Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas
Departamento de Computação

raphael carlos cruz

Roteamento de veículos com coleta e entrega simultânea: uma abordagem heurística - Parte II

OURO PRETO - MG

2012

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas
Departamento de Computação

RAPHAEL CARLOS CRUZ

Roteamento de veículos com coleta e entrega simultânea: uma abordagem heurística - Parte II

Relatório Final, referente ao período agosto de 2011 a julho de 2012, apresentado à Universidade Federal de Ouro Preto (UFOP), como parte das exigências do Programa Institucional de Bolsas de Iniciação Científica do CNPq – PIBIC/CNPq.

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

OURO PRETO - MG

2012

Roteamento de veículos com coleta e entrega simultânea: uma abordagem heurística - Parte II

Raphael Carlos Cruz

Relatório Final, referente ao período agosto de 2011 a julho de 2012, apresentado à Universidade Federal de Ouro Preto (UFOP), como parte das exigências do Programa Institucional de Bolsas de Iniciação Científica do CNPq – PIBIC/CNPq.

Raphael Carlos Cruz
(Bolsista)

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

Ouro Preto, 07 de agosto de 2012.

Resumo

Este trabalho tem seu foco no Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Dada sua dificuldade de solução na otimalidade, é proposto um algoritmo heurístico, chamado de GENILS-TS-CL-PR, que combina os procedimentos heurísticos Inserção Mais Barata Rota a Rota, Inserção Mais Barata com Múltiplas Rotas, GENIUS, *Iterated Local Search* (ILS), Descida em Vizinhança Variável (VND), Busca Tabu (TS) e Reconexão de Caminhos (PR). Os três primeiros procedimentos visam à obtenção de uma boa solução inicial, enquanto os procedimentos VND e TS são usados como métodos de busca local para o ILS. A Busca Tabu somente é acionada após certo número de iterações sem sucesso do VND. O procedimento PR é acionado a cada iteração do ILS e conecta um ótimo local a uma solução elite gerada durante a busca. Uma Lista de Candidatos também é usada para reduzir o número de soluções avaliadas no espaço de soluções. Finalmente, o algoritmo foi paralelizado de forma a explorar o multiprocessamento existente nas máquinas atuais. O algoritmo proposto foi testado em problemas-teste da literatura e se mostrou capaz de gerar soluções de alta qualidade e baixa variabilidade.

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

Abstract

This work addresses the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD). Due to its complexity, we propose a heuristic algorithm for solving it, so-called GENILS-TS-CL-PR. This algorithm combines the heuristic procedures Cheapest Insertion, Cheapest Insertion with multiple routes, GENIUS, Iterated Local Search (ILS), Variable Neighborhood Descent (VND), Tabu Search (TS) and Path Relinking (PR). The first three procedures aim to obtain an good initial solution, and the VND and TS are used as local search methods for ILS. TS is called after some iterations without any improvement through of the VND. The PR procedure is called after ILS iteration and it connects a local optimum with an elite solution generated during the search. The algorithm also uses an strategy based on Candidate List to reduce the number of solutions evaluated in the solution space. Finally, the algorithm was parallelized in order to explore the multiprocessing capability of the current computers. The algorithm was tested on benchmark instances taken from the literature and it was able to generate high quality solutions.

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

Siglas e Abreviações

AG	:	Algoritmos Genéticos
BT	:	Busca Tabu
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 por [Mine *et al.*, 2010]
GENILS-TS-CL-PR	:	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
TS	:	*Tabu Search*
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

O objetivo geral deste projeto de pesquisa é desenvolver um algoritmo eficiente de otimização, baseado em metaheurísticas, para resolver o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES).

1.2.1 Objetivos específicos

São os seguintes os objetivos específicos:

1. Fazer uma revisão de literatura sobre os métodos utilizados para resolver o PRVCES;
2. Fazer uma revisão de literatura sobre técnicas heurísticas, em especial as metaheurísticas;
3. Aperfeiçoar o algoritmo GENILS, desenvolvido por [Mine, 2009] e [Mine *et al.*, 2010], e o GENILS-TS, desenvolvido na primeira etapa do projeto e submetido a publicação em 2010;

4. Paralelizar o algoritmo resultante do item anterior ou parte dele;

5. Testar o(s) algoritmo(s) desenvolvido(s);

6. Contribuir com a divulgação de técnicas de otimização aplicadas à resolução do problema, possibilitando aos desenvolvedores de aplicativos comerciais e/ou empresas de transporte melhorar a produtividade em logística;

7. Contribuir com a formação de recursos humanos especializados nessa área do conhecimento;

8. Contribuir para a consolidação do grupo de pesquisa **Logística e Pesquisa Operacional**.

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-TS-CL-PR, para resolver o PRVCES. Para detalhar esse algoritmo, o qual é baseado nas metaheurísticas *Iterated Local Search* e Busca Tabu, 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, o procedimento VND usado como busca local do GENILS, bem como a inserção da Busca Tabu.

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.

No Capítulo 7 são apresentadas todas as publicações de artigos em eventos científicos nacionais e internacionais conquistadas com o projeto desenvolvido.

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.

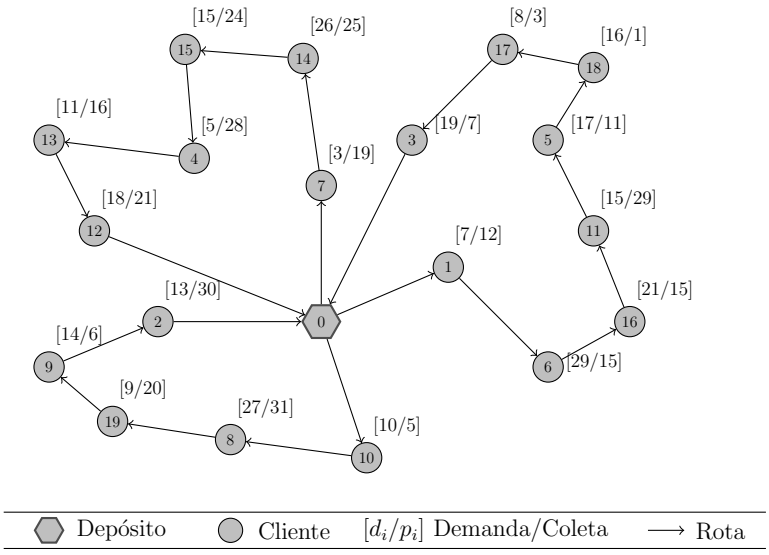

Figura 2.1: Exemplo do PRVCES.

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 7, 14, 15, 4, 13 e 12, retornando ao depósito no final. No primeiro cliente atendido nessa rota, é feita uma entrega de 3 unidades do produto e recolhida outras 19 unidades. A última visita do veículo ocorre no cliente 12, o qual demanda 18 unidades do produto e necessita que sejam coletadas 21 unidades.

Capítulo 3

Revisão Bibliográfica

3.1 Introdução

O Problema de Roteamento de Veículos (PRV), do inglês *Vehicle Routing Problem* (VRP), foi proposto por [Dantzig and Ramser, 1959], pode ser definido como segue. Dado um conjunto N de clientes, cada qual associado a uma demanda d_i, e uma frota homogênea de veículos de capacidade Q, o objetivo é obter um conjunto de rotas a serem percorridas pelos veículos cujo custo seja mínimo, levando em consideração o atendimento completo da demanda dos clientes.

Devido a inúmeras diversidades nos problemas reais, muitas variações do PRV têm sendo criadas com o intuito de atender uma gama maior de problemas. Uma importante variação do PRV, objeto de estudo deste trabalho, é o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Proposto por [Min, 1989], este problema se diferencia do PRV clássico por ter associado aos clientes não só uma demanda d_i, mas também uma quantidade p_i de produtos a serem totalmente coletados, sendo que ambas as operações devem ser realizadas simultâneamente.

Para resolver o PRVCES, cuja aplicação era voltada para o planejamento de distribuição de materiais para bibliotecas públicas, um método de três fases foi proposto por [Min, 1989]. Inicialmente o método agrupa os clientes em *clusters* por meio do Método de Ligação por Médias (*Avarage Linkage Method*) [Anderberg, 2007]. Na fase seguinte os veículos são associados às suas respectivas rotas e por último, o método consiste em resolver cada *cluster* por meio de uma heurística apropriada ao Problema do Caixeiro Viajante. Essa heurística atribui, a cada iteração, uma penalidade aos arcos nos quais a

carga do veículo foi excedida.

[Halse, 1992] tratou o PRVCES por meio de uma heurística que consiste em, inicialmente, associar os clientes aos veículos e, em seguida, criar rotas através de um método baseado na busca local *3-opt*.

Uma adaptação do método de Inserção Mais Barata foi criado por [Dethloff, 2001], em que os clientes são adicionados às rotas seguindo três padrões: *(i)* distância, *(ii)* capacidade residual e *(iii)* distância do cliente ao depósito. O método de solução proposto consiste apenas em uma heurística de construção, sem a aplicação de nenhum método de refinamento.

[Vural, 2003] apresentou duas versões de um Algoritmo Genético [Goldberg, 1989]. A primeira faz a codificação dos indivíduos através de chaves aleatórias [Bean, 1994] e a segunda foi implementada como uma heurística de refinamento baseada na estrutura do Algoritmo Genético desenvolvida por [Topcuoglu and Sevilmis, 2002].

O PRVCES é resolvido por [Gökçe, 2004] por meio da metaheurística Colônia de Formigas [Dorigo *et al.*, 1996], tendo a heurística *2-opt* como método de refinamento das soluções.

[Nagy and Salhi, 2005] resolveram um PRVCES com restrição do limite de tempo para percorrer cada rota. O algoritmo implementado utiliza diferentes heurísticas, tais como as buscas locais *2-opt*, *3-opt*, e as relacionadas em realocação e troca, assim como métodos para viabilizar a solução.

O PRVCES é resolvido em [Crispim and Brandão, 2005] por uma técnica híbrida que combina as metaheurísticas Busca Tabu [Glover and Laguna, 1997] e *Variable Neighborhood Descent* - VND [Hansen and Mladenović, 2001]. Os autores utilizaram o método de varredura (*sweep method*) para gerar a solução inicial e exploraram o espaço de soluções por meio de movimentos de troca e realocação.

[Röpke and Pisinger, 2006] estruturaram uma heurística baseada na *Large Neighborhood Search* - LNS [Shaw, 1998] para resolver o PRVCES. O LNS é uma busca local que consiste em duas maneiras para definir e pesquisar as estruturas de vizinhaça de grande complexidade. Inicialmente fixa-se uma parte da solução, facilitando a busca nessa porção do espaço de soluções. Em seguida, prossegue-se com a busca por meio de programação por restrições, programação inteira, técnicas *branch-and-cut* e outras.

A metaheurística Busca Tabu foi adaptada por [Montané and Galvão, 2006] para resolver o PRVCES. São usadas as estratégias *First Improvement* e *Best Improvement* e

movimentos de realocação, troca e *crossover*.

[Chen, 2006] abordou o problema através de uma técnica baseada nas metaheurísticas *Simulated Annealing* - SA [Kirkpatrick *et al.*, 1983] e Busca Tabu, ao passo que [Chen and Wu, 2006] criaram uma estrutura originada da heurística *record-to-record travel* [Dueck, 1993], tida como variação do SA.

Para a solução do PRVCES, [Bianchessi and Righini, 2007] usaram heurísticas de refinamento, algoritmos construtivos e técnicas fundamentadas nas metaheurísticas e movimentos de troca de nós (*node-exchange-based*) e troca de arcos (*arc-exchange-based*) na exploração do espaço de soluções.

Um método reativo da metaheurística Busca Tabu proposto por [Wassan *et al.*, 2007] foi utilizado para resolver o PRVCES. O método de varredura (*sweep method*) foi usado para gerar uma solução inicial. Movimentos de realocação, troca e inversão do sentido da rota são usados para explorar o espaço de soluções.

[Subramanian *et al.*, 2008] criou um algoritmo baseado no *Iterated Local Search* (ILS), tendo o procedimento *Variable Neighborhood Descent* (VND) como busca local. Uma solução inicial é gerada por uma adaptação da heurística de inserção que fora proposta por [Dethloff, 2001], contudo sem considerar a capacidade residual do veículo. O VND percorre o espaço de soluções usando seis movimentos fundamentados em troca, realocação e *crossover* entre clientes. Os mecanismos de perturbação usados no ILS foram o *ejection chain*, o *double swap* e o *double bridge*. O *ejection chain* transfere um cliente de cada rota a outra adjacente. O *double swap* faz duas trocas sucessivas e o *double bridge* remove quatro arcos e insere quatro novos arcos de maneira a criar uma nova rota.

[Zachariadis *et al.*, 2009] utilizaram uma técnica híbrida para resolver o PRVCES, unindo as metaheurísitcas Busca Tabu e *Guided Local Search* [Voudouris and Tsang, 1996]. Posteriormente, os mesmos autores propuseram um algoritmo evolucionário em uma publicação de [Zachariadis *et al.*, 2010], que utiliza uma memória adaptativa para guardar as informações das soluções de alta qualidade obtidas durante a busca. Essas informações são usadas para gerar novas soluções em regiões que possuem grande chance de trazer melhores resultados, sendo posteriormente, melhoradas pela Busca Tabu.

[Mine *et al.*, 2010] propuseram um algoritmo, nomeado GENILS, com o intuito de resolver o PRVCES. Este algoritmo utiliza a melhor solução gerada pelos métodos de Inseção Mais Barata, Inserção Mais Barata com Múltiplas Rotas e uma adaptação da heurística GENIUS [Gendreau *et al.*, 1992], como solução inicial. Como refinamento é

utilizado o ILS, que realiza a busca local por VND, utilizando sete estruturas de vizinhança. O GENILS foi testado em três conjuntos de problemas-teste consagrados da literatura e comparado com os algoritmos de [Wassan *et al.*, 2007], [Zachariadis *et al.*, 2010] e [Subramanian *et al.*, 2008], até então os melhores algoritmos para o problema. Dos 72 problemas-teste, o GENILS obteve os melhores resultados da literatura em 58 casos, enquanto o algoritmo de [Zachariadis *et al.*, 2010] obteve em 54 casos e o algoritmo de [Subramanian *et al.*, 2008] em 51.

[Subramanian *et al.*, 2010] apresentaram um algoritmo paralelo para resolver o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea. O algoritmo utiliza uma heurística *multi-start*, onde a cada iteração uma solução inicial é gerada através do procedimento Inserção mais Barata com Múltiplas Rotas. Essa solução é refinada através do ILS, que utiliza como metódo de busca local o procedimento *Random Neighborhood Ordering* (RVND). O RVND explora o espaço de soluções por meio de seis movimentos e a ordem das vizinhanças é aleatória a cada chamada. Os experimentos foram realizados em dois *clusters*, sendo um com uma arquitetura composta por 128 e outro com 256 núcleos. Os resultados obtidos com instâncias consagradas na literatura provaram que este algoritmo melhorou várias soluções conhecidas.

Dada a competitividade do algoritmo GENILS-TS frente aos demais algoritmos da literatura, propõe-se aperfeiçoá-lo com a inserção de uma Lista de Candidatos para evitar a avaliação de soluções não promissoras, implementar a Reconexão por Caminhos aplicada a cada ótimo local gerado pelo ILS e paralelizar as estruturas de vizinhança. Além disso, o algoritmo proposto, nomeado GENILS-TS-CL-PR, baseado na Parte I da pesquisa anterior, utiliza recursos computacionais mais acessíveis do que aquele de [Subramanian *et al.*, 2010], o que permite seu uso em empresas de menor porte.

