}$;
          $r''' \leftarrow r''' \cup \{(v_i, v_j), (v_{i+1}, v_k), (v_{j+1}, v_{k+1})\}$;
          se ($f(r''') < f(r)$) **então**
            $r \leftarrow r'''$;
          fim se
        fim para
      fim para
    fim para
  fim para
fim enquanto
Retorne s;

Já o procedimento *G4-Opt* é semelhante ao *G3-Opt*, com a diferença de que, a cada iteração, são removidos os arcos (v_i, v_{i+1}), (v_{l-1}, v_l), (v_j, v_{j+1}) e (v_{k-1}, v_k) e adicionados os arcos (v_i, v_j), (v_l, v_{j+1}), (v_{k-1}, v_{l-1}) e (v_{i+1}, v_k). O Algoritmo 17 apresenta o pseudocódigo do *G4-Opt*.

Algoritmo 17 *G4-Opt*

Seja r uma rota da solução s;
$r' \leftarrow \emptyset \Rightarrow f(r') \leftarrow \infty$;
enquanto ($f(r) \leq f(r')$) **faça**
  $r' \leftarrow r$;
  para ($r'' \in \{ r', \bar{r'} \}$) **faça**
    para ($v_i \in N^{r'}$) **faça**
      para ($v_j \in N_p(v_i),\ v_j \neq v_i, v_{i+1}$) **faça**
        para ($v_k \in N_p(v_{i+1}),\ v_k \neq v_j, v_{j+1}$) **faça**
          para ($v_l \in N_p(v_{j+1}),\ v_l \neq v_i, v_{i+1}$) **faça**
            $r''' \leftarrow r'' \setminus \{(v_i, v_{i+1}), (v_{l-1}, v_l), (v_j, v_{j+1}), (v_{k-1}, v_k)\}$;
            $r''' \leftarrow r''' \cup \{(v_i, v_j), (v_l, v_{j+1}), (v_{k-1}, v_{l-1}), (v_{i+1}, v_k)\}$;
            se ($f(r''') < f(r)$) **então**
              $r \leftarrow r'''$;
            fim se
          fim para
        fim para
      fim para
    fim para
  fim para
fim enquanto
Retorne s;

4.6.2 Procedimento *Reverse*

O procedimento *Reverse* consiste em inverter o sentido de uma rota r, sendo aplicado somente se ocorrer um aumento na carga residual da rota. A carga residual de uma rota é o valor da capacidade do veículo subtraído da maior carga do veículo nessa rota. A Figura 4.22 mostra a aplicação do *Reverse* na Rota 2.

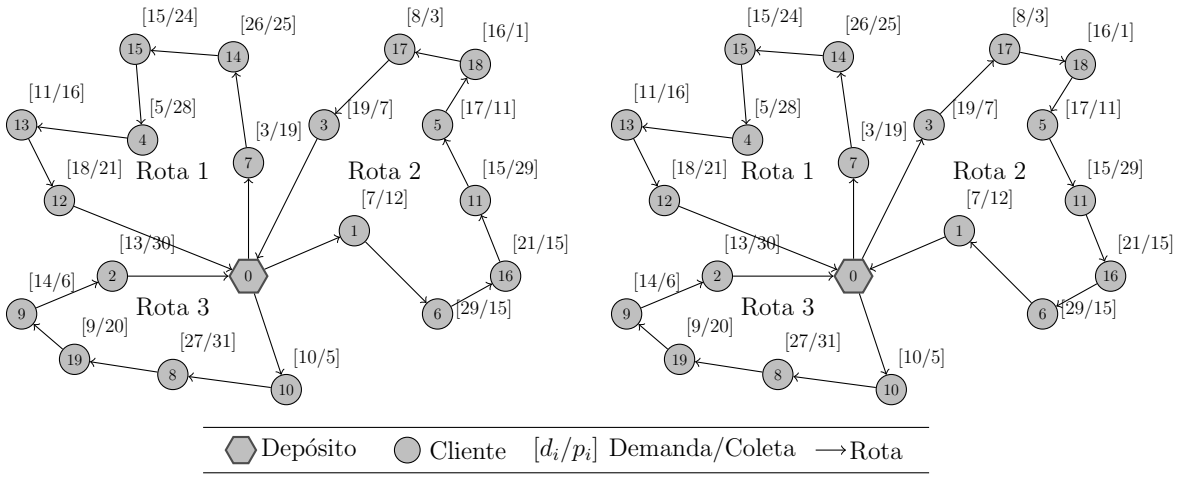

Figura 4.22: Aplicação do procedimento *Reverse* na Rota 2.

Nessa Figura, o valor da carga residual na Rota 2 aumentou de 13 para 18, considerando que a capacidade do veículo é de 150.

Capítulo 5

Resultados Computacionais

Apresentam-se, neste capítulo, os resultados computacionais obtidos pelo algoritmo GENILS proposto para resolver o PRVCES. O sistema foi desenvolvido na linguagem C++, utilizando o ambiente Microsoft Visual C++, versão 2005. Para testá-lo, foi utilizado um computador Intel Core 2 Duo com 1,66 GHz e 2 GB de memória RAM e sistema operacional Windows Vista Home Premium de 32 bits. Apesar de este equipamento contar com dois núcleos, não foi utilizada essa capacidade de multiprocessamento.

Para validar o algoritmo proposto, foram utilizados os problemas-teste propostos por [Salhi and Nagy, 1999], [Dethloff, 2001] e [Montané and Galvão, 2006], apresentados na Seção 3.1, com exceção dos problemas envolvendo restrições de limite de tempo. Para cada problema-teste foram realizadas 100 execuções, cada qual partindo de uma semente diferente de números aleatórios. Os parâmetros adotados para o GENILS (Subseção 4.5) foram número máximo de iterações ($maxIter$) igual a 10.000 e $p = 12$ o número de vizinhos analisados pelo procedimento construtivo VRGENIUS (Subseção 4.4.3) e pelos procedimentos de refinamento *G3-Opt* e *G4-Opt* (Subseção 4.6.1).