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';

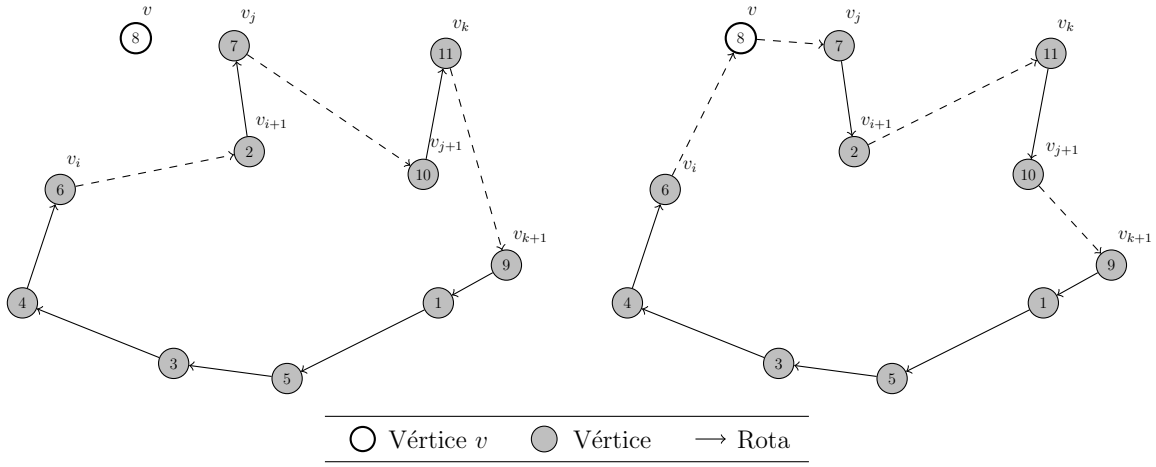

Figura 3.1: Exemplo da Inserção Tipo I da fase GENI.

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';

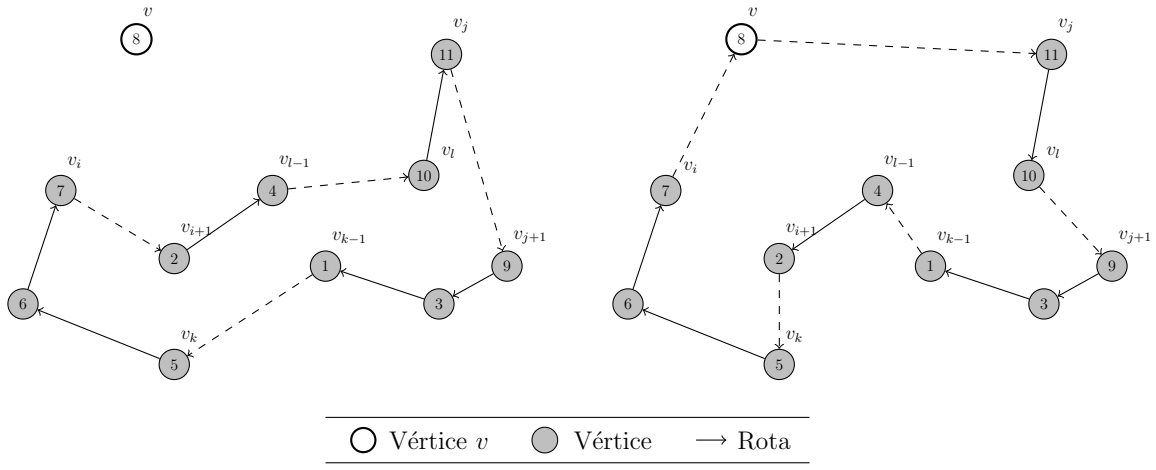

Figura 3.2: Exemplo da Inserção Tipo II da fase GENI.

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.

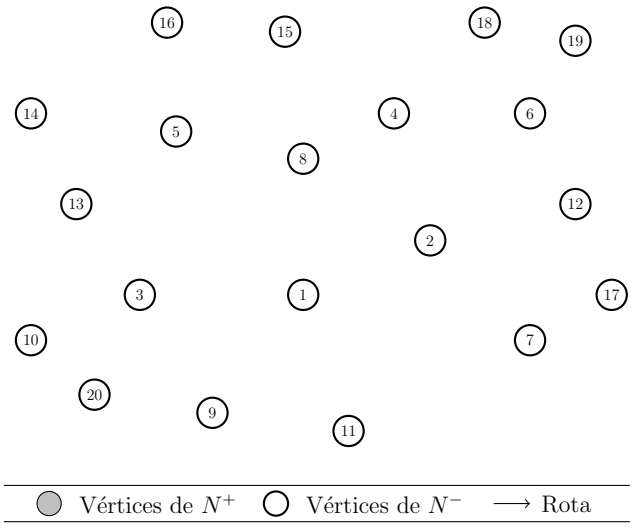

Figura 3.3: Exemplo de um problema do PCV, considerando 20 clientes.

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

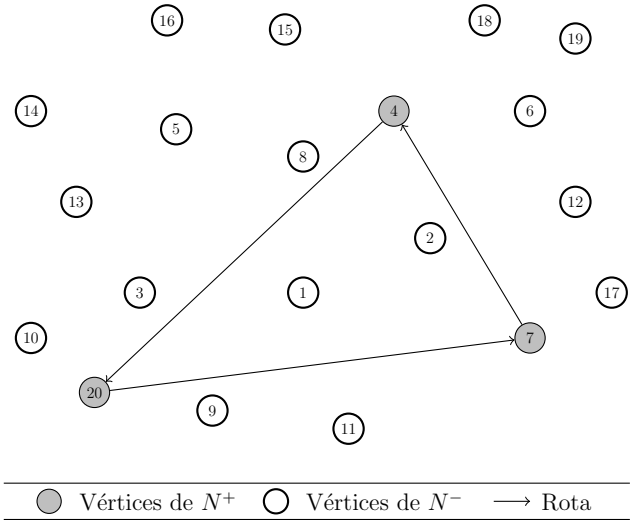

Figura 3.4: Subrota inicial do GENI contendo os vértices 7, 4 e 20.

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.

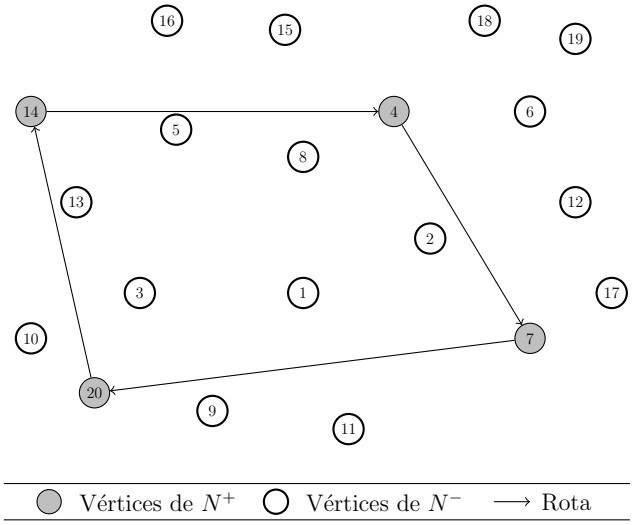

Figura 3.5: Inserção do vértice 14 com a heurística GENI.

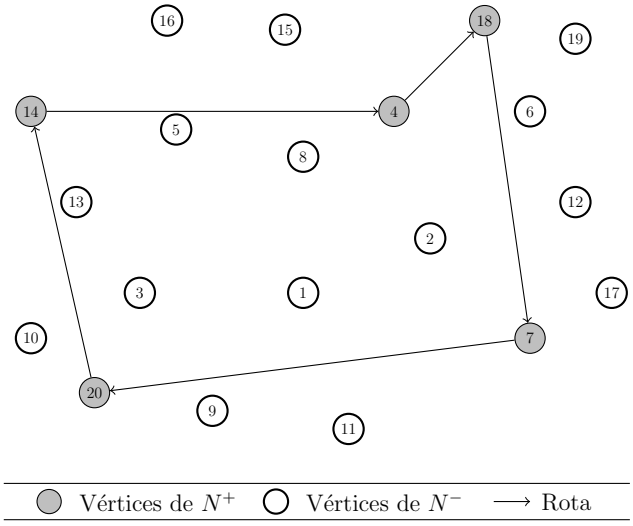

Figura 3.6: Inserção do vértice 18 com a heurística GENI.

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.

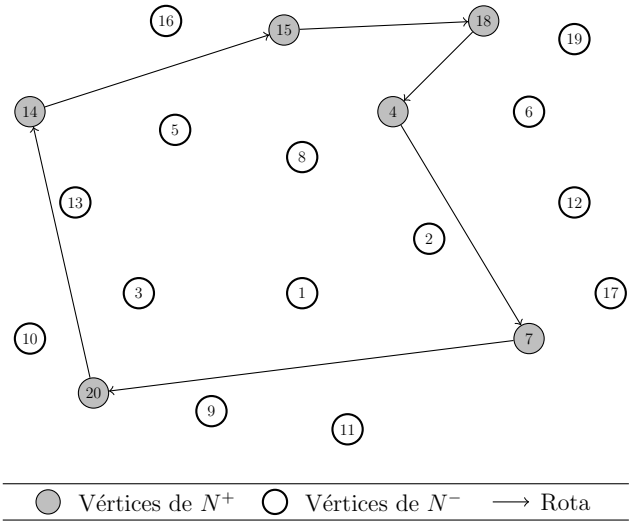

Figura 3.7: Inserção do vértice 15 com a heurística GENI.

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

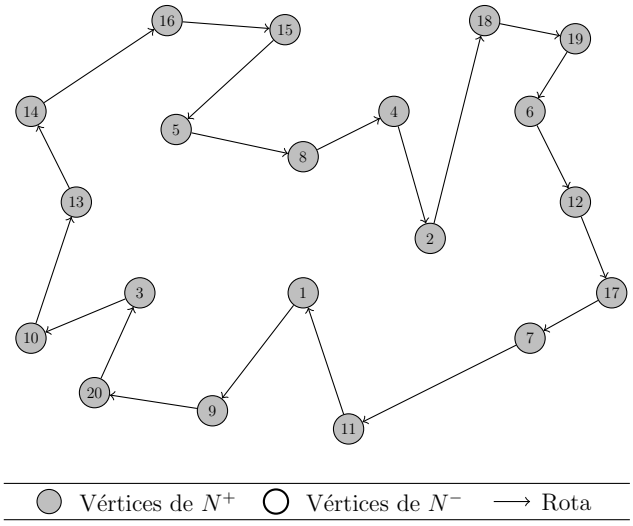

Figura 3.8: Solução inicial gerada pela heurística GENI.

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';

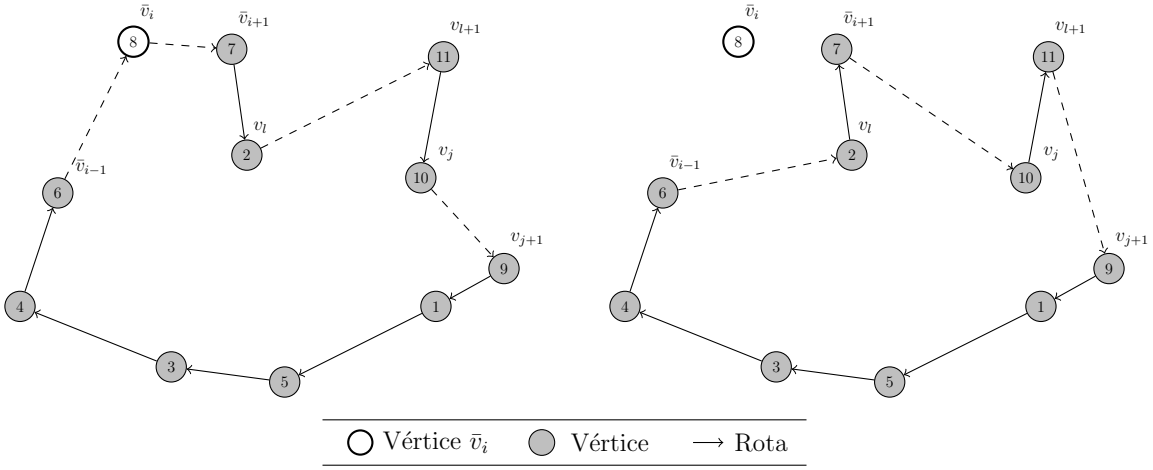

Figura 3.9: Exemplo da Remoção Tipo I da fase US.

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';

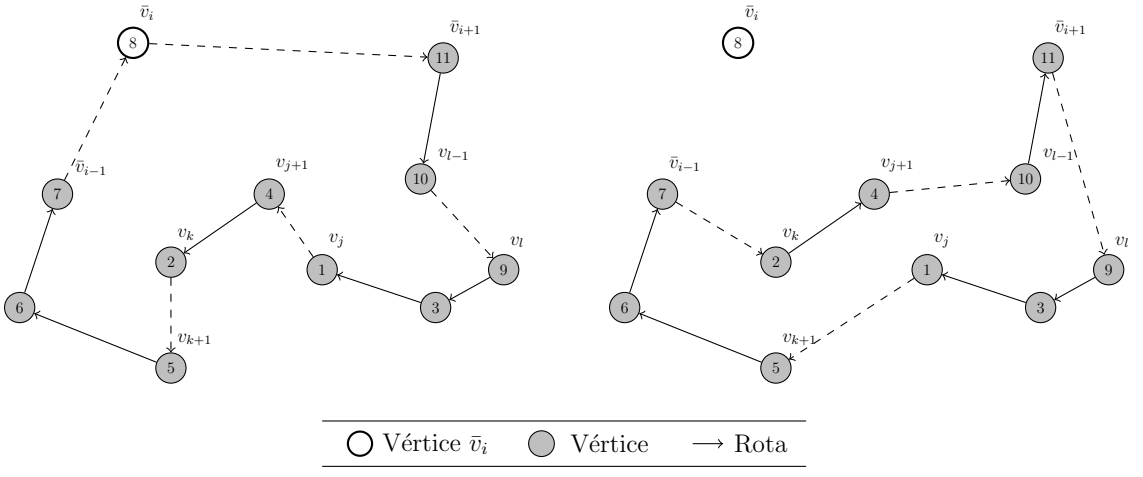

Figura 3.10: Exemplo da Remoção Tipo II da fase US.

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.

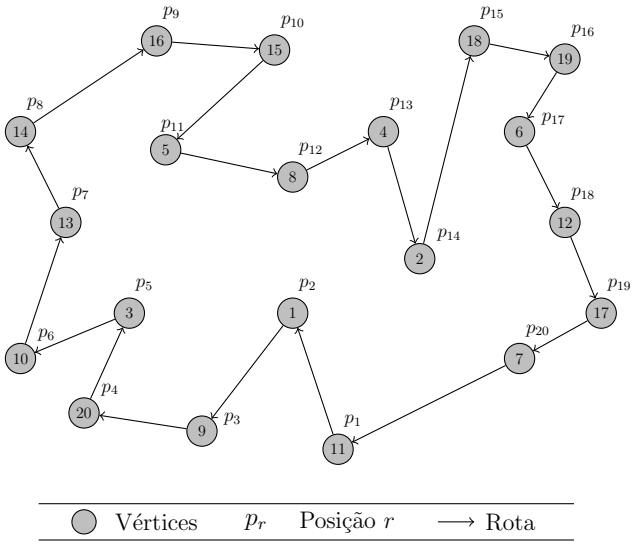

Figura 3.11: Exemplo da fase US.

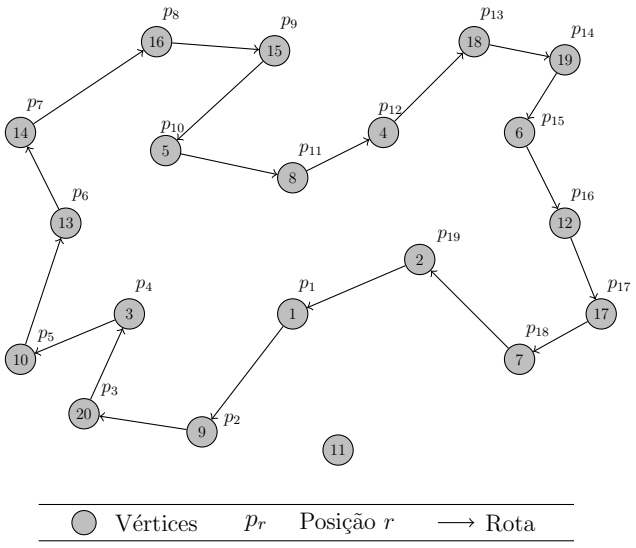

Figura 3.12: Remoção do vértice 11 localizado na posição p_1.

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*).