Nas Tabelas 5.1, 5.2 e 5.3, a coluna **#C** refere-se ao número de clientes, **#V** apresenta o número de veículos utilizados, **MelhorLit** refere-se ao melhor valor encontrado na literatura até então e **Melhor**, **Média** e **Tempo** são, respectivamente, o melhor valor encontrado pelo método GENILS, a média em 100 execuções e o tempo médio, dado em segundos. Por fim, a coluna **Gap** representa o desvio percentual da média em relação ao melhor valor da literatura, calculado pela equação dada em (5.1).

$$Gap = 100 \times \frac{Média - MelhorLit}{MelhorLit} \tag{5.1}$$

Tabela 5.1: Resultados para os problemas-teste de [Dethloff, 2001]

Problema	#C	#V	MelhorLit	Melhor	Média	Tempo (s)	Gap (%)
SCA3-0	50	4	**635,62**	**635,62**	638,89	9,46	0,51
SCA3-1	50	4	**697,84**	**697,84**	699,44	10,14	0,23
SCA3-2	50	4	**659,34**	**659,34**	**659,34**	9,06	0,00
SCA3-3	50	4	**680,04**	**680,04**	681,55	9,87	0,22
SCA3-4	50	4	**690,50**	**690,50**	**690,50**	9,27	0,00
SCA3-5	50	4	**659,90**	**659,90**	**659,90**	9,45	0,00
SCA3-6	50	4	**651,09**	**651,09**	**651,09**	10,03	0,00
SCA3-7	50	4	**659,17**	**659,17**	661,09	10,11	0,29
SCA3-8	50	4	**719,47**	**719,47**	720,38	11,14	0,13
SCA3-9	50	4	**681,00**	**681,00**	681,15	10,32	0,02
SCA8-0	50	9	**961,50**	**961,50**	968,02	7,54	0,68
SCA8-1	50	9	**1049,65**	**1049,65**	1057,50	7,86	0,75
SCA8-2	50	9	**1039,64**	**1039,64**	1049,67	8,83	0,97
SCA8-3	50	9	**983,34**	**983,34**	984,88	7,74	0,16
SCA8-4	50	9	**1065,49**	**1065,49**	1067,32	7,40	0,17
SCA8-5	50	9	**1027,08**	**1027,08**	1028,73	9,96	0,16
SCA8-6	50	9	**971,82**	**971,82**	972,80	8,11	0,10
SCA8-7	50	9	**1051,28**	**1051,28**	1060,40	9,50	0,87
SCA8-8	50	9	**1071,18**	**1071,18**	1077,18	7,42	0,56
SCA8-9	50	9	**1060,50**	**1060,50**	1065,10	7,25	0,43
CON3-0	50	4	**616,52**	**616,52**	618,89	9,46	0,38
CON3-1	50	4	**554,47**	**554,47**	556,02	9,78	0,28
CON3-2	50	4	**518,00**	**518,00**	522,41	11,12	0,85
CON3-3	50	4	**591,19**	**591,19**	**591,19**	9,83	0,00
CON3-4	50	4	**588,79**	**588,79**	591,60	9,88	0,48
CON3-5	50	4	**563,70**	**563,70**	566,75	11,05	0,54
CON3-6	50	4	**499,05**	**499,05**	502,56	10,41	0,70
CON3-7	50	4	**576,48**	**576,48**	580,48	8,70	0,69
CON3-8	50	4	**523,05**	**523,05**	523,49	8,10	0,08
CON3-9	50	4	**578,24**	**578,24**	583,77	11,51	0,96
CON8-0	50	9	**857,17**	**857,17**	859,97	9,71	0,33
CON8-1	50	9	**740,85**	**740,85**	742,78	9,21	0,26
CON8-2	50	9	**712,89**	**712,89**	713,15	9,20	0,04
CON8-3	50	9	**811,07**	**811,07**	817,91	7,93	0,84
CON8-4	50	9	**772,25**	**772,25**	772,98	6,75	0,10
CON8-5	50	9	**754,88**	**754,88**	757,20	7,77	0,31
CON8-6	50	9	**678,92**	**678,92**	684,38	9,37	0,80
CON8-7	50	9	**811,96**	**811,96**	**811,96**	4,55	0,00
CON8-8	50	9	**767,53**	**767,53**	769,68	8,35	0,28
CON8-9	50	9	**809,00**	**809,00**	810,55	7,77	0,19

Tabela 5.2: Resultados para os problemas-teste de [Salhi and Nagy, 1999]

Problema	#C	#V	MelhorLit	Melhor	Média	Tempo (s)	Gap (%)
CMT1X	50	3	**466,77**	**466,77**	472,23	11,37	1,17
CMT1Y	50	3	**458,96**	466,77	472,22	10,89	2,89
CMT2X	75	6	**668,77**	684,11	693,94	19,70	3,76
CMT2Y	75	6	**663,25**	684,11	693,58	20,97	4,57
CMX3X	100	5	**721,27**	721,40	726,30	60,60	0,70
CMT3Y	100	5	**721,27**	**721,27**	730,76	61,07	1,32
CMT12X	100	6	**644,70**	662,22	677,64	56,66	5,11
CMT12Y	100	6	**659,52**	663,50	678,85	45,50	2,93
CMT11X	125	4	**838,66**	846,23	875,66	84,20	4,41
CMT11Y	125	4	**830,29**	836,04	871,93	133,12	5,02
CMT4X	150	7	**852,46**	**852,46**	872,68	194,59	2,37
CMT4Y	150	7	**852,35**	862,28	878,79	149,12	3,10
CMT5X	199	10	**1030,55**	1033,51	1066,43	410,52	3,48
CMT5Y	199	10	**1030,55**	1036,14	1044,37	688,91	1,34

Tabela 5.3: Resultados para os problemas-teste de [Montané and Galvão, 2006]