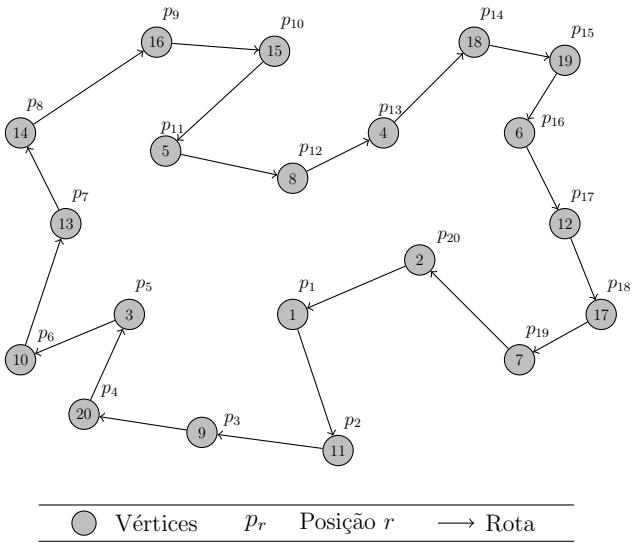

Figura 3.13: Reinserção do vértice 11 com o Algoritmo 5.

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.

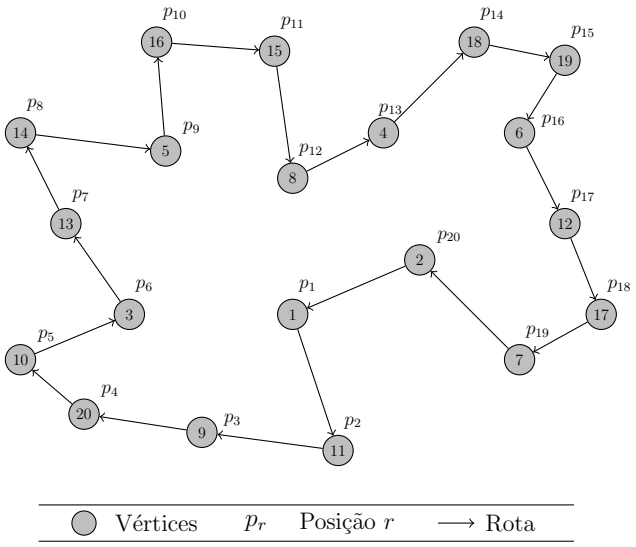

Figura 3.14: Solução gerada pela fase US e pela heurística GENIUS.

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), *Iterated Local Search* (ILS) e Busca Tabu, as quais são descritas nas Seções 3.4.1, 3.4.2 e 4.7, 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*

Seja s_0 uma solução inicial;
$s \leftarrow BuscaLocal(s_0)$;
enquanto (*critério de parada não satisfeito*) **faça**
  $s' \leftarrow Perturbação(s)$;
  $s'' \leftarrow BuscaLocal(s')$;
  $s \leftarrow CritérioAceitação(s, s'')$;
fim enquanto
Retorne s;

3.4.3 Busca Tabu

A Busca Tabu (BT), conhecida na literatura inglesa como *Tabu Search* – TS, é uma metaheurística de busca local originada nos trabalhos independentes de [Glover, 1997] e [Hansen, 1986].

Algoritmo 12 Busca Tabu

Seja s_0 solução inicial;
$s^* \leftarrow s$; {Melhor solução obtida até então}
$Iter \leftarrow 0$; {Contador do número de iterações}
$MelhorIter \leftarrow 0$; {Iteração mais recente que forneceu s^*}
Seja $BTmax$ o número máximo de iterações sem melhora em s^*;
$T \leftarrow \oslash$; {Lista Tabu}
Inicialize a função de aspiração A;
enquanto ($Iter - MelhorIter \leq BTmax$) **faça**
  $Iter \leftarrow Iter + 1$;
  Seja $s' \leftarrow s \oplus m$ melhor elemento de $V \subseteq N(s)$ tal que o movimento m não seja tabu ($m \notin T$) **ou** s' atenda a condição de aspiração ($f(s') < A(f(s))$);
  Atualize a Lista Tabu T;
  $s \leftarrow s'$;
  se ($f(s) < f(s^*)$) **então**
    $s^* \leftarrow s$;
    $MelhorIter \leftarrow Iter$;
  fim se
  Atualize a função de aspiração A;
fim enquanto
Retorne s^*;

A BT surgiu como uma técnica para guiar uma heurística de busca local tradicional na exploração do espaço de soluções além da otimalidade local. Ela utiliza uma estrutura de memória para explorar o espaço de soluções. Na BT caminha-se no espaço de soluções movendo-se de uma solução para outra que seja seu melhor vizinho, mesmo que este vizinho seja um movimento de piora. Como estratégia para não retornar a uma solução já gerada anteriormente e, portanto, ciclar, a BT guarda em uma lista, a chamada lista

tabu, informações sobre as soluções já geradas. O tamanho dessa lista é um parâmetro do método, que determina quanto tempo o movimento vai permanecer proibido (tabu).

O Algoritmo 12 mostra o pseudocódigo deste procedimento. Os seus parâmetros são: $BTMax$, que representa o número máximo de iterações sem melhora na solução global e o tamanho $|T|$ da lista tabu.

3.5 Reconexão por Caminhos

A técnica Reconexão por Caminhos, do inglês *Path Relinking*, foi proposta por [Glover, 1997] como uma estratégia de intensificação. Para isso, são exploradas trajetórias que conectam boas soluções encontradas ao longo da busca, com o intuito de encontrar soluções ainda melhores.

Para realizar a busca, é construído um conjunto chamado elite, que contém soluções geradas anteriormente pelo algoritmo. Geralmente, os critérios adotados para selecionar os membros são feitos através do valor da função de avaliação (que avalia a solução), e/ou da diversidade em relação às outras soluções do conjunto, para que o conjunto não contenham soluções muito parecidas.

Sendo assim, a Reconexão de Caminhos consiste em gerar e explorar caminhos no espaço de soluções, partindo de uma ou mais soluções elite e levando a outras soluções elite. Como estratégia para se gerar as soluções, são selecionados movimentos que introduzem atributos das soluções guia na solução corrente.

Essa técnica de intensificação pode ser aplicada segundo duas estratégias básicas:

- Reconexão por Caminhos aplicada como uma estratégia de pós-otimização entre todos os pares de soluções elite;
- Reconexão por Caminhos aplicada como uma estratégia de intensificação a cada ótimo local obtido após a fase de busca local.

Como pode ser observado no Algoritmo 13, primeiro é computado a diferença simétrica $\Delta(s, g)$ entre s e g, resultando no conjunto de movimentos que deve ser aplicado a uma delas, dita solução inicial s, para alcançar a outra, dita solução guia g. A partir da solução inicial, o melhor movimento ainda não executado de $\Delta(s, g)$ é aplicado à solução corrente $\bar{g}$ até que a solução guia seja atingida. A melhor solução encontrada ao longo

desta trajetória é considerada como candidata à inserção no conjunto elite e a melhor solução já encontrada é atualizada.

Algoritmo 13 Reconexão-Caminhos

1: $\bar{g} \leftarrow s$;
2: Atribuir a g' a melhor solução entre s e g;
3: Calcular o conjunto de movimentos possíveis $\Delta(s, g)$;
4: **enquanto** ($|\Delta(s, g)| \neq 0$) **faça**
5: Atribuir a g"a melhor solução obtida aplicando o melhor movimento de $\Delta(s, g)$ a $\bar{g}$;
6: Excluir de $\Delta(s, g)$ este movimento;
7: $\bar{g} \leftarrow g$";
8: **se** ($f(\bar{g}) \leq f(g')$) **então**
9: $g' \leftarrow \bar{g}$;
10: **fim se**
11: **fim enquanto**
12: Retorne g';

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.

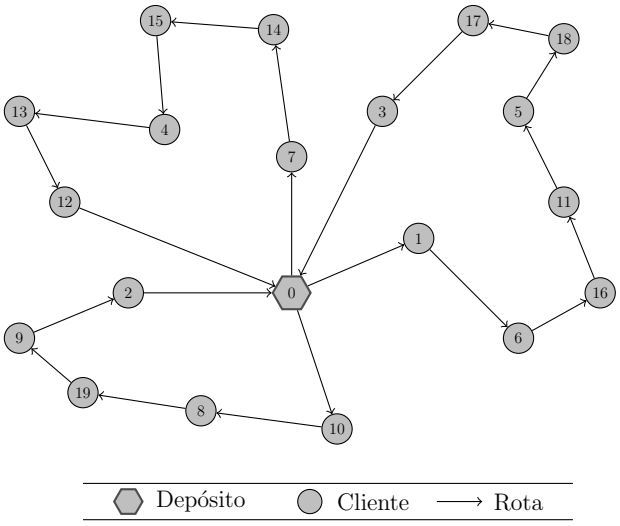

Figura 4.1: Exemplo de uma solução do PRVCES.

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 4.25 ilustra um exemplo em que o cliente 6 é transferido da Rota 2 à Rota 3.

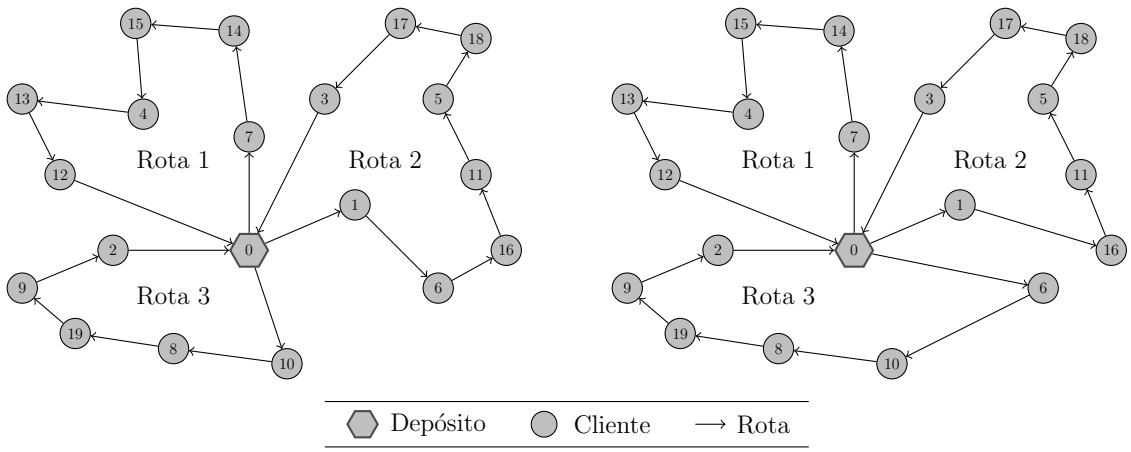

Figura 4.2: Exemplo do movimento *Shift*.

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 1 para a Rota 3.

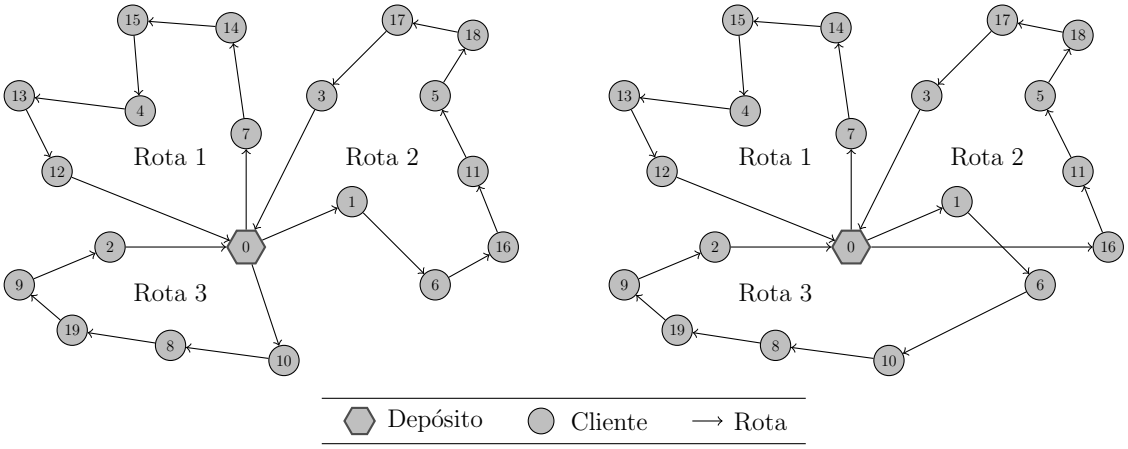

Figura 4.3: Exemplo do movimento *Shift*(2,0).

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.

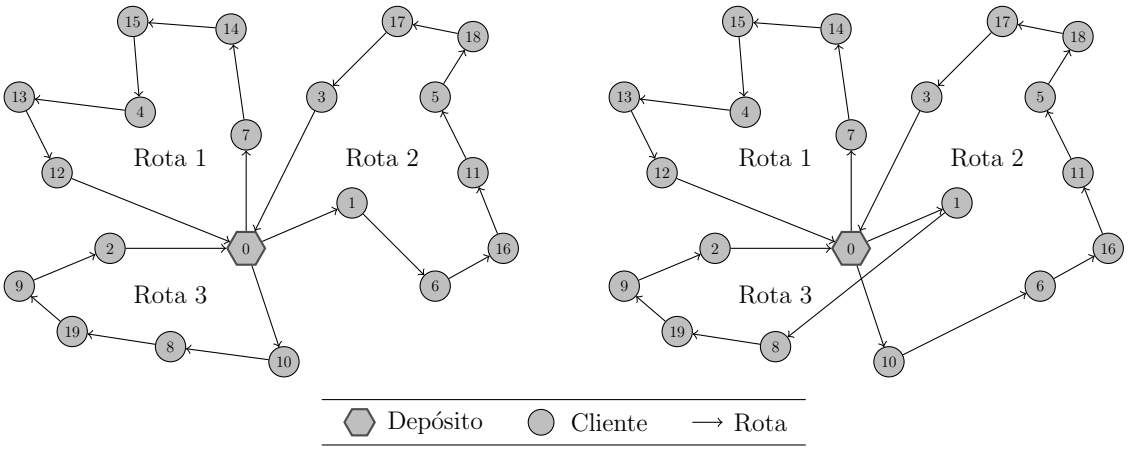

Figura 4.4: Exemplo do movimento *Swap*.

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.

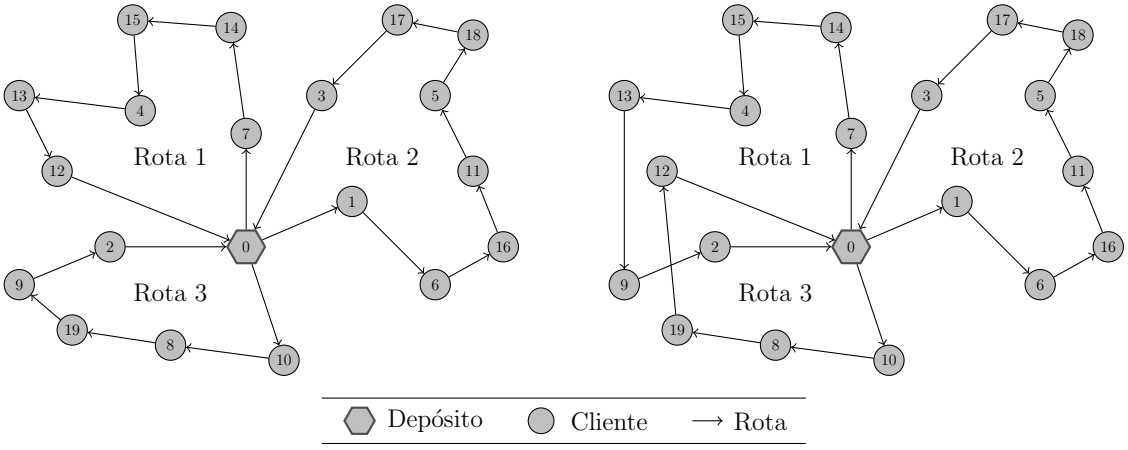

Figura 4.5: Exemplo do movimento *Swap*(2,1).

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.

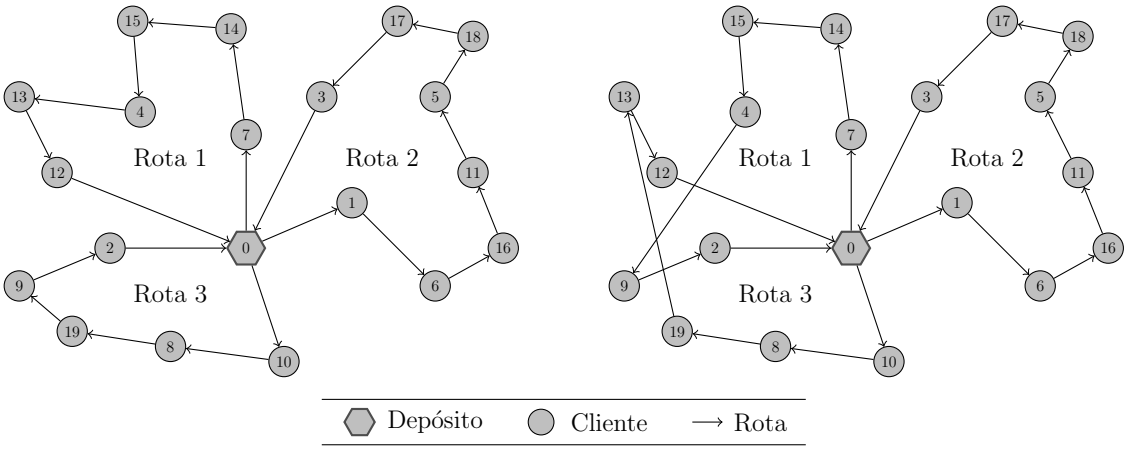

Figura 4.6: Exemplo do movimento *Swap*(2,2).

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 $(0, 6)$.

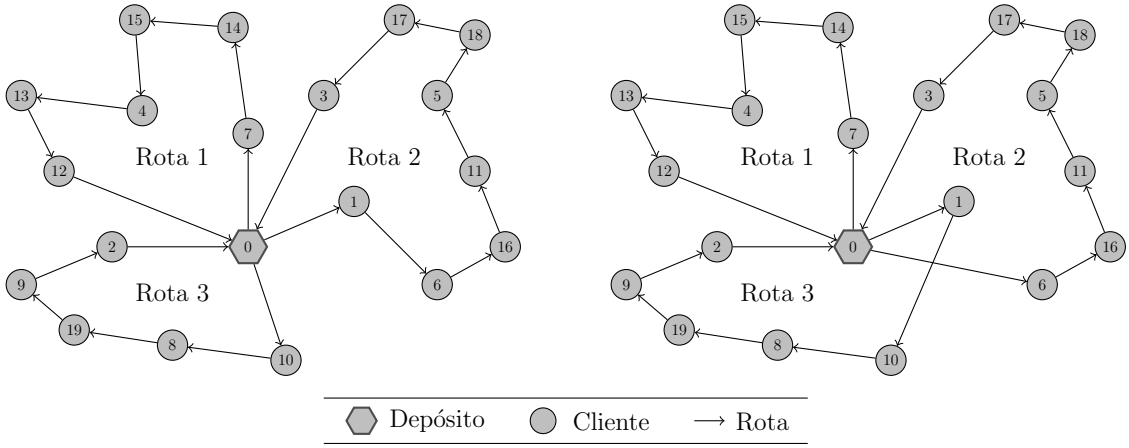

Figura 4.7: Exemplo do movimento *2-Opt*.

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 $(11, 5)$.

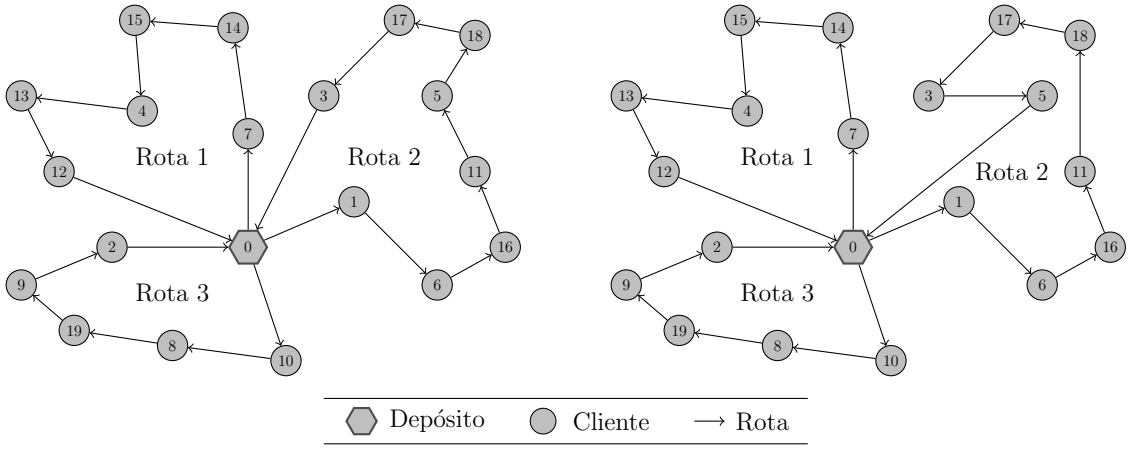

Figura 4.8: Exemplo do movimento *kOrOpt*.

4.3 Função de avaliação

Uma solução é avaliada pela expressão (4.1). A primeira parcela dessa função corresponde à distância total percorrida. Já a segunda parcela é uma penalidade aplicada ao excesso de carga no veículo, ou seja, toda vez que o limite de carga é ultrapassado, o valor excedido é multiplicado por β, que corresponde a um valor suficientemente grande. Vale ressaltar que a geração de soluções inviáveis, onde a carga do veiculo não é respeitada, é permitida, porém não é incentivada, já que a função de avaliação deve ser minimizada.

$$f(s) = \sum_{(i,j)\in A} c_{ij}x_{ij} + \beta \times \sum_{l\in R} \max\{0, \sum_{j\in N_l}(d_j + p_j) - Q\} \tag{4.1}$$

Na função de avaliação f, dada pela expressão (4.1), têm-se:
N: conjunto dos clientes, incluindo o depósito;
A: conjunto de arcos (i, j), com $i, j \in N$;
R: conjunto de rotas existentes na solução;
N_l: conjunto de clientes j pertencentes à rota l, com $j \in N$ e $l \in R$;
c_{ij}: custo de deslocamento ou distância de i a j, com $i, j \in N$;
x_{ij}: variável binária que assume valor 1 se na solução s o arco $(i, j) \in A$ é utilizado ($x_{ij} = 1$) e valor zero ($x_{ij} = 0$), caso contrário.
d_j: quantidade a ser entregue ao cliente $j \in N$;
p_j: quantidade a ser coletada no cliente $j \in N$;
Q: capacidade de carga do veículo;

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_{ij}^k, 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.

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

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 $Q = 120$.

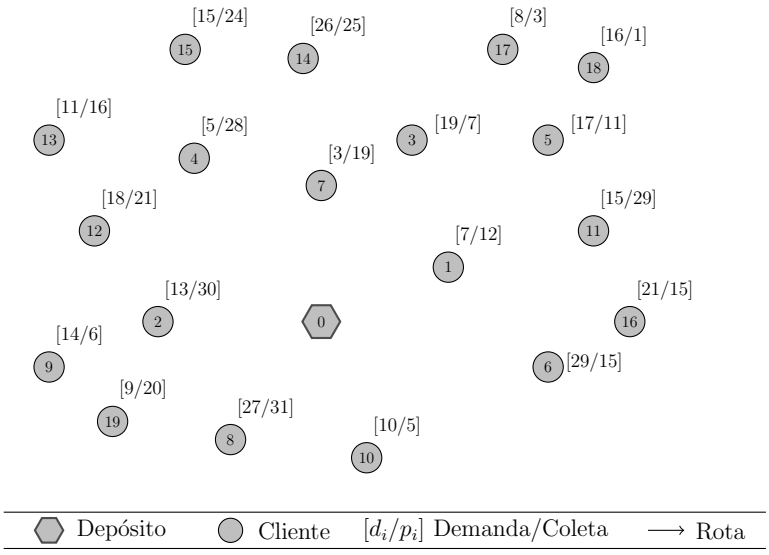

Figura 4.9: Exemplo de um problema envolvendo 19 clientes.

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

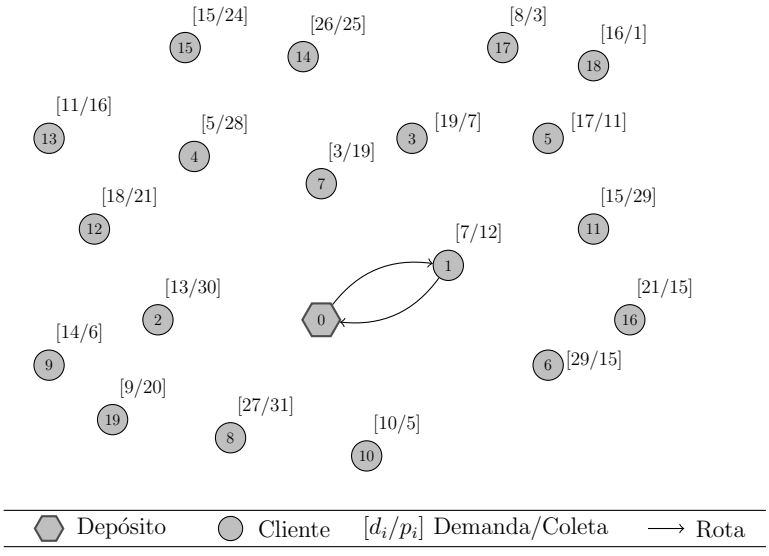

Figura 4.10: Construção de uma rota com o cliente 1.

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.

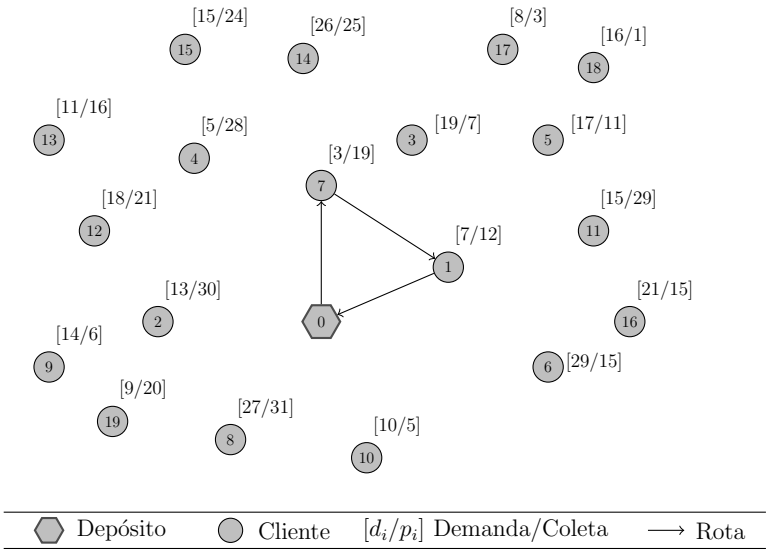

Figura 4.11: Inserção do cliente 7 na rota.

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

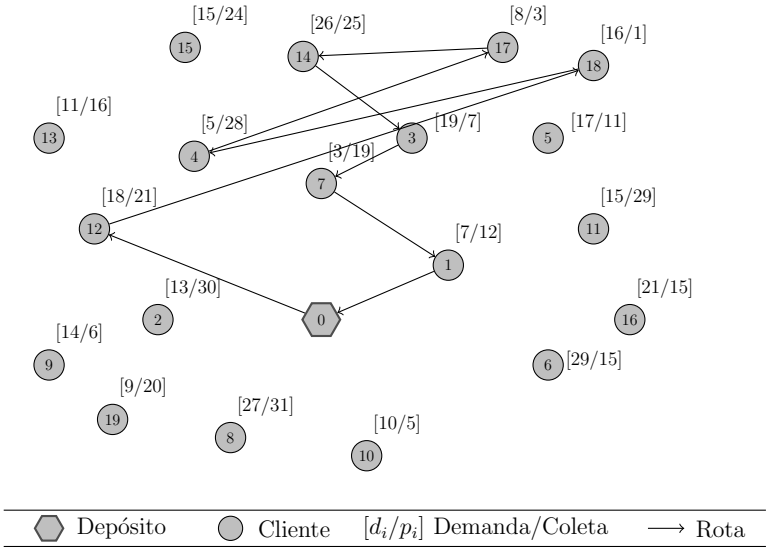

Figura 4.12: Construção completa de uma rota.

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.

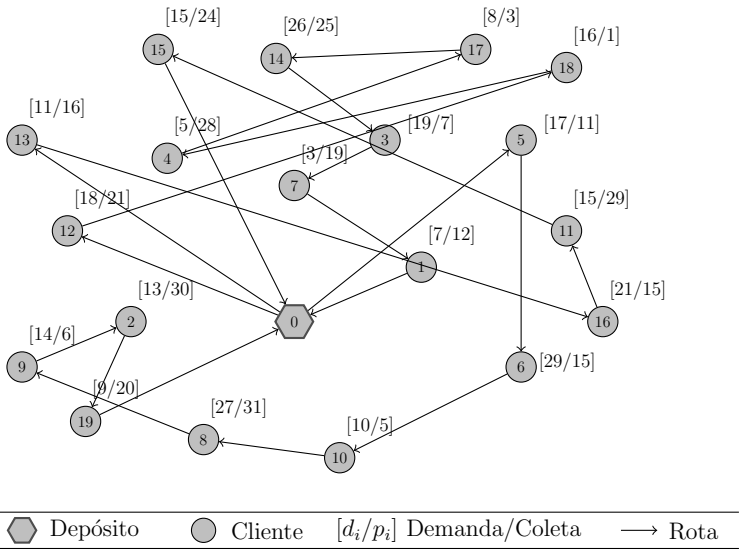

Figura 4.13: Solução gerada pela heurística de inserção mais barata rota a rota.

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 $0,10$.

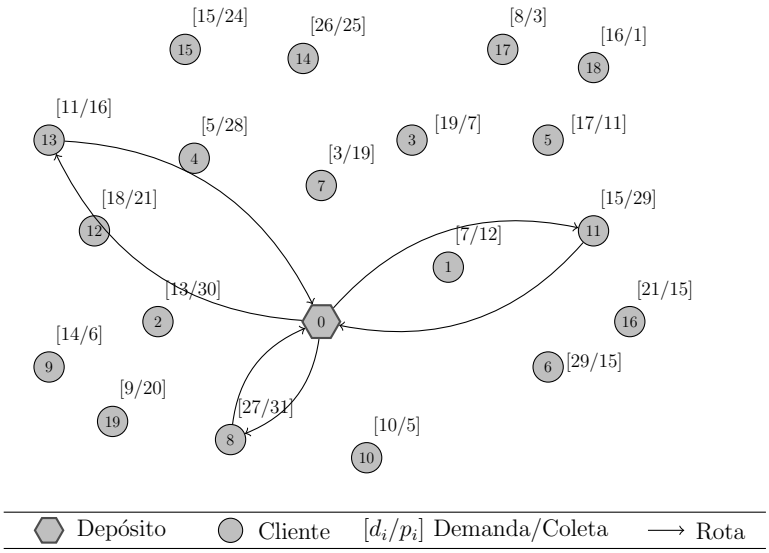

Figura 4.14: Etapa inicial do método, considerando três rotas.

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.

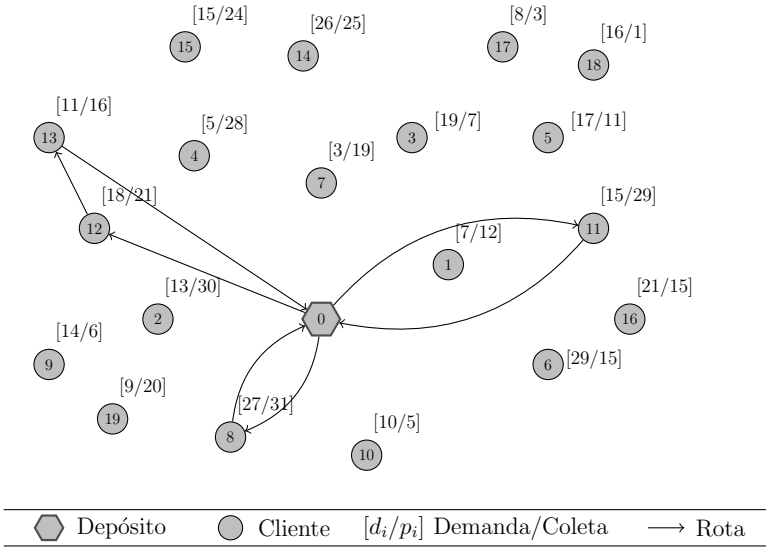

Figura 4.15: Inserção do cliente 12 entre o depósito e o cliente 13.