Problema	#C	#V	MelhorLit	Melhor	Média	Tempo (s)	Gap (%)
r101	100	12	1010,90	**1009,52**	1016,70	24,72	0,57
r201	100	3	**666,20**	**666,20**	667,72	37,23	0,23
c101	100	16	1220,26	**1220,18**	1224,91	22,93	0,38
c201	100	5	**662,07**	**662,07**	664,00	24,04	0,36
rc101	100	10	**1059,32**	**1059,32**	1065,97	21,26	0,63
rc201	100	3	**672,92**	**672,92**	680,00	32,60	1,06
r1_2_1	200	23	3371,29	**3357,64**	3433,83	196,71	1,86
r2_2_1	200	5	**1665,58**	**1665,58**	1687,39	143,94	1,31
c1_2_1	200	27	3640,20	**3636,74**	3658,05	188,17	0,49
c2_2_1	200	9	1728,14	**1726,59**	1751,59	116,87	1,36
rc1_2_1	200	23	3327,98	**3312,92**	3345,80	207,58	0,54
rc2_2_1	200	5	**1560,00**	**1560,00**	1579,64	135,44	1,26
r1_4_1	400	54	9695,77	**9627,43**	9730,69	1599,83	0,36
r2_4_1	400	10	**3574,86**	3582,08	3648,61	895,47	2,06
c1_4_1	400	63	11124,30	**11098,21**	11147,40	1437,12	0,21
c2_4_1	400	15	**3575,63**	3596,37	3664,53	782,31	2,49
rc1_4_1	400	52	9602,53	**9535,46**	9640,17	1947,84	0,39
rc2_4_1	400	11	**3416,61**	3422,11	3503,29	2005,14	2,54

De acordo com os resultados obtidos nas Tabelas 5.1, 5.2 e 5.3, verifica-se que o GENILS é capaz de gerar soluções médias de boa qualidade e com baixo desvio. Observa-

se que, nos problemas propostos por [Dethloff, 2001], o algoritmo proposto obteve soluções com desvio inferior a 0,97%, além de encontrar, em seis problemas-teste, o melhor valor da literatura em todas as 100 execuções. Para os problemas de [Salhi and Nagy, 1999] e [Montané and Galvão, 2006], o GENILS obteve soluções com *gap* máximo de 5,11% e 2,54%, respectivamente.

As Tabelas 5.4, 5.5, 5.6 e 5.7 comparam o desempenho do GENILS com as abordagens da literatura para os problemas-teste propostos por [Dethloff, 2001], [Salhi and Nagy, 1999] e [Montané and Galvão, 2006]. Nessas tabelas, **Melhor** é o melhor resultado obtido pelo método do respectivo autor e **Tempo** é o tempo, em segundos, de processamento do algoritmo. Calculou-se também o desvio percentual (coluna **Gap***) do GENILS em relação ao melhor resultado existente na literatura até então. Esse desvio é calculado pela equação dada em (5.2). Valores em negrito representam o melhor resultado existente.

$$Gap^* = 100 \times \frac{Melhor - MelhorLit}{MelhorLit} \tag{5.2}$$

Observa-se, nas Tabelas 5.4 a 5.7, que, em relação aos problemas-teste propostos por [Dethloff, 2001], o GENILS foi capaz de alcançar todas as melhores soluções da literatura. Dos 14 problemas-teste de [Salhi and Nagy, 1999], o algoritmo proposto alcançou três melhores soluções da literatura, enquanto que nos demais problemas, o *gap* máximo foi de 3,16%. Ressalta-se que nesse último conjunto não há uma supremacia de um algoritmo em relação aos demais. Os que mais se destacaram nesse conjunto foram os algoritmos de [Wassan *et al.*, 2007] e [Zachariadis *et al.*, 2009], com seis melhores resultados cada, seguido dos algoritmos de [Subramanian *et al.*, 2008] e GENILS, com três melhores resultados cada e, finalmente, o algoritmo de [Chen and Wu, 2006], com um único melhor resultado. O melhor desempenho do GENILS se deu nos problemas de [Montané and Galvão, 2006], em que, dos 18 problemas desse conjunto, em 9 foram geradas novas melhores soluções, em 6 foram encontrados os melhores resultados da literatura e nos 3 restantes, o *gap* foi inferior a 0,58%.

Comparando o GENILS com outros algoritmos, verifica-se que o mesmo tem desempenho bem próximo ao de [Subramanian *et al.*, 2008]. De fato, tanto nos problemas-teste de [Dethloff, 2001] quanto nos de [Montané and Galvão, 2006], são esses os únicos algoritmos que têm todos os melhores resultados da literatura. Nesse segundo conjunto de problemas-teste, o GENILS foi superior ao de [Subramanian *et al.*, 2008] em 9 e inferior em 3. Já no conjunto de [Salhi and Nagy, 1999], o GENILS superou o algoritmo de [Subramanian *et al.*, 2008] em 2 casos e teve desempenho pior em 5.

Tabela 5.4: Comparação entre o GENILS e as abordagens da literatura para os problemas-teste de [Dethloff, 2001]