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.

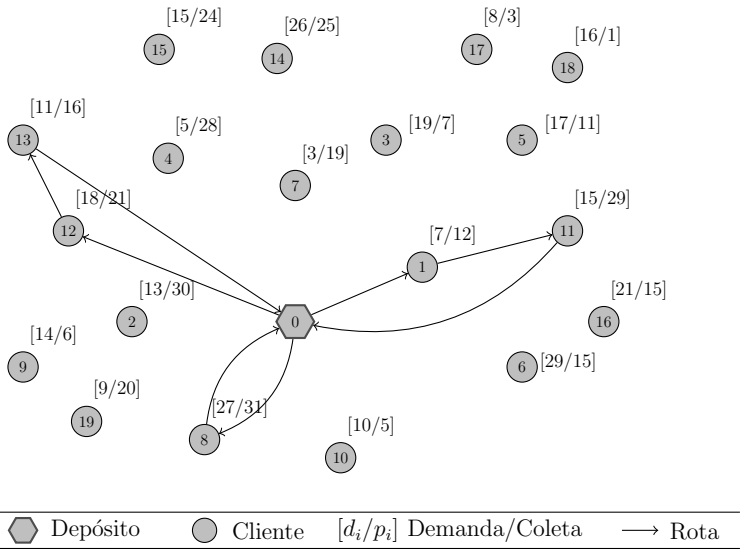

Figura 4.16: Inserção do cliente 1 entre o depósito e o cliente 11.

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

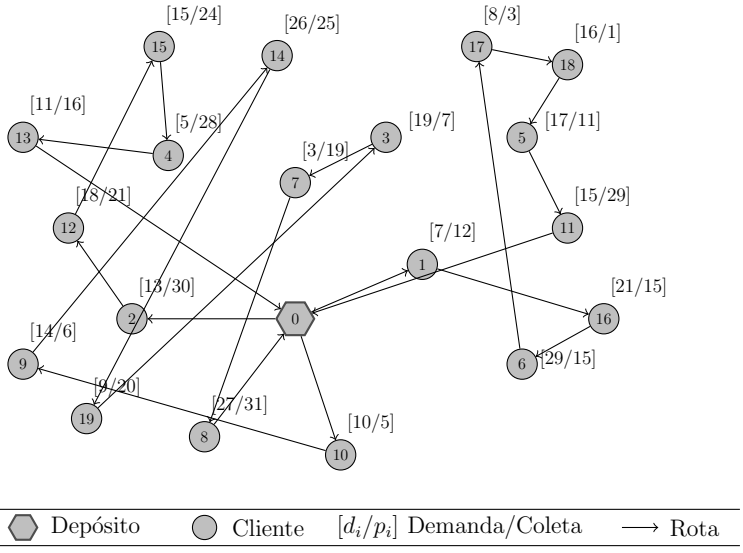

Figura 4.17: Solução gerada pelo método de inserção mais barata com múltiplas rotas.

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.

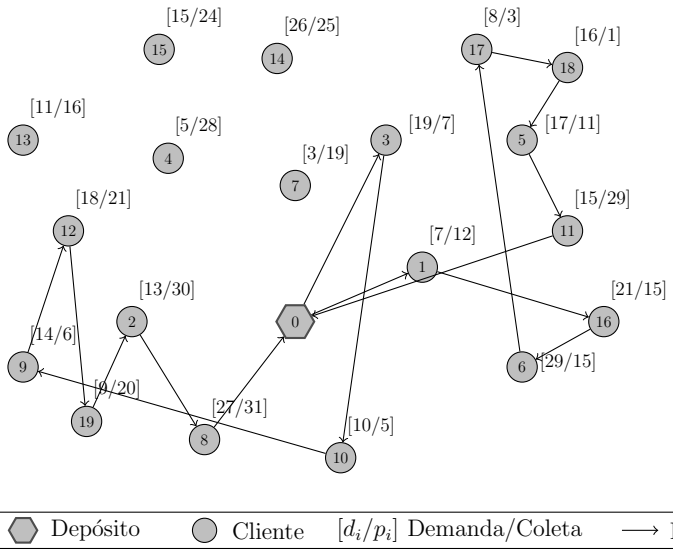

Figura 4.18: Exemplo da geração de uma solução incompleta.

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.

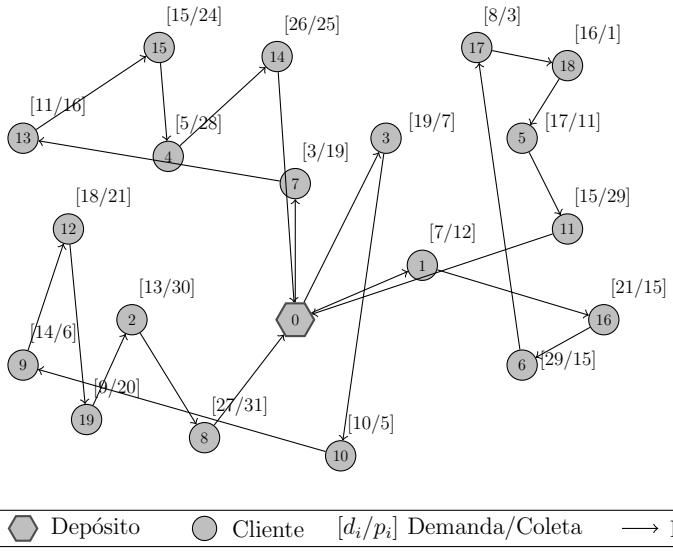

Figura 4.19: Solução gerada pela adaptação do método IMB-MR.

4.4.3 Procedimento construtivo VRGENIUS

O terceiro procedimento usado 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 14 apresenta o pseudocódigo da Fase GENI adaptada para o PRVCES, denominada VRGENI.

Algoritmo 14 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.

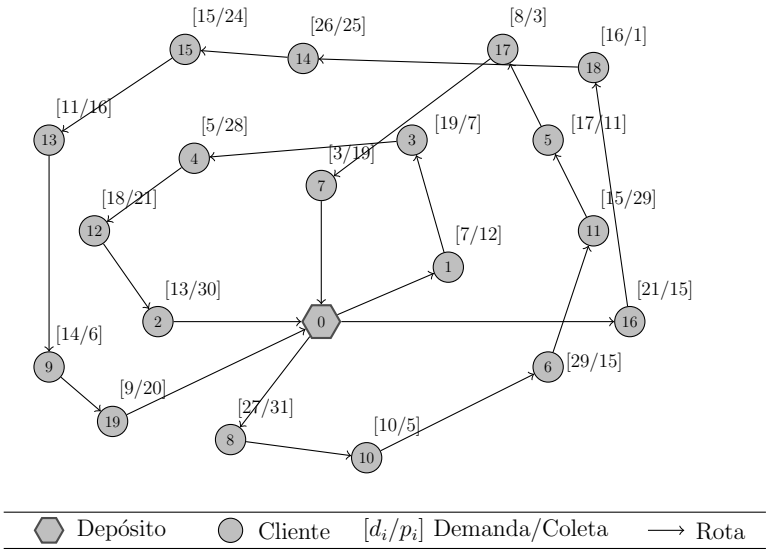

Figura 4.20: Solução gerada pela fase VRGENI.

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 15. 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 15 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.

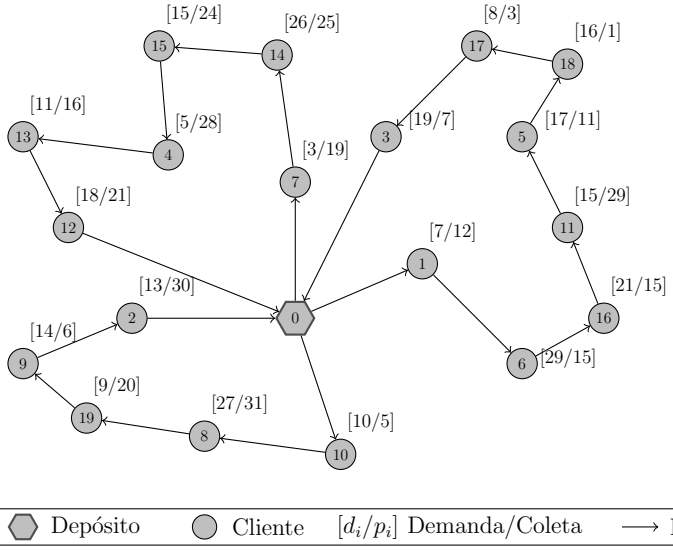

Figura 4.21: Solução gerada pela fase VRUS e pela heurística VRGENIUS.

4.5 Algoritmo proposto

Nesta seção mostra-se o algoritmo proposto, denominado GENILS-TS-CL-PR, para resolver o PRVCES. Esse algoritmo é uma adaptação da metaheurística *Iterated Local Search*, descrita na Subseção 3.4.2, ao PRVCES. O Algoritmo 16 apresenta o seu pseudocódigo.

Algoritmo 16 *GENILS-TS-CL-PR*

1: $\gamma \leftarrow$ número real aleatório no intervalo $[0.0, 0.7]$;
2: $s^A \leftarrow$ *InserçãoMaisBarataRotaARota*(γ);
3: $s^A \leftarrow$ *VND*(s^A);z
4: $nRotas \leftarrow$ número de rotas da solução s^A;
5: $s^B \leftarrow$ *InserçãoMaisBarataMRotas*($\gamma, nRotas$); $s^B \leftarrow$ *VND*(s^B);
6: $s^C \leftarrow$ *VRGENIUS*($nRotas$); $s^C \leftarrow$ *VND*(s^C);
7: $s \leftarrow argmin\{f(s^A), f(s^B), f(s^C)\}$;
8: $iter \leftarrow 1$;
9: **enquanto** ($iter \leq maxIter$) **faça**
10: $s' \leftarrow$ *Perturbação*(s);
11: **se** ($iter \leq iterMaxSemMelhora$) **então**
12: $s'' \leftarrow$ *VND*(s');
13: **senão**
14: $s'' \leftarrow$ *BuscaTabu*(s', *TamListaTabu*, *iterMaxTS*);
15: **fim se**
16: *ReconexaoDeCaminhos(solElite, s")*;
17: **se** ($f(s'') < f(s)$) **então**
18: $s \leftarrow s''$; $iter \leftarrow 1$;
19: **senão**
20: $iter \leftarrow iter + 1$;
21: **fim se**
22: *AtualizaConjElite(s")*;
23: **fim enquanto**
24: Retorne s;

O GENILS-TS-CL-PR 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, inicialmente, o método VND, descrito na Seção 4.6. Após um certo número de iterações sem melhora pelo VND, definida pelo parâmetro *aux*, aciona-se o método de Busca Tabu, descrito na Seção 4.7. O método é interrompido quando o número

máximo de iterações sem melhora na solução corrente ($iterMaxSemMelhora$) é atingido. A cada iteração do ILS também é aplicada a técnica Reconexão por Caminhos (descrita na Seção 4.8) na solução gerada pela busca local s''.

4.5.1 Mecanismos de perturbação

Para realizar uma perturbação na solução corrente, o GENILS-TS-CL-PR 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 VND do GENILS-TS-CL-PR

Descreve-se, a seguir, o procedimento de refinamento baseado em VND do Algoritmo 16, relativo ao GENILS-TS-CL-PR, descrito à página 47.

O procedimento VND implementado é aquele 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*, *Shift(2,0)*, *Swap*, *2-opt*, *Swap(2,1)*, *Swap(2,2)* 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 17 mostra o pseudocódigo do

VND, utilizado como método de busca local do ILS, apresentado na Seção 4.5. Logo, GENILS-TS-CL-PR usa como refinamento o ILS que por sua vez utiliza dois métodos de busca local o VND e a Busca Tabu (Subseção 4.7).

Algoritmo 17 Descida em Vizinhança Variável

1: Seja r o número de estruturas de vizinhança distintas;
2: $RN \leftarrow$ conjunto das vizinhanças N, descritas na Seção 4.2, em ordem aleatória;
3: $k \leftarrow 1$;
4: **enquanto** ($k \leq r$) **faça**
5: Encontre o melhor vizinho $s' \in RN^{(k)}(s)$;
6: **se** ($f(s') < f(s)$) **então**
7: $s \leftarrow s'$;
8: $k \leftarrow 1$;
9: {Intensificação nas rotas alteradas}
10: $s \leftarrow$ *BuscaLocalShift*(s);
11: $s \leftarrow$ *BuscaLocalShift20*(s);
12: $s \leftarrow$ *BuscaLocalSwap*(s);
13: $s \leftarrow$ *BuscaLocal2-Opt*(s);
14: $s \leftarrow$ *BuscaLocalSwap21*(s);
15: $s \leftarrow$ *BuscaLocalSwap22*(s);
16: $s \leftarrow$ *BuscaLocalG3-Opt*(s);
17: $s \leftarrow$ *BuscaLocalG4-Opt*(s);
18: $s \leftarrow$ *BuscaLocalkOr-Opt*(s) com $k = 3, 4, 5$;
19: $s \leftarrow$ *Reverse*(s);
20: **senão**
21: $k \leftarrow k + 1$;
22: **fim se**
23: **fim enquanto**
24: 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 18.