Problema	Röpke e Pisinger		Zachariadis *et al.*		Subramanian *et al.*		GENILS		
-	**Melhor**	**Tempo[1] (s)**	**Melhor**	**Tempo[2] (s)**	**Melhor**	**Tempo[3] (s)**	**Melhor**	**Tempo[4] (s)**	***Gap**** (%)
SCA3-0	636,10	232,00	636,06	2,83	**635,62**	0,90	**635,62**	6,77	0,00
SCA3-1	**697,80**	170,00	**697,84**	2,12	**697,84**	1,12	**697,84**	8,49	0,00
SCA3-2	**659,30**	160,00	**659,34**	2,58	**659,34**	1,19	**659,34**	8,13	0,00
SCA3-3	680,60	182,00	**680,04**	3,13	**680,04**	1,13	**680,04**	8,45	0,00
SCA3-4	**690,50**	160,00	**690,50**	2,68	**690,50**	1,32	**690,50**	8,09	0,00
SCA3-5	**659,90**	178,00	**659,90**	2,56	**659,90**	1,17	**659,90**	8,19	0,00
SCA3-6	**651,10**	171,00	**651,09**	4,40	**651,09**	1,23	**651,09**	8,21	0,00
SCA3-7	666,10	162,00	**659,17**	2,98	**659,17**	1,69	**659,17**	6,76	0,00
SCA3-8	**719,50**	157,00	**719,47**	3,98	**719,47**	1,08	**719,47**	8,85	0,00
SCA3-9	**681,00**	167,00	**681,00**	3,86	**681,00**	1,03	**681,00**	8,63	0,00
SCA8-0	975,10	98,00	**961,50**	3,21	**961,50**	2,52	**961,50**	5,65	0,00
SCA8-1	1052,40	95,00	1050,20	3,55	**1049,65**	2,98	**1049,65**	5,67	0,00
SCA8-2	**1039,60**	83,00	**1039,64**	4,67	**1039,64**	3,42	**1039,64**	5,92	0,00
SCA8-3	991,10	94,00	**983,34**	3,29	**983,34**	3,44	**983,34**	4,58	0,00
SCA8-4	**1065,50**	84,00	**1065,49**	2,68	**1065,49**	2,74	**1065,49**	5,98	0,00
SCA8-5	**1027,10**	96,00	**1027,08**	4,50	**1027,08**	3,44	**1027,08**	6,62	0,00
SCA8-6	972,50	93,00	**971,82**	2,67	**971,82**	2,48	**971,82**	6,57	0,00
SCA8-7	1061,00	92,00	1052,17	4,32	**1051,28**	5,39	**1051,28**	5,56	0,00
SCA8-8	**1071,20**	85,00	**1071,18**	3,43	**1071,18**	2,05	**1071,18**	5,57	0,00
SCA8-9	**1060,50**	86,00	**1060,50**	4,12	**1060,50**	3,10	**1060,50**	5,62	0,00

[1] Tempo de CPU em um computador Pentium IV 1.5 GHz.
[2] Tempo de CPU em um computador Pentium IV 2.4 GHz.
[3] Tempo de CPU em um computador Intel Core 2 Quad 2.5 GHz.
[4] Tempo de CPU em um computador Intel Core 2 Duo 1.66 GHz.

Tabela 5.5: Comparação entre o GENILS e as abordagens da literatura para os problemas-teste de [Dethloff, 2001]

Problema	Röpke e Pisinger		Zachariadis *et al.*		Subramanian *et al.*		GENILS		
-	**Melhor**	**Tempo[1] (s)**	**Melhor**	**Tempo[2] (s)**	**Melhor**	**Tempo[3] (s)**	**Melhor**	**Tempo[4] (s)**	***Gap**** (%)
CON3-0	**616,50**	171,00	**616,52**	3,89	**616,52**	2,02	**616,52**	6,77	0,00
CON3-1	**554,50**	190,00	**554,47**	2,97	**554,47**	1,83	**554,47**	7,76	0,00
CON3-2	521,40	176,00	519,26	3,32	**518,00**	2,10	**518,00**	9,28	0,00
CON3-3	**591,20**	177,00	**591,19**	2,78	**591,19**	1,34	**591,19**	9,18	0,00
CON3-4	**588,80**	173,00	589,32	3,12	**588,79**	1,79	**588,79**	6,29	0,00
CON3-5	**563,70**	179,00	**563,70**	3,45	**563,70**	1,71	**563,70**	9,16	0,00
CON3-6	**499,10**	195,00	500,80	2,98	**499,05**	1,93	**499,05**	7,33	0,00
CON3-7	**576,50**	226,00	**576,48**	2,40	**576,48**	1,52	**576,48**	6,96	0,00
CON3-8	**523,10**	174,00	**523,05**	5,02	**523,05**	1,51	**523,05**	8,75	0,00
CON3-9	**578,20**	163,00	580,05	3,14	**578,24**	1,58	**578,24**	6,87	0,00
CON8-0	**857,20**	86,00	**857,17**	3,40	**857,17**	3,74	**857,17**	6,36	0,00
CON8-1	**740,90**	81,00	**740,85**	3,73	**740,85**	2,82	**740,85**	4,88	0,00
CON8-2	716,00	84,00	713,14	2,87	**712,89**	2,46	**712,89**	6,95	0,00
CON8-3	**811,10**	91,00	**811,07**	3,82	**811,07**	2,82	**811,07**	5,87	0,00
CON8-4	**772,30**	87,00	**772,25**	2,98	**772,25**	3,37	**772,25**	5,01	0,00
CON8-5	755,70	94,00	756,91	5,76	**754,88**	3,30	**754,88**	5,82	0,00
CON8-6	693,10	96,00	**678,92**	4,00	**678,92**	3,04	**678,92**	5,67	0,00
CON8-7	814,80	94,00	**811,96**	2,46	**811,96**	2,73	**811,96**	4,71	0,00
CON8-8	774,00	94,00	**767,53**	4,21	**767,53**	3,42	**767,53**	5,23	0,00
CON8-9	809,30	92,00	**809,00**	3,87	**809,00**	3,60	**809,00**	5,86	0,00

[1] Tempo de CPU em um computador Pentium IV 1.5 GHz.
[2] Tempo de CPU em um computador Pentium IV 2.4 GHz.
[3] Tempo de CPU em um computador Intel Core 2 Quad 2.5 GHz.
[4] Tempo de CPU em um computador Intel Core 2 Duo 1.66 GHz.

Tabela 5.6: Comparação entre o GENILS e as abordagens da literatura para os problemas-teste de [Salhi and Nagy, 1999]

Problema	Wassan *et al.*		Zachariadis *et al.*		Subramanian *et al.*		GENILS		
-	**Melhor**	**Tempo[1] (s)**	**Melhor**	**Tempo[2] (s)**	**Melhor**	**Tempo[3] (s)**	**Melhor**	**Tempo[4] (s)**	***Gap**** (%)
CMT1X	468,30	48	469,80	2,89	**466,77**	1,10	**466,77**	7,82	0,00
CMT1Y	**458,96**	69	469,80	3,85	466,77	1,08	466,77	7,61	1,68
CMT2X	**668,77**	94	684,21	7,42	684,21	6,99	684,21	17,62	2,31
CMT2Y	**663,25**	102	684,21	8,02	684,21	5,84	684,21	20,10	3,16
CMX3X	729,63	294	**721,27**	11,62	721,40	6,80	721,40	59,61	0,02
CMT3Y	745,46	285	**721,27**	13,53	721,40	7,37	**721,27**	58,72	0,00
CMT12X	**644,70**	242	662,22	11,80	662,22	8,02	662,22	22,89	2,72
CMT12Y	**659,52**	254	662,22	7,59	662,22	7,32	663,50	22,33	0,60
CMT11X	861,97	504	**838,66**	17,78	839,39	12,58	846,23	48,85	0,90
CMT11Y	**830,39**	325	837,08	14,26	841,88	14,80	836,04	287,30	0,68
CMT4X	876,50	558	**852,46**	27,75	**852,46**	50,72	**852,46**	134,26	0,00
CMT4Y	870,44	405	852,46[a]	31,20	852,46	46,06	862,28	266,76	1,17[b]
CMT5X	1044,51	483	**1030,55**	51,67	**1030,55**	53,51	1033,51	768,94	0,29
CMT5Y	1054,46	533	**1030,55**	58,81	1031,17	58,74	1036,14	398,75	0,54

[1] Tempo de CPU em um computador Sun-Fire-V440 com um processador UltraSPARC-IIIi 1062 MHz.
[2] Tempo de CPU em um computador Pentium IV 2.4 GHz.
[3] Tempo de CPU em um computador Intel Core 2 Quad 2.5 GHz.
[4] Tempo de CPU em um computador Intel Core 2 Duo 1.66 GHz.
[a] Um resultado melhor de valor 852,35 foi obtido por [Chen and Wu, 2006].
[b] *Gap* em relação ao valor encontrado por [Chen and Wu, 2006].

Tabela 5.7: Comparação entre o GENILS e as abordagens da literatura para os problemas-teste de [Montané and Galvão, 2006]

Problema	Montané e Galvão		Zachariadis *et al.*		Subramanian *et al.*		GENILS		
-	**Melhor**	**Tempo[1] (s)**	**Melhor**	**Tempo[2] (s)**	**Melhor**	**Tempo[3] (s)**	**Melhor**	**Tempo[4] (s)**	***Gap**** (%)
r101	1042,62	12,20	1019,48	10,50	1010,90	10,51	**1009,95**	35,65	-0,09
r201	671,03	12,02	**666,20**	8,70	**666,20**	6,24	**666,20**	39,62	0,00
c101	1259,79	12,07	1220,99	10,20	1220,26	12,73	**1220,18**	18,34	-0,01
c201	666,01	12,40	**662,07**	5,70	**662,07**	4,18	**662,07**	16,62	0,00
rc101	1094,15	12,30	**1059,32**	12,90	**1059,32**	9,48	**1059,32**	12,79	0,00
rc201	674,46	12,07	**672,92**	10,50	**672,92**	4,21	**672,92**	24,03	0,00
r1_2_1	3447,20	55,56	3393,31	61,80	3371,29	95,79	**3357,64**	175,81	-0,40
r2_2_1	1690,67	50,95	1673,65	47,40	**1665,58**	24,13	**1665,58**	103,44	0,00
c1_2_1	3792,62	52,21	3652,76	66,30	3640,20	95,17	**3636,74**	117,62	-0,10
c2_2_1	1767,58	65,79	1753,68	60,90	1728,14	41,94	**1726,59**	127,81	-0,09
rc1_2_1	3427,19	58,39	3341,25	45,30	3327,98	76,30	**3312,92**	299,30	-0,45
rc2_2_1	1645,94	52,93	1562,34	62,40	**1560,00**	34,28	**1560,00**	77,48	0,00
r1_4_1	10027,81	330,42	9758,77	315,30	9695,77	546,39	**9627,43**	2928,31	-0,71
r2_4_1	3685,26	324,44	3606,72	273,60	**3574,86**	231,73	3582,08	768,60	0,20
c1_4_1	11676,27	287,12	11207,37	283,50	11124,30	524,35	**11098,21**	1510,44	-0,23
c2_4_1	3732,00	330,20	3630,72	336,00	**3575,63**	293,18	3596,37	569,01	0,58
rc1_4_1	9883,31	286,66	9697,65	145,80	9602,53	550,90	**9535,46**	2244,18	-0,70
rc2_4_1	3603,53	328,16	3498,30	345,00	**3416,61**	291,15	3422,11	3306,84	0,16

[1] Tempo de CPU em um computador Athlon XP 2.0 GHz.
[2] Tempo de CPU em um computador Pentium IV 2.4 GHz.
[3] Tempo de CPU em um computador Intel Core 2 Quad 2.5 GHz.
[4] Tempo de CPU em um computador Intel Core 2 Duo 1.66 GHz.

Uma comparação em termos de tempos computacionais não foi feita porque os algoritmos que foram comparados utilizaram máquinas distintas.

Para verificar a convergência do algoritmo GENILS, foi realizada uma análise de probabilidade empírica. Para tanto, foram escolhidos três problemas-teste, um de cada conjunto, a saber, o `CON8-7` de [Dethloff, 2001], o `CMT1X` de [Salhi and Nagy, 1999] e o `rc101` de [Montané and Galvão, 2006]. O critério de parada do algoritmo foi alterado para finalizar quando fosse encontrado o valor alvo, definido como o melhor encontrado na literatura para o respectivo problema-teste.