Algoritmo 18 *G3-Opt*

```
1: Seja r uma rota da solução s;
2: r' ← ∅ ⇒ f(r') ← ∞;
3: enquanto ( f(r) ≤ f(r') ) faça
4:    r' ← r;
5:    para ( r'' ∈ { r', r̄' } ) faça
6:       para ( v_i ∈ N^{r'} ) faça
7:          para ( v_j ∈ N_p(v_i), v_j ≠ v_i ) faça
8:             para ( v_k ∈ N_p(v_{i+1}), v_k ≠ v_i, v_j ) faça
9:                r''' ← r'' \ {(v_i, v_{i+1}), (v_j, v_{j+1}), (v_k, v_{k+1})};
10:               r''' ← r''' ∪ {(v_i, v_j), (v_{i+1}, v_k), (v_{j+1}, v_{k+1})};
11:               se ( f(r''') < f(r) ) então
12:                  r ← r''';
13:               fim se
14:            fim para
15:         fim para
16:      fim para
17:   fim para
18: fim enquanto
19: 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 19 apresenta o pseudocódigo do *G4-Opt*.

Algoritmo 19 *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.

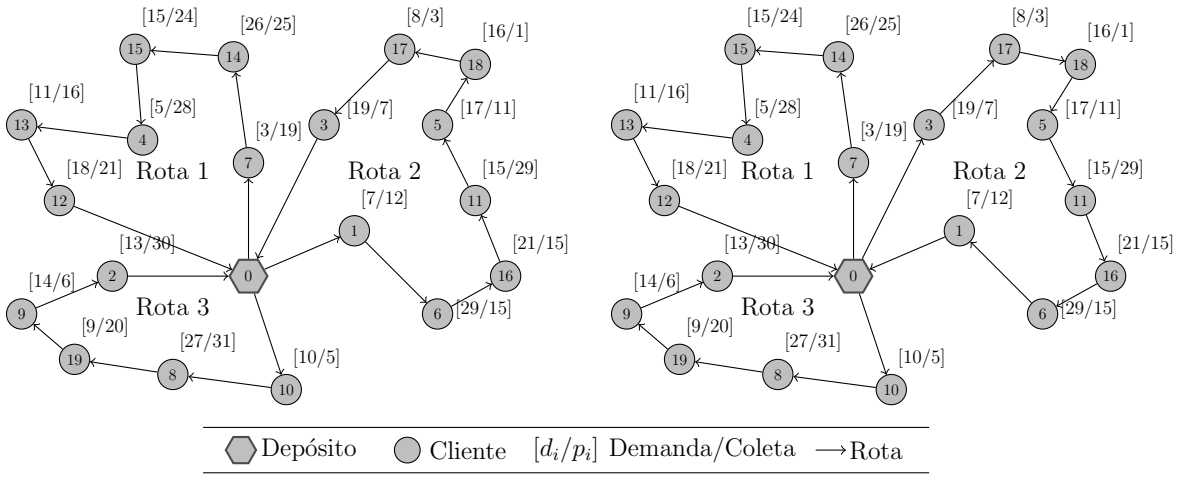

Figura 4.22: Aplicação do procedimento *Reverse* na Rota 2.

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

4.7 Busca Tabu do GENILS-TS-CL-PR

Descreve-se, a seguir, o procedimento de refinamento baseado em Busca Tabu do Algoritmo 16, relativo ao GENILS-TS-CL-PR, descrito à página 47.

O Algoritmo 20 mostra o pseudocódigo desta fase de refinamento.

Algoritmo 20 BUSCA TABU do GENILS-TS-CL-PR

1: $s^* \leftarrow s$; {Melhor solução obtida até então}
2: $SizeOriginal \leftarrow TabuListSize$; {Tamanho original da Lista Tabu}
3: $iter \leftarrow 0$; {Número de iterações}
4: $MelhorIter \leftarrow 0$;{Número de iterações sem melhora na solução}
5: **enquanto** ($iter - MelhorIter \leq iterMaxBT$) **faça**
6: **se** ($iter - MelhorIter \leq IterAux$) **então**
7: $TabuListSize \leftarrow TabuListSize + 1$;
8: **fim se**
9: $s_0 \leftarrow Shift(s)$;
10: $s_1 \leftarrow Swap(s)$;
11: $s_2 \leftarrow Shift(2,0)(s)$;
12: $s_3 \leftarrow Swap(2,1)(s)$;
13: $s_4 \leftarrow Swap(2,2)(s)$;
14: $s \leftarrow \text{argmin}\{f(s_0), f(s_1), f(s_2), f(s_3), f(s_4)\}$;
15: $AtualizaBuscaTabu(TabuListSize, Lista, Ant, Next, iter, \Delta)$ {Algoritmo 21 }
16: **se** ($f(s) < f(s*)$) **então**
17: $s* \leftarrow s$;
18: $MelhorIter \leftarrow iter$;
19: $TabuListSize \leftarrow SizeOriginal$;
20: **fim se**
21: $iter \leftarrow iter + 1$;
22: **fim enquanto**
23: Retorne s^*;

No Algoritmo 20, as cinco soluções s_0, s_1, s_2, s_3 e s_4 são geradas a cada iteração utilizando os movimentos *Shift* (Subseção 4.2.1), *Swap* (Subseção 4.2.3), *Shift (2,0)* (Subseção 4.2.2), *Swap(2,1)* (Subseção 4.2.4) e *Swap(2,2)* (Subseção 4.2.5), sendo que a melhor delas passa a ser a solução corrente s. Cada vez que se move para uma nova solução, a Lista Tabu é atualizada, como descrita na Subseção 4.7.1. O tamanho da Lista Tabu é incrementado em uma unidade à medida que o número de iterações sem melhora vai aumentando; porém, para o algoritmo não ficar extremamente restritivo, essa atualização para de ser efetuada quando certo número de iterações sem melhora é atingido (linhas

6 e 7 do Algoritmo 20). O procedimento é interrompido quando o número máximo de iterações sem melhora na solução é alcançado.

4.7.1 Lista Tabu

O movimento tabu utilizado pelo procedimento TS para evitar o retorno a uma solução gerada anteriormente consiste em proibir que o cliente afetado pelo movimento gerado seja sucessor (*Prox*) do cliente adjacente (*Ant*) a ele antes do movimento. Ou seja, utilizando o exemplo da Figura 4.25, em que é realizado o movimento *Shift*, o cliente 6 é deslocado da Rota 2 para a Rota 3. Sendo assim, é considerado tabu que o cliente 6 seja sucessor do cliente 1, já que eles tinham essa configuração antes do movimento.

Para se atualizar a lista tabu é utilizada uma função descrita no Algoritmo 21.

Algoritmo 21 *Atualiza Busca Tabu (TabuListSize, Lista, Ant, Next, iter, Δ)*

1: $\min \leftarrow TabuListSize + iter - \Delta$;
2: $\max \leftarrow TabuListSize + iter + \Delta$;
3: $TamanhoBT \leftarrow$ *Valor aleatório no intervalo* $[\min, \max]$;
4: $Lista(Ant, Next) \leftarrow TamanhoBT$;
5: Retorne *Lista*;

Neste trabalho foi utilizada uma lista Tabu de tamanho 10, e um Δ igual a 3. Desta forma, a lista assume tamanhos diferentes, devido a aleatoriedade presente no procedimento. Quando o tamanho da lista é menor, ou seja, menos proibitiva, se incentiva a intensificação da busca naquele espaço de soluções, contrastando com a diversificação que ocorre quando a lista é maior. Sendo assim, além de explorar melhor o espaço de soluções, essa estratégia possibilidade evitar a ocorrência de ciclagem.

Para representar computacionalmente a Lista Tabu, utilizou-se uma matriz quadrada, em que as linhas e colunas correspondem aos clientes. Quando o cliente j fica proibido de ser sucessor do cliente i, é atualizada a posição (i, j) dessa matriz. As Figuras 4.23 e 4.24 ilustram essa operação com o valor (variável TamanhoBT) calculado pelo Algoritmo 21. A principal vantagem dessa representação está na ordem de complexidade da consulta para verificar se o movimento é tabu, a qual é $O(1)$. Caso fosse utilizada uma lista encadeada, a consulta seria $O(m)$ no pior caso, considerando m o tamanho da lista.

Na Figura 4.23, o lado esquerdo representa a Lista Tabu em forma de matriz. Considerando que é a primeira iteração, ainda não existe nenhum movimento tabu. Já do lado direito temos a configuração da solução, onde pode ser observado que na rota 2 o veículo

	0	1	2	3	4	5	6	...	19
0	0	0	0	0	0	0	0	...	0
1	0	0	0	0	0	0	0	...	0
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
19	0	0	0	0	0	0	0	...	0

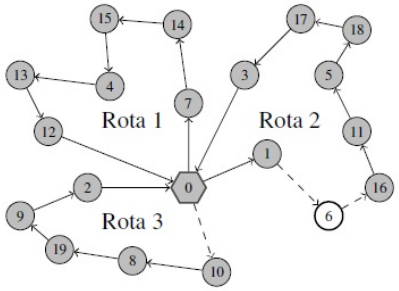

Figura 4.23: Configuração da Matriz antes do Movimento

	0	1	2	3	4	5	6	...	19
0	0	0	0	0	0	0	0	...	0
1	0	0	0	0	0	0	10	...	0
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
19	0	0	0	0	0	0	0	...	0

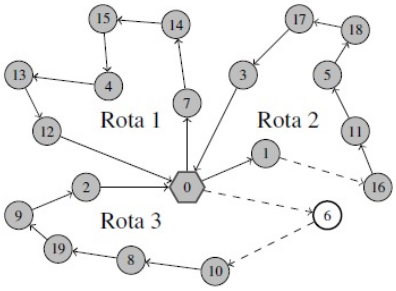

Figura 4.24: Atualização da Matriz após o movimento

sai do depósito e atende aos clientes 1, 6, 16, 11, 5, 18, 17 e 3, nesta ordem, e retorna ao depósito. Já na Figura 4.24, o cliente 6 foi realocado da rota 2 para rota 3 pelo movimento *Shift*. Dessa forma, a atualização da matriz é realizada na linha 1 e coluna 6 da matriz, já que na configuração anterior o cliente 6 é o sucessor do cliente 1. Neste exemplo, qualquer movimento em que o cliente 1 seja sucedido pelo cliente 6 estará proibido até a iteração 10.

4.8 Reconexão por Caminhos

Como forma de fazer um balanço entre intensificação e diversificação, foi utilizada a técnica de Reconexão de Caminhos (PR, do inglês *Path Relinking*) [Glover, 1997], que é aplicada após cada busca local do ILS.

O conjunto elite é composto por cinco soluções. Assim, quando uma nova solução é adicionada ao conjunto e sua capacidade é extrapolada, a solução de pior custo sai, dando lugar à nova solução. Para determinar quais soluções farão parte do conjunto elite foram utilizados os seguintes critérios:

- Se a solução corrente tiver a função de avaliação menor que a melhor solução encontrada até então;
- Se a solução corrente for melhor que a pior do conjunto elite, e for pelo menos 10% diferente das demais soluções do conjunto.

No trabalho de [Ribeiro *et al.*, 2002] foi mostrado que a exploração de duas trajetórias potencialmente diferentes para cada par de soluções consome, aproximadamente, o dobro do tempo de processamento necessário para explorar apenas uma delas, com um ganho marginal muito pequeno em termos de qualidade de solução. Assim, neste trabalho, adotou-se como estratégia utilizar cada uma das soluções do conjunto elite como solução base, e como guia o ótimo local gerado após a aplicação da busca local do procedimento ILS descrito nas Seções 4.6 e 4.7.1. Ou seja, são realizadas cinco aplicações de Reconexão por Caminhos. Caso durante a aplicação desta estratégia seja encontrada uma solução melhor que a melhor já obtida, o procedimento de Reconexão é abortado.

Cada iteração da Reconexão por Caminhos consiste em incluir na solução inicial, aqui chamada de solução base, um atributo da solução guia. O atributo escolhido para ser inserido é aquele que produz na solução base o melhor valor para a função de avaliação. A seguir, à solução com o atributo inserido é aplicada uma busca local que não altere os atributos herdados, no caso, uma descida completa utilizando o movimento *Shift*. Foi considerado como atributo a ser herdado pela solução base, a posição de determinado cliente no vetor referente à solução guia.

Para facilitar o entendimento deste procedimento, a seguir é mostrado um exemplo de seu funcionamento. Seja uma solução base ($sBase$) e uma solução guia ($sGuia$), representadas por vetores com as seguintes configurações:

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

e

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

.

Seguindo o exemplo, o próximo passo é verificar qual cliente está na segunda posição do vetor *sGuia*, nesse caso o cliente 14, já que a primeira posição é o depósito em ambas as soluções. O cliente 14 é colocado, então, na segunda posição da solução base no lugar do cliente 7. O cliente 7 é, então, reposicionado no lugar do cliente 14, isto é, na terceira posiçao de *sBase*. Os vetores a seguir ilustram esta operação.

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

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

A seguir, esta operação é repetida para a terceira posição do vetor *sGuia*, isto é, para o cliente 18. Ele é colocado na terceira posição do vetor *sBase*, em lugar do cliente 14. Este último, por sua vez, ocupa o lugar do cliente 18. Os vetores a seguir mostram o resultado desta operação.

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

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

O procedimento prossegue com a inserção do cliente 11 na quarta posição da solução base, com o cliente 1 na quinta e assim por diante, até incluir o retorno ao depósito (última posição).

Cada inserção de um cliente na solução base é avaliada, e aquela inserção que resultar em um menor valor de função de avaliação é escolhida para se aplicar um procedimento de busca local, no caso, uma descida completa utilizando o movimento *Shift*. Observa-se que nesta busca local o cliente inserido permanece fixo durante o processo, isto é, não muda de posição, pois do contrário não se conseguiria atingir a solução guia.

Ao final deste primeiro passo, um cliente é inserido na solução base na mesma posição da solução guia. O passo seguinte consiste em escolher qual o próximo cliente da solução guia a ser inserido na solução base. Para tanto, repete-se o procedimento do passo anterior, mantendo-se fixa a alocação feita no passo anterior. Este procedimento é repetido até que as soluções tenham as mesmas configurações.

4.9 Lista de Candidatos

Para diminuir a quantidade de possibilidades analisadas pelas estruturas de vizinhança descritas na Seção 4.2, foi utilizada uma Lista de Candidatos (CL, do inglês *Candidate List*). Essa Lista permite apenas que movimentos promissores sejam analisados, descartando aqueles considerados desnecessários.

Pela análise das melhores soluções encontradas pelo algoritmo proposto (sem a lista de candidatos), observou-se que os comprimentos das arestas contidas nessas soluções

eram sempre menores que a média da soma das distâncias entre todos os pares de vértices das instâncias analisadas. Então, teve-se a ideia de utilizar esta métrica para restringir os movimentos que produziriam soluções de má qualidade. A Equação (4.3), descrita a seguir, é, então, usada para excluir movimentos não promissores.

$$f(s) = \left(\sum_{(i,j) \in E} c_{ij} \right) / (n-1)^2 \tag{4.3}$$

Na Equação (4.3), n indica o número de vértices do conjunto e E o conjunto de arestas ligando todos os vértices.

Sendo assim, o critério adotado para restringir a análise, é que um movimento somente é realizado se todas as arestas resultantes forem de comprimento menor que o valor dado pela equação (4.3).

Como exemplo, seja a Figura 4.25, na qual é aplicado o movimento *Shift* ao cliente 6 da Rota 2, transferindo-o para a Rota 3. A estratégia Lista de Candidatos aceita que seja feita a avaliação do vizinho resultante da aplicação desse movimento apenas se o comprimento de cada uma das arestas $0 \rightarrow 6$, $6 \rightarrow 10$ e $1 \rightarrow 16$ for menor que o valor dado pela equação (4.3).

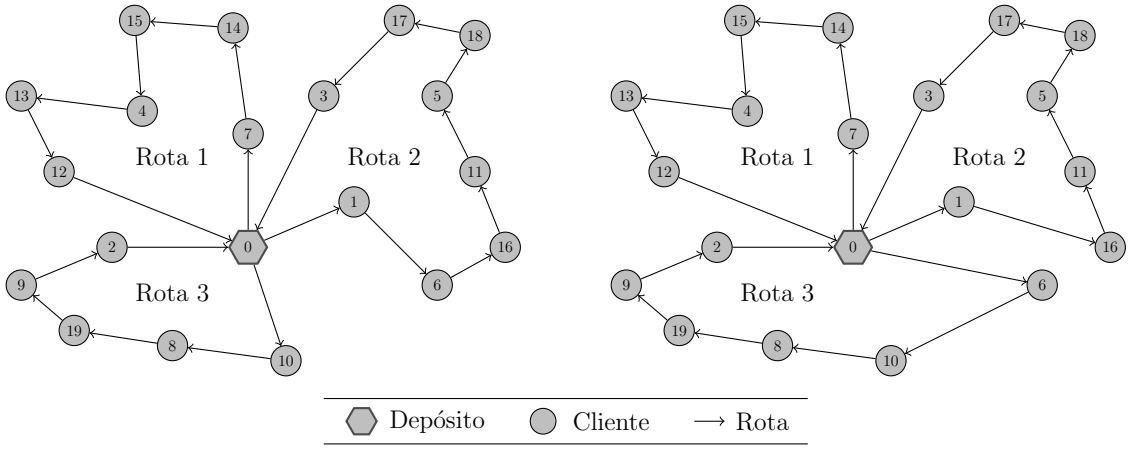

Figura 4.25: Exemplo do movimento *Shift*.

4.10 Paralelização das Estruturas de Vizinhança

A introdução de arquiteturas de processamento paralelo permite que o tempo de processamento de um algoritmo possa ser reduzido dividindo o esforço computacional para

mais de um processador. A proposta de paralelização do algoritmo foi baseada no trabalho de [Gonçalvez, 2010], em que retirou-se informações importantes inerentes à diversos métodos envolvendo multiprocessamento de tarefas, que do termo em inglês é designado como *thread*. Após realizadas diversas pesquisas ligadas à algoritmos paralelos, decidiu-se implementar no atual projeto a paralelização das estruturas de vizinhança.

Internamente em um processador, diversas unidades funcionais executam tarefas em paralelo que estão escondidas no conjunto de instruções da arquitetura do processador. Em algumas aplicações específicas ou determinados programas, os compiladores podem automaticamente detectar e explorar o paralelismo entre vários computadores de uma maneira eficiente. No entanto, na maioria dos casos, os programadores necessitam instruir aos compiladores, quando e onde utilizar ações paralelas. Não é uma tarefa trivial, pois inclui técnicas de replicação de dados, movimentação de dados, balanceamento entre a carga de execução e comunicação.