Para gerar o gráfico de distribuição de probabilidade em relação ao tempo para encontrar o alvo, foi utilizado o método descrito em [Aiex *et al.*, 2002]. Esse método consiste, inicialmente, em ordenar os tempos $T = \{t_1, t_2, \ldots, t_m\}$ que o GENILS encontrou o valor alvo, sendo m o número de execuções. Em seguida, calcula-se a probabilidade acumulada $p_i = (i - 1/2)/m$ associada ao i-ésimo tempo de execução. Por fim, geram-se m pontos $z_i = (t_i, p_i)$, os quais são utilizados para construir o gráfico.

As Figuras 5.1, 5.2 e 5.3 mostram os gráficos de distribuição de probabilidade para os problemas-teste `CON8-7`, `CMT1X` e `rc101`, respectivamente, considerando $m = 100$ execuções.

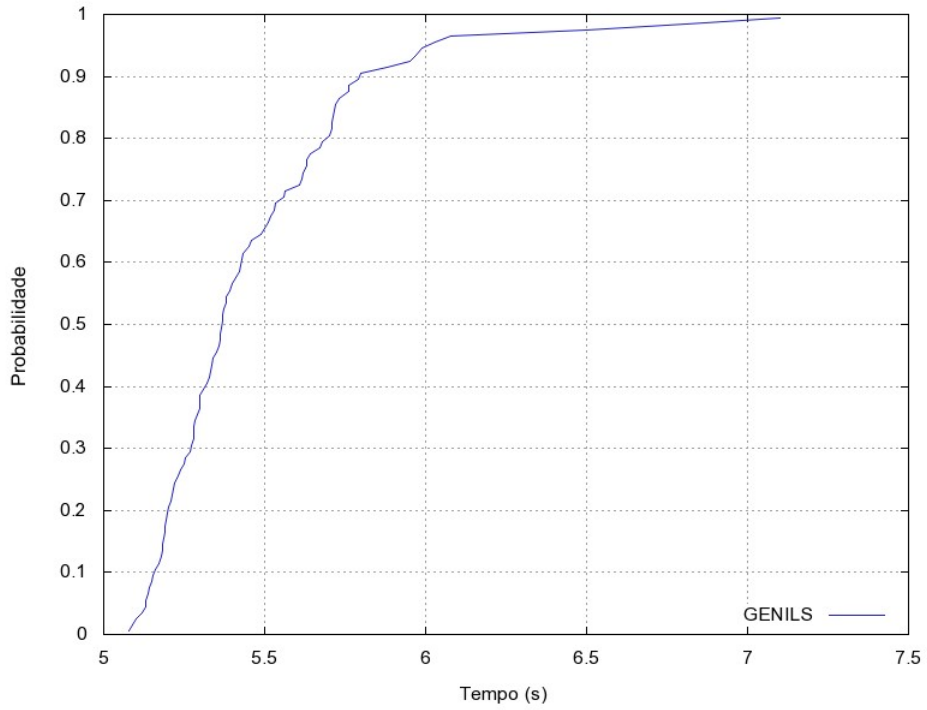

Figura 5.1: Probabilidade acumulada para o problema-teste `CON8-7`

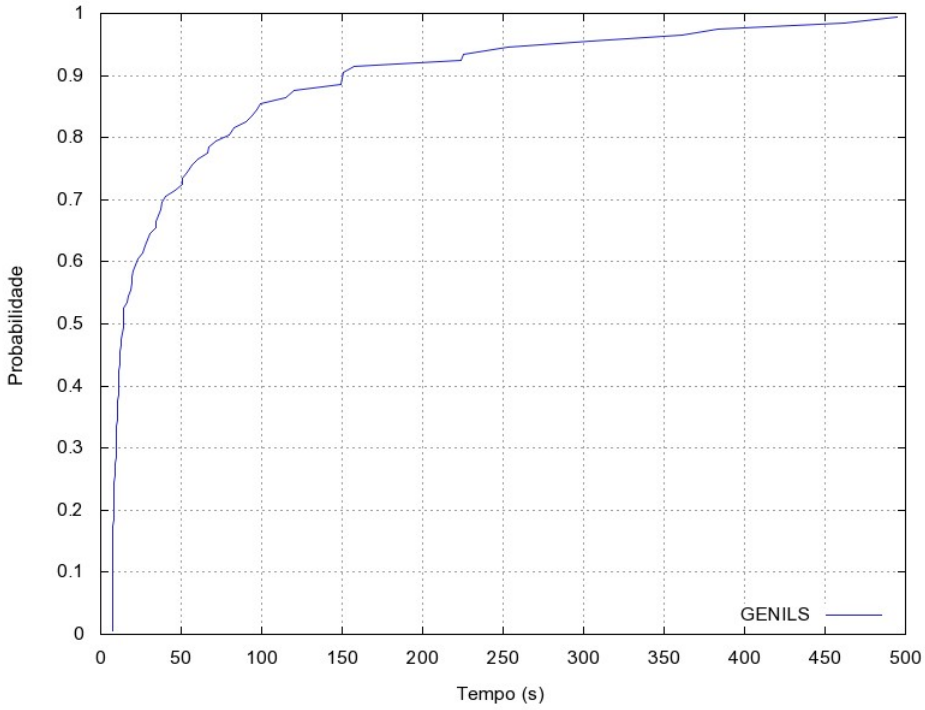

Figura 5.2: Probabilidade acumulada para o problema-teste `CMT1X`

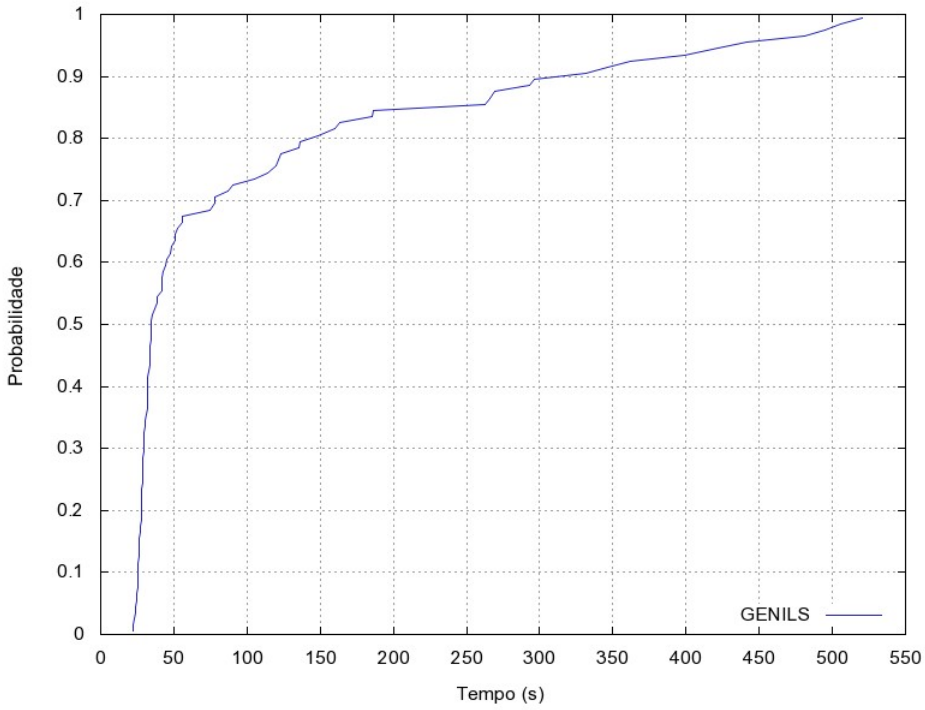

Figura 5.3: Probabilidade acumulada para o problema-teste `rc101`

De acordo com a Figura 5.1, observa-se que, em 6 segundos, o GENILS conseguiu obter

o valor alvo em aproximadamente 95% dos casos. Em relação ao gráfico apresentado na Figura 5.2, em 150 segundos, o algoritmo proposto foi capaz de encontrar o valor alvo em 90% das execuções e em pouco menos de 500 segundos, essa taxa de sucesso foi de quase 100%. Para o gráfico referente ao problema-teste `rc101`, apresentado na Figura 5.3, verifica-se que, em 70% e em quase 100% dos casos, o GENILS obteve o melhor valor da literatura em cerca de 75 e 520 segundos, respectivamente.

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho abordou o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Dada a sua dificuldade de resolução na otimalidade, em virtude de o mesmo ser da classe NP-difícil, foi proposto um algoritmo heurístico híbrido, denominado GENILS, descrito na Seção 4.5. Esse algoritmo é baseado na metaheurística *Iterated Local Search* e utiliza adaptações dos procedimentos GENIUS, Inserção Mais Barata e Descida em Vizinhança Variável (VND). Os dois primeiros procedimentos são utilizados para gerar uma solução inicial, enquanto o último é utilizado como método de busca local do GENILS. O VND explora a vizinhança de uma solução utilizando os movimentos *Shift*, *Shift*(2,0), *Swap*, *Swap*(2,1), *Swap*(2,2), *M2-Opt* e *kOr-Opt*, apresentados na Seção 4.2. Além disso, ele realiza uma intensificação da busca sempre que ocorre uma melhora na solução corrente. Essa intensificação é feita somente nas rotas modificadas e é realizada pelos procedimentos *G3-Opt* e *G4-Opt*, apresentados na Subseção 4.6.1, bem como pelo procedimento *Reverse*, apresentado na Subseção 4.6.2. As perturbações do GENILS consistem em aplicar de um a três movimentos *Shift* ou *Swap* ou o movimento *Ejection Chain*, descritos na Subseção 4.5.1. A escolha do mecanismo de perturbação é feita de forma aleatória a cada iteração do GENILS.