Para ajudar nesta tarefa, utilizou-se a interface de programação chamada *OpenMP* baseada no modelo de programação paralela de memória compartilhada para arquiteturas de múltiplos processadores. A biblioteca utilizada possui diversos recursos, que por meio de chamada de funções já implementadas, consegue-se paralelizar uma estrutura do algoritmo.

No projeto proposto a paralelização das estruturas de vizinhanças foi realizada de forma a ordenar todos os movimentos descritos na Seção 4.2 e executá-los simultaneamente em cada um dos núcleos disponíveis na máquina. Terminada a busca local com os movimentos alocados nos núcleos do processador, uma nova chamada é realizada para novos movimentos entrarem na lista de execução, conforme ilustração 4.26. Esse procedimento é realizado até que todos os movimentos sejam concluídos. Desta forma, ganha-se tempo no processo de busca local, uma vez que não é necessário esperar que apenas um movimento seja executado até o fim para que em seguida um outro seja processado.

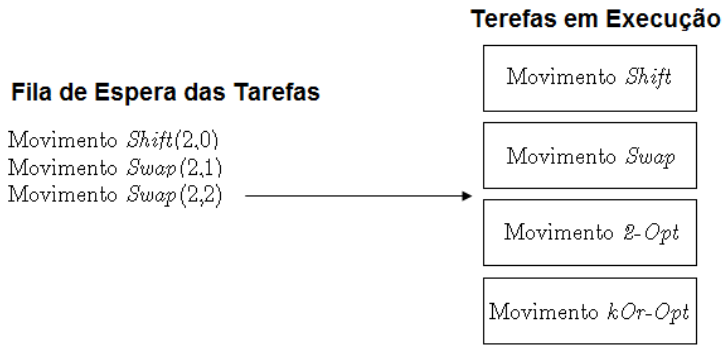

Figura 4.26: Paralelização das Estruturas de Vizinhança

Capítulo 5

Resultados Computacionais

O algoritmo GENILS-TS-CL-PR foi codificado em C++ usando o compilador Visual C++ 2005 e executado em um microcomputador *Quad core*, 1,66 GHz e 4 GB de memória RAM e sistema operacional Windows Vista. O microcomputador possuir quatro núcleos, portanto, o algoritmo proposto explora multiprocessamento para realizar os movimentos da estrutura de vizinhanças.

Para validá-lo, os resultados alcançados pelo GENILS-TS-CL-PR foram comparados com os de outros quatro trabalhos consagrados da literatura: o algoritmo evolutivo de [Zachariadis *et al.*, 2010], o algoritmo paralelo de ILS-RVND de [Subramanian *et al.*, 2010] (256 núcleos), o GENILS de [Mine *et al.*, 2010] e a versão anterior do GENILS-TS de [Silva *et al.*, 2012b]. Destaca-se que os algoritmos propostos por [Zachariadis *et al.*, 2010] e [Mine *et al.*, 2010] eram, de nosso conhecimento, os melhores algoritmos sequenciais conhecidos, enquanto o algoritmo desenvolvido por [Subramanian *et al.*, 2010] é o melhor algoritmo encontrado para o PRVCES.

Para testar o algoritmo foram usados 40 problemas-teste de [Dethloff, 2001]; 14 de [Salhi and Nagy, 1999] e 18 de [Montané and Galvão, 2006]. Os parâmetros adotados foram os seguintes: $TamListaTabu = 10$, $iterMaxTS$ =300, $maxIter = 10000$ e $\Delta = 3$.

Dado seu caráter estocástico, o algoritmo foi executado 50 vezes em cada problema-teste. As Tabelas 5.1, 5.2 e 5.3 apresentam os melhores resultados encontrados pelo GENILS-TS-CL-PR e os compara com os melhores resultados obtidos pelos trabalhos de [Zachariadis *et al.*, 2010], [Subramanian *et al.*, 2010] (256 núcleos) e [Mine *et al.*, 2010]. Nesta tabela, a primeira coluna representa o problema-teste e a segunda o melhor resultado da literatura. As colunas *Melhor* e $Desv^{Best}$ apresentam, respectivamente, o melhor resultado de cada algoritmo e o desvio percentual em relação ao melhor resultado co-

nhecido na literatura. A coluna *Tempo* apresenta o tempo médio das 50 execuções do GENILS-TS-CL-PR, em segundos.

Como pode ser observado na Tabela 5.1, o algoritmo proposto foi capaz de alcançar as melhores soluções conhecidas em todos os problemas-teste de [Dethloff, 2001].

A Tabela 5.2, que contém os resultados dos problemas-teste de [Salhi and Nagy, 1999], verifica-se que o algoritmo de [Subramanian *et al.*, 2010] é o de melhor desempenho. De fato, ele alcança 8 das melhores soluções com um desvio médio de 0,65%, enquanto GENILS-TS-CL-PR obtém 6 melhores resultados da literatura com um desvio médio de 0,69%, melhorando em 0,01% o desvio médio apresentado por [Silva *et al.*, 2011a]. Por sua vez, o algoritmo de [Zachariadis *et al.*, 2010] encontrou 4 melhores soluções e um desvio de 0,76%, enquanto o de [Mine *et al.*, 2010] alcançou 4 das melhores soluções da literatura, mas com um desvio mais elevado, de 0,94%.

Na Tabela 5.3, em que são apresentados os resultados dos problemas-teste propostos em [Montané and Galvão, 2006], o algoritmo de [Subramanian *et al.*, 2010] é o de melhor desempenho, pois encontra a maior quantidade de melhores soluções (15 no total) e com o menor desvio (0,01%), seguido do GENILS-TS-CL-PR com 12 melhores soluções e um desvio de 0,18%. Observa-se que o GENILS-TS-CL-PR, além de não perder para os demais algoritmos com relação à qualidade das melhores soluções e variabilidade das soluções, supera 9 resultados de [Silva *et al.*, 2011a], 6 de [Mine *et al.*, 2010] e 10 de [Zachariadis *et al.*, 2010].

Considerando o conjunto de todos os 72 problemas-teste, em termos de número de melhores resultados, tem-se: [Subramanian *et al.*, 2010]: 64; GENILS-TS-CL-PR: 58, [Mine *et al.*, 2010]: 57 e [Zachariadis *et al.*, 2010]: 52. Já em termos de desvio médio, tem-se: [Subramanian *et al.*, 2010]: 0,22%, GENILS-TS-CL-PR: 0,29%, [Zachariadis *et al.*, 2010]: 0,36% e [Mine *et al.*, 2010]: 0,38%.

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

Problema	MelhorLit	Zachariadis *et al.* (2010)		Subramanian *et al.* (2010)		Mine *et al.*(2010)		Silva *et al.* (2011a)		GENILS-TS-CL-PR		
-	-	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Tempo(s)**
SCA3-0	**635,62**	**635,62**	0,00	**635,62**	0,00	**635,62**	0,00	**635,62**	0,00	**635,62**	0,00	0,74
SCA3-1	**697,84**	**697,84**	0,00	**697,84**	0,00	**697,84**	0,00	**697,84**	0,00	**697,84**	0,00	0,55
SCA3-2	**659,34**	**659,34**	0,00	**659,34**	0,00	**659,34**	0,00	**659,34**	0,00	**659,34**	0,00	0,27
SCA3-3	**680,04**	**680,04**	0,00	**680,04**	0,00	**680,04**	0,00	**680,04**	0,00	**680,04**	0,00	10,06
SCA3-4	**690,50**	**690,50**	0,00	**690,50**	0,00	**690,50**	0,00	**690,50**	0,00	**690,50**	0,00	0,08
SCA3-5	**659,90**	**659,90**	0,00	**659,90**	0,00	**659,90**	0,00	**659,90**	0,00	**659,90**	0,00	0,01
SCA3-6	**651,09**	**651,09**	0,00	**651,09**	0,00	**651,09**	0,00	**651,09**	0,00	**651,09**	0,00	0,95
SCA3-7	**659,17**	**659,17**	0,00	**659,17**	0,00	**659,17**	0,00	**659,17**	0,00	**659,17**	0,00	0,54
SCA3-8	**719,48**	**719,48**	0,00	**719,48**	0,00	**719,48**	0,00	**719,48**	0,00	**719,48**	0,00	0,97
SCA3-9	**681,00**	**681,00**	0,00	**681,00**	0,00	**681,00**	0,00	**681,00**	0,00	**681,00**	0,00	0,08
SCA8-0	**961,50**	**961,50**	0,00	**961,50**	0,00	**961,50**	0,00	**961,50**	0,00	**961,50**	0,00	2,97
SCA8-1	**1049,65**	**1049,65**	0,00	**1049,65**	0,00	**1049,65**	0,00	**1049,65**	0,00	**1049,65**	0,00	0,72
SCA8-2	**1039,64**	**1039,64**	0,00	**1039,64**	0,00	**1039,64**	0,00	**1039,64**	0,00	**1039,64**	0,00	2,11
SCA8-3	**983,34**	**983,34**	0,00	**983,34**	0,00	**983,34**	0,00	**983,34**	0,00	**983,34**	0,00	0,50
SCA8-4	**1065,49**	**1065,49**	0,00	**1065,49**	0,00	**1065,49**	0,00	**1065,49**	0,00	**1065,49**	0,00	0,39
SCA8-5	**1027,08**	**1027,08**	0,00	**1027,08**	0,00	**1027,08**	0,00	**1027,08**	0,00	**1027,08**	0,00	0,12
SCA8-6	**971,82**	**971,82**	0,00	**971,82**	0,00	**971,82**	0,00	**971,82**	0,00	**971,82**	0,00	3,24
SCA8-7	**1051,28**	**1051,28**	0,00	**1051,28**	0,00	**1051,28**	0,00	**1051,28**	0,00	**1051,28**	0,00	1,89
SCA8-8	**1071,18**	**1071,18**	0,00	**1071,18**	0,00	**1071,18**	0,00	**1071,18**	0,00	**1071,18**	0,00	0,70
SCA8-9	**1060,50**	**1060,50**	0,00	**1060,50**	0,00	**1060,50**	0,00	**1060,50**	0,00	**1060,50**	0,00	0,44
CON3-0	**616,52**	**616,52**	0,00	**616,52**	0,00	**616,52**	0,00	**616,52**	0,00	**616,52**	0,00	0,37
CON3-1	**554,47**	**554,47**	0,00	**554,47**	0,00	**554,47**	0,00	**554,47**	0,00	**554,47**	0,00	0,76
CON3-2	**518,00**	**518,00**	0,00	**518,00**	0,00	**518,00**	0,00	**518,00**	0,00	**518,00**	0,00	0,81
CON3-3	**591,19**	**591,19**	0,00	**591,19**	0,00	**591,19**	0,00	**591,19**	0,00	**591,19**	0,00	0,32
CON3-4	**588,79**	**588,79**	0,00	**588,79**	0,00	**588,79**	0,00	**588,79**	0,00	**588,79**	0,00	0,42
CON3-5	**563,70**	**563,70**	0,00	**563,70**	0,00	**563,70**	0,00	**563,70**	0,00	**563,70**	0,00	0,23
CON3-6	**499,05**	**499,05**	0,00	**499,05**	0,00	**499,05**	0,00	**499,05**	0,00	**499,05**	0,00	0,24
CON3-7	**576,48**	**576,48**	0,00	**576,48**	0,00	**576,48**	0,00	**576,48**	0,00	**576,48**	0,00	0,34
CON3-8	**523,05**	**523,05**	0,00	**523,05**	0,00	**523,05**	0,00	**523,05**	0,00	**523,05**	0,00	0,43
CON3-9	**578,25**	**578,25**	0,00	**578,25**	0,00	**578,25**	0,00	**578,25**	0,00	**578,25**	0,00	0,64
CON8-0	**857,17**	**857,17**	0,00	**857,17**	0,00	**857,17**	0,00	**857,17**	0,00	**857,17**	0,00	0,49
CON8-1	**740,85**	**740,85**	0,00	**740,85**	0,00	**740,85**	0,00	**740,85**	0,00	**740,85**	0,00	1,20
CON8-2	**712,89**	**712,89**	0,00	**712,89**	0,00	**712,89**	0,00	**712,89**	0,00	**712,89**	0,00	0,34
CON8-3	**811,07**	**811,07**	0,00	**811,07**	0,00	**811,07**	0,00	**811,07**	0,00	**811,07**	0,00	0,21
CON8-4	**772,25**	**772,25**	0,00	**772,25**	0,00	**772,25**	0,00	**772,25**	0,00	**772,25**	0,00	1,14
CON8-5	**754,88**	**754,88**	0,00	**754,88**	0,00	**754,88**	0,00	**754,88**	0,00	**754,88**	0,00	0,67
CON8-6	**678,92**	**678,92**	0,00	**678,92**	0,00	**678,92**	0,00	**678,92**	0,00	**678,92**	0,00	0,86
CON8-7	**811,96**	**811,96**	0,00	**811,96**	0,00	**811,96**	0,00	**811,96**	0,00	**811,96**	0,00	1,18
CON8-8	**767,53**	**767,53**	0,00	**767,53**	0,00	**767,53**	0,00	**767,53**	0,00	**767,53**	0,00	2,97
CON8-9	**809,00**	**809,00**	0,00	**809,00**	0,00	**809,00**	0,00	**809,00**	0,00	**809,00**	0,00	1,65
Média	**-**	-	0,00	-	0,00	-	0,00	-	0,00	-	0,00	-

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

Problema	MelhorLit	Zachariadis *et al.* (2010)		Subramanian *et al.* (2010)		Mine *et al.*(2010)		Silva *et al.* (2011a)		GENILS-TS-CL-PR		
-	-	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Tempo(s)**
CMT1X	**466,77**	469,80	0,65	**466,77**	0,00	**466,77**	0,00	**466,77**	0,00	**466,77**	0,00	2,25
CMT1Y	**466,77**	469,80	0,65	**466,77**	0,00	**466,77**	0,00	**466,77**	0,00	**466,77**	0,00	32,43
CMT2X	**668,77**	684,21	2,31	684,21	2,31	684,11	2,29	684,11	2,29	684,11	2,29	98,65
CMT2Y	**663,25**	684,21	3,16	684,21	3,16	684,11	3,15	684,11	3,15	684,11	3,15	78,45
CMT3X	**721,27**	**721,27**	0,00	**721,27**	0,00	**721,40**	0,02	**721,27**	0,00	**721,27**	0,00	160,65
CMT3Y	**721,27**	**721,27**	0,00	**721,27**	0,00	**721,27**	0,00	**721,27**	0,00	**721,27**	0,00	165,54
CMT12X	**644,7**	662,22	2,72	662,22	2,72	662,22	2,72	662,22	2,72	662,22	2,72	123,60
CMT12Y	**659,52**	662,22	0,41	662,22	0,41	663,50	0,60	663,50	0,60	662,22	0,41	78,41
CMT11X	**833,92**	**833,92**	0,00	**833,92**	0,00	846,23	1,48	**833,92**	0,00	**833,92**	0,00	300,52
CMT11Y	**830,39**	833,92	0,43	833,92	0,43	836,04	0,68	833,92	0,43	833,92	0,43	541,76
CMT4X	**852,46**	**852,46**	0,00	**852,46**	0,00	**852,46**	0,00	**852,46**	0,00	**852,46**	0,00	256,76
CMT4Y	**852,35**	852,46	0,01	**852,46**	0,01	862,28	1,17	855,52	0,37	855,52	0,37	137,98
CMT5X	**1029,25**	1030,55	0,13	**1029,25**	0,00	1033,51	0,41	1030,5	0,13	1030,50	0,12	689,41
CMT5Y	**1029,25**	1030,55	0,13	**1029,25**	0,00	1036,14	0,67	1030,5	0,13	1030,50	0,12	489,52
Média	**-**	-	0,76	-	0,65	-	0,94	-	0,70	-	0,69	-

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

Problema	MelhorLit	Zachariadis *et al.* (2010)		Subramanian *et al.* (2010)		Mine *et al.*(2010)		Silva *et al.* (2011a)		GENILS-TS-CL-PR		
-	-	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Tempo(s)**
r101	**1009,95**	**1009,95**	0,00	**1009,95**	0,00	**1009,95**	0,00	**1009,95**	0,00	**1009,95**	0,00	108,48
r201	**666,20**	**666,20**	0,00	**666,20**	0,00	**666,20**	0,00	**666,20**	0,00	**666,20**	0,00	127,92
c101	**1220,18**	1220,99	0,07	**1220,18**	0,00	**1220,18**	0,00	**1220,18**	0,00	**1220,18**	0,00	139,49
c201	**662,07**	**662,07**	0,00	**662,07**	0,00	**662,07**	0,00	**662,07**	0,00	**662,07**	0,00	47,53
rc101	**1059,32**	**1059,32**	0,00	**1059,32**	0,00	**1059,32**	0,00	**1059,32**	0,00	**1059,32**	0,00	32,85
rc201	**672,92**	**672,92**	0,00	**672,92**	0,00	**672,92**	0,00	**672,92**	0,00	**672,92**	0,00	167,53
r1_2_1	**3357,64**	3376,30	0,55	3360,02	0,07	**3357,64**	0,00	**3357,64**	0,00	**3357,64**	0,00	420,67
r2_2_1	**1665,58**	**1665,58**	0,00	**1665,58**	0,00	**1665,58**	0,00	1672,5	0,42	**1665,58**	0,00	795,39
c1_2_1	**3629,89**	3643,82	0,38	**3629,89**	0,00	3636,74	0,19	3645,45	0,43	3634,65	0,13	642,99
c2_2_1	**1726,59**	**1726,59**	0,00	**1726,59**	0,00	**1726,59**	0,00	1731,74	0,30	**1726,58**	0,00	461,37
rc1_2_1	**3306,00**	3323,56	0,53	**3306,00**	0,00	3312,92	0,21	3344,79	1,17	3312,92	0,21	692,71
rc2_2_1	**1560,00**	**1560,00**	0,00	**1560,00**	0,00	**1560,00**	0,00	**1560,00**	0,00	**1560,00**	0,00	74,21
r1_4_1	**9605,75**	9691,60	0,89	**9605,75**	0,00	9627,43	0,23	9627,88	0,23	9627,88	0,23	1568,23
r2_4_1	**3551,38**	3572,38	0,59	**3551,38**	0,00	3582,08	0,86	3595,88	1,25	3582,08	0,86	1967,34
c1_4_1	**11098,21**	11179,36	0,73	11099,54	0,01	**11098,21**	0,00	11125,55	0,25	**11098,21**	0,00	3198,49
c2_4_1	**3546,10**	3549,27	0,09	**3546,10**	0,00	3596,37	1,42	3605,22	1,67	3592,71	1,31	2071,48
rc1_4_1	**9535,46**	9645,27	1,14	9536,77	0,01	**9535,46**	0,00	**9535,46**	0,00	**9535,46**	0,00	4820,13
rc2_4_1	**3403,70**	3423,62	0,58	**3403,70**	0,00	3422,11	0,54	3432,41	0,84	3419,76	0,47	3281,65
Média	**-**	-	0,31	-	0,01	-	0,19	-	0,36	-	0,18	-

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho teve seu foco no Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Este problema envolve um conjunto de clientes, cada qual com uma demanda d_i e coleta p_i a serem completamente atendidas, e um depósito, que é o local onde ficam armazenados os produtos a serem entregues aos clientes ou coletados desses, e um conjunto de veículos de capacidade Q de carga, os quais são utilizados para fazer as operações de entrega e coleta. O objetivo é construir um conjunto de rotas, a custo mínimo, que iniciam e terminam no depósito, e atendam a todos os clientes sem ultrapassar a capacidade dos veículos.