Para validar o algoritmo proposto, foram utilizados três conjuntos de problemas-teste consagrados na literatura. O primeiro, de [Dethloff, 2001], envolve 40 problemas com 50 clientes; o segundo, de [Salhi and Nagy, 1999], contém 14 problemas de 50 a 199 clientes e o terceiro, de [Montané and Galvão, 2006], possui 18 problemas com 100, 200 e 400 clientes.

De acordo com os resultados obtidos, verifica-se que o GENILS é competitivo com as melhores abordagens da literatura, sendo capaz de produzir soluções de qualidade. De

fato, em um conjunto de problemas-teste, foram alcançados todos os melhores resultados da literatura; em outro, foram gerados nove melhores resultados e seis resultados iguais aos melhores da literatura; e no terceiro, foram igualados três resultados da literatura, tendo-se um *gap* máximo igual a 3,16% para os demais problemas desse conjunto. Na média, o GENILS obteve soluções com desvio inferior a 5,11%, considerando todos os problemas-teste, sendo que a variabilidade das soluções finais foi inferior a 1% em 71% dos casos. Também analisou-se a distribuição de probabilidade empírica de se alcançar um dado valor alvo em função do tempo em três diferentes problemas-teste, um relativo a cada conjunto de problemas da literatura. Os resultados mostraram que nos instantes iniciais, a probabilidade de alcançar o alvo é grande e eleva-se quanto mais tempo é concedido ao GENILS.

Como trabalho futuro, pretende-se aprimorar os procedimentos *G3-Opt* e *G4-Opt*, descritos na Subseção 4.6.1, de forma a considerar a recombinação de múltiplas rotas. Além disso, é estratégico combinar o algoritmo GENILS com a metaheurística Busca Tabu, sendo esta acionada em substituição ao VND, por exemplo, após um certo número de iterações do GENILS. Isso se deve ao fato de que a Busca Tabu é o algoritmo base de [Wassan *et al.*, 2007] e [Zachariadis *et al.*, 2009], os quais têm a maioria dos melhores resultados dos problemas-teste de [Salhi and Nagy, 1999].

Referências

[Aiex *et al.*, 2002] Aiex, R. M., Resende, M. G. C., and Ribeiro, C. C. (2002). Probability distribution of solution time in grasp: An experimental investigation. *Journal of Heuristics*, 8(3):343–373.

[Anderberg, 1973] Anderberg, M. R. (1973). *Cluster analysis for applications*. Monographs and Textbooks on Probability and Mathematical Statistics. Academic Press, Inc., New York.

[Angelelli and Mansini, 2002] Angelelli, E. and Mansini, R. (2002). *Quantitative Approaches to Distribution Logistics and Supply Chain Management*, chapter A branch-and-price algorithm for a simultaneous pick-up and delivery problem, pages 249–267. Springer, Berlin-Heidelberg.

[Bean, 1994] Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160.

[Bianchessi and Righini, 2007] Bianchessi, N. and Righini, G. (2007). Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers and Operations Research*, 34(2):578–594.

[Chen, 2006] Chen, J. F. (2006). Approaches for the vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Chinese Institute of Industrial Engineers*, 23(2):141–150.

[Chen and Wu, 2006] Chen, J. F. and Wu, T. H. (2006). Vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Operational Research Society*, 57(5):579–587.

[Crispim and Brandão, 2005] Crispim, J. and Brandão, J. (2005). Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. *Journal of the Operational Research Society*, 56(7):1296–1302.

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

[Dell'Amico *et al.*, 2006] Dell'Amico, M., Righini, G., and Salanim, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2):235–247.

[Dethloff, 2001] Dethloff, J. (2001). Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. *OR Spektrum*, 23:79–96.

[Dorigo *et al.*, 1996] Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:29–41.

[Dueck, 1993] Dueck, G. (1993). New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104:86–92.

[Gendreau *et al.*, 1992] Gendreau, M., Hertz, A., and Laporte, G. (1992). New insertion and post optimization procedures or the traveling salesman problem. *Operations Research*, 40:1086–1094.

[Gökçe, 2004] Gökçe, E. I. (2004). A revised ant colony system approach to vehicle routing problems. Master's thesis, Graduate School of Engineering and Natural Sciences, Sabanci University.

[Glover and Laguna, 1997] Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publisher, Boston.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional.

[Halse, 1992] Halse, K. (1992). *Modeling and solving complex vehicle routing problems*. PhD thesis, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, Denmark.

[Hansen and Mladenović, 2001] Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.

[Kirkpatrick *et al.*, 1983] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 4598(13):671–680.

[Lourenço *et al.*, 2003] Lourenço, H. R., Martin, O., and Stützle, T. (2003). Iterated local search. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research and Management Science*, pages 321–353. Kluwer Academic Publishers, Norwell, MA.

[Min, 1989] Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research A*, 23(5):377–386.

[Mladenović and Hansen, 1997] Mladenović, N. and Hansen, P. (1997). Variable neighboorhood search. *Computers and Operations Research*, 24:1097–1100.

[Montané and Galvão, 2006] Montané, F. A. T. and Galvão, R. D. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Comput. Oper. Res.*, 33(3):595–619.

[Nagy and Salhi, 2005] Nagy, G. and Salhi, S. (2005). Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162:126–141.

[Or, 1976] Or, I. (1976). *Traveling salesman-type combinational problems and their relation to the logistics of blood banking*. PhD thesis, Northwestern University, USA.

[Parragh *et al.*, 2008a] Parragh, S., Doerner, K., and Hartl, R. (2008a). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51.

[Parragh *et al.*, 2008b] Parragh, S., Doerner, K., and Hartl, R. (2008b). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(2):81–117.

[Rego and Roucairol, 1996] Rego, C. and Roucairol, C. (1996). *Meta-Heuristics Theory and Applications*, chapter A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem, pages 661–675. Kluwer Academic Publisher, Boston.

[Röpke and Pisinger, 2006] Röpke, S. and Pisinger, D. (2006). A unified heuristic for a large class of vehicle routing problems with backhauls. Technical Report 2004/14, University of Copenhagen.

[Salhi and Nagy, 1999] Salhi, S. and Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 50:1034–1042.

[Shaw, 1998] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *CP '98: Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming*, pages 417–431, London, UK. Springer-Verlag.

[Steiglitz and Weiner, 1968] Steiglitz, K. and Weiner, P. (1968). Some improved algorithms for computer solution of the traveling salesman problem. In *Proceedings of the Sixth Allerton Conference on Circuit Theory*, pages 814–821.

[Stützle and Hoos, 1999] Stützle, T. and Hoos, H. H. (1999). Analyzing the run-time behaviour of iterated local search for the tsp. In *Proceedings of the Third Metaheuristics International Conference*, pages 449–453, Angra dos Reis, Rio de Janeiro.

[Subramanian, 2008] Subramanian, A. (2008). Metaheurística Iterated Local Search aplicada ao problema de roteamento de veículos com coleta e entrega simultânea. Master's thesis, Universidade Federal da Paraíba, João Pessoa.

[Subramanian *et al.*, 2008] Subramanian, A., Cabral, L. A. F., and Ochi, L. S. (2008). An efficient ils heuristic for the vehicle routing problem with simultaneous pickup and delivery. Technical Report 07/2008, Universidade Federal Fluminense. http://www.ic.uff.br/∼satoru/index.php?id=2.

[Topcuoglu and Sevilmis, 2002] Topcuoglu, H. and Sevilmis, C. (2002). Task scheduling with conflicting objectives. In Yakhno, T. M., editor, *ADVIS*, volume 2457 of *Lecture Notes in Computer Science*, pages 346–355. Springer.

[Voudouris and Tsang, 1996] Voudouris, C. and Tsang, E. (1996). Partial constraint satisfaction problems and guided local search. In *In The Second International Conference on the Practical Application of Constraint Technology (PACT'96)*, pages 337–356.

[Vural, 2003] Vural, A. V. (2003). A GA based meta-heuristic for capacited vehicle routing problem with simultaneous pick-up and deliveries. Master's thesis, Graduate School of Engineering and Natural Sciences, Sabanci University.

[Wassan *et al.*, 2007] Wassan, N. A., Wassan, A. H., and Nagy, G. (2007). A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimization*, 15(4):368–386.

[Zachariadis *et al.*, 2009] Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T. (2009). A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications*, 36(2):1070–1081.