Dada a dificuldade de resolução desse problema em tempos computacionais aceitáveis no caso geral, foi proposto um algoritmo heurístico que combina os procedimentos *Iterated Local Search*, *Variable Neighborhood Descent*, Busca Tabu, Inserção Mais Barata Rota a Rota, Inserção Mais Barata com Múltiplas Rotas e uma adaptação da heurística GENIUS. Os três últimos são usados para gerar uma solução inicial e o VND é usado como busca local do ILS.

O GENILS foi aperfeiçoado com a introdução de um módulo de Busca Tabu (TS, do inglês *Tabu Search*), de uma lista de candidatos (CL, do inglês *Candidate List*) para a execução das buscas locais, e de um módulo de Reconexão por Caminhos (PR, do inglês *Path Relinking*), resultando no algoritmo nomeado GENILS-TS-CL-PR.

O módulo de Busca Tabu é acionado depois de um certo número de iterações sem melhora usando o VND como busca local do ILS. A lista de candidatos é utilizada para evitar o uso de movimentos desnecessários e a Reconexão por Caminhos é aplicada após cada ótimo local encontrado pelos módulos de busca local (VND ou Busca Tabu). A Reconexão por Caminhos faz um balanço entre intensificação e diversificação, reconectando

ótimos locais produzidos após cada busca local e as soluções elite. A Reconexão é aplicada em uma sequência de cinco pares de soluções, sendo cada par formado pelo ótimo local gerado pela busca local do ILS e uma das cinco soluções elite formadas durante a busca.

O algoritmo proposto, com cada um de seus módulos tomados individualmente, foi testado em um conjunto consagrado de problemas-teste da literatura.

Inicialmente, verificou-se a influência da inserção do módulo de Busca Tabu no algoritmo GENILS. Para tanto, o GENILS foi comparado com o algoritmo GENILS-TS-CL-PR, isto é, com o algoritmo GENILS tendo o módulo de Busca Tabu acionado após certo número de iterações sem melhora com o VND. Desta forma, para se obter uma comparação detalhada dos algoritmos, foram feitos dois testes. No primeiro, adotou-se como critério de parada do ILS, tanto no GENILS quanto no GENILS-TS-CL-PR, o número máximo de iterações sem melhora. Já na segunda fase de testes, utilizou-se como critério de parada do ILS um tempo previamente determinado. Os dois testes apresentaram resultados bastante semelhantes, sendo que em ambos os casos, o GENILS-TS-CL-PR superou o GENILS, pois foi capaz de encontrar um maior número de melhores soluções e com menor variabilidade. Também foi mostrado que há diferença estatística entre os dois algoritmos, com grau de confiança de 95%. Ainda como ponto positivo, o GENILS-TS-CL-PR exige um tempo computacional muito bom, visto que houve uma melhora significativa com a paralelização das estruturas de vizinhança.

Assim, de forma a tentar diminuir esse tempo computacional, sem prejuízo da qualidade das soluções finais, foi inserida uma Lista de Candidatos na exploração do espaço de soluções. Essa lista consiste em restringir a busca apenas a movimentos que não gerem arcos com comprimento maior que a média do comprimento das arestas que compõem a instância do problema. Essa variante do algoritmo, denominada GENILS-TS-CL, foi comparada com o algoritmo GENILS-TS em termos de capacidade de encontrar as melhores soluções, variabilidade das soluções finais e tempo de processamento. Mostrou-se que essa versão melhorou significativamente o tempo médio, quando comparado à versão que não utiliza a lista de candidatos, e que manteve a capacidade de encontrar as melhores soluções. Entretanto, a variabilidade das soluções foi pouco maior.

Posteriormente, como forma de diminuir a variabilidade das soluções finais, foi acionado um mecanismo de Reconexão por Caminhos à versão anterior. Tal algoritmo, nomeado GENILS-TS-CL-PR, foi comparado com a versão que não aciona o módulo de Reconexão, assim como com três outros algoritmos da literatura, no caso, os algoritmos desenvolvidos por [Subramanian *et al.*, 2010], [Mine *et al.*, 2010] e [Zachariadis *et al.*, 2010].

Observou-se que o algoritmo GENILS-TS-CL-PR reduziu a variabilidade das soluções finais, porém com um tempo de processamento maior que o do algoritmo GENILS-TS-CL, mas ainda menor que o do algoritmo GENILS-TS. Em relação à comparação com os algoritmos da literatura, verificou-se que o algoritmo proposto é o segundo melhor em termos de capacidade de encontrar as melhores soluções e também em termos de variabilidade das soluções finais.

O algoritmo foi testado em um conjunto de problemas-teste da literatura e comparado com quatro algoritmos eficientes da literatura. O algoritmo de [Subramanian *et al.*, 2010] foi, claramente, o de melhor desempenho. Entretanto, esse algoritmo usa 256 núcleos de processamento para explorar o espaço de soluções, enquanto os demais usam apenas um. O GENILS-TS-CL-PR, por sua vez, teve o segundo melhor desempenho, em termos de desvio médio e quantidade de melhores soluções. Tendo em vista a estrutura computacional sofisticada requerida para aplicação do algoritmo paralelo de [Subramanian *et al.*, 2010], e que essa não é provavelmente a realidade da maioria das empresas de transporte, o GENILS-TS-CL-PR pode ser considerado, assim, uma alternativa mais factível, pois o recurso de paralelização implementado foi executado em computadores comerciais de quatro núcleos que são altamente comercilizados atualmente, visto que possuem custo de compra baixo.

Quanto à paralelização das estruturas de vizinhança do algoritmo GENILS-TS-CL-PR percebeu-se um forte ganho no tempo de execução, uma vez que são executados diferentes tipos de movimentos simultaneamente. Salienta-se que a estrutura do algoritmo continua a mesma, ou seja, a alteração foi apenas na arquitetura das vizinhanças, que por meio da biblioteca *OpenMP* da linguagem C++, os movimentos são realizados em forma de *Thread*. Sendo assim, os movimentos são divididos e executados concorrentemente.

A ideia da paralelização teve o objetivo alcançado de diminuir o tempo computacional, mas levou-se em consideração, também, o recurso computacional disponível nas máquinas atuais. Como atualmente os computadores multiprocessados com até quatro núcleos possuem bom custo benefício, o algoritmo proposto ainda é uma ótima solução para as empresas de transporte.

Considerando a métrica implantada, os resultados apresentados pelo GENILS-TS-CL-PR mostram que o algoritmo é promissor, já que alcançou bons resultados em comparação com aqueles produzidos pelo algoritmo de [Zachariadis *et al.*, 2010], melhorou muitos resultados de [Mine *et al.*, 2010] e atingiu grande parte dos resultados alcançados pelo algoritmo paralelo de [Subramanian *et al.*, 2010]. Tais resultados mostram, claramente,

que o aperfeiçoamento proposto no GENILS foi eficiente.

Destacamos, finalmente, que houve um ganho bastante considerável nos resultados do artigo [Silva *et al.*, 2011b] se comparado com a publicação anterior de [Silva *et al.*, 2011a]. Quanto à nova proposta, que rendeu as publicações [Silva *et al.*, 2012a] e [Silva *et al.*, 2012b], nota-se que os resultados encontrados se mantiveram, mas deve-se salientar o ganho em tempo de solução do problema, o qual caiu consideravelmente, principalmente depois da implementação da paralelização das estruturas de vizinhança. Portanto, as melhorias realizadas fizeram com que o algoritmo proposto seja considerado o segundo melhor em desempenho da literatura, atrás apenas do de [Subramanian *et al.*, 2010].

Como toda pesquisa, não se esgotam os aperfeiçoamentos possíveis para este algoritmo. Como trabalho futuro pretende-se sofisticar a Busca Tabu com a inserção de outras estruturas de vizinhança.

Capítulo 7

Produções

Este projeto, com cronograma de duração de dois anos, proporcionou a publicação de seis artigos em eventos científicos nacionais e internacionais. São eles: [Silva *et al.*, 2011a], publicado nos anais do Simpósio Brasileiro de Pesquisa Operacional (XLIII SBPO 2011); [Silva *et al.*, 2012b], aceito para publicação nos anais do XIV SBPO 2012; [Silva *et al.*, 2011b], publicado nos anais do Simpósio de Pesquisa Operacional & Logística da Marinha (XIV SPOLM 2011) e [Silva *et al.*, 2012a], aceito para publicação nos anais do XV SPOLM 2012. No âmbito internacional, o trabalho [Cruz *et al.*, 2012a] foi aceito na *International Conference on Metaheuristics and Nature Inspired Computing* (META 2012), a se realizar na França e o trabalho [Cruz *et al.*, 2012b], aceito para publicação na Euro mini conference on VNS (`http://toledo.mi.sanu.ac.rs/~grujicic/vnsconference/`), a ocorrer em Belgrado, entre 4 e 7 de outubro de 2012.

O Simpósio Brasileiro de Pesquisa Operacional é um evento anual promovido pela Sociedade Brasileira de Pesquisa Operacional (SOBRAPO), fundada em 1969 e responsável por manter uma revista de produções científicas, sob o título Pesquisa Operacional, que é indexada no *International Abstracts in Operations Research* da IFORS e desde 2002 ao SciELO. Este Simpósio, ao longo dos anos, tem reunido a grande maioria dos profissionais da Pesquisa Operacional no Brasil, tanto nas universidades como nas empresas e em órgãos públicos diversos, sejam eles federais, estaduais ou municipais.

Já o Simpósio de Pesquisa Operacional & Logística da Marinha é organizado pela Marinha do Brasil, através do Centro de Análises de Sistemas Navais (CASNAV) e tem como objetivo promover a partilha de informações e experiências entre empresas privadas do segmento de Pesquisa Operacional e Logística, acadêmicos, pesquisadores, Forças Armadas, órgãos do governo e do setor produtivo de bens e serviços. Além de formar um

ótimo ambiente para identificar sinergias para a execução de projetos de desenvolvimento tecnológico, está nas intenções do SPOLM a formação de parcerias e captação de recursos humanos qualificados.

A *International Conference on Metaheuristics and Nature Inspired Computing* é uma conferência que promove um ótimo espaço para capacitação técnica e uma ampla integração entre usuários, pesquisadores, estudantes, prestadores de serviços e demais profissionais da área de pesquisa operacional, constituindo um ótimo ambiente para aprendizado. Possui sessões da pesquisa onde funcionam apresentações e também integra tutoriais, oficinas e uma formação profissional em metaheurísticas e computação evolutiva.

Finalmente, a *EURO Mini Conference XXVIII on Variable Neighbourhood Search* é uma conferência específica para trabalhos voltados para o uso da técnica *Variable Neighborhood Search* – VNS. É um evento de expressão por reunir pesquisadores de toda a comunidade científica que usam tal técnica, incluindo seus proponentes, Nenad Mladenovic e Pierre Hansen.

Referências

[Anderberg, 2007] Anderberg, M. R. (2007). *Cluster analysis for applications*. Monographs and Textbooks on Probability and Mathematical Statistics. Academic Press, Inc., New York.

[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 & 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.

[Cruz *et al.*, 2012a] Cruz, R. C., Silva, T. C. B., Souza, M. J. F., Coelho, V. N., Martins, A. X., and Mine, M. T. (2012a). A heuristic algorithm for solving the vehicle routing problem with simultaneous pickup and delivery. In *International Conference on Metaheuristics and Nature Inspired Computing – META'12*, Port El-Kantaoui, Tunisia.

[Cruz *et al.*, 2012b] Cruz, R. C., Silva, T. C. B., Souza, M. J. F., Coelho, V. N., Mine, M. T., and Martins, A. X. (2012b). Genvns-ts-cl-pr: A heuristic approach for solving the vehicle routing problem with simultaneous pickup and delivery. In *EURO Mini Conference XXVIII on Variable Neighbourhood Search*, Belgrado.

[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. *J. Comp. Phys.*, pages 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, 1997] Glover, F. (1997). A template for scatter search and path relinking. *Lecture Notes in Computer Science*. J. K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers, (Eds.).

[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.

[Gonçalvez, 2010] Gonçalvez, F. A. C. A. (2010). Sequenciamento em uma máquina: otimização heurística via multiprocessamento paralelo. Dissertação de mestrado, Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, Belo Horizonte.

[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, 1986] Hansen, P. (1986). The steepest ascent mildest descent heuristic for combinatorial programming. *Congress on Numerical Methods in Combinatorial Optimization*.

[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 & 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.

[Mine, 2009] Mine, M. T. (2009). Um algoritmo heurístico híbrido para o problema de roteamento de veículos com coleta e entrega simultânea. Dissertação de mestrado, Programa de Pós-Graduação em Ciência da Computação, Universidade Federal Fluminense, Niterói.

[Mine *et al.*, 2010] Mine, M. T., Silva, M. S. A., Ochi, L. S., and Souza, M. J. F. (2010). O problema de roteamento de veículos com coleta e entrega simultânea: uma abordagem via iterated local search e genius. In *Transporte em transformação XIV: trabalhos vencedores do prêmio CNT de Produção Acadêmica 2009*, pages 59–78. Editora Positiva, Brasília.

[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.

[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.

[Ribeiro *et al.*, 2002] Ribeiro, C. C., Uchoa, E., and Werneck, R. F. (2002). A hybrid grasp with perturbations for the steiner problem in graphs. *INFORMS Journal on Computing*, 14:228–246.

[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 4th International Conference on Principles and Practice of Constraint Programming*, pages 417–431, London, UK. Springer-Verlag.

[Silva *et al.*, 2011a] Silva, T. C. B., Cruz, R. C., Mine, M. T., Souza, M. J. F., and Santibanez, E. R. (2011a). Genils-ts: um algoritmo heurístico híbrido para resolução do problema de roteamento de veículos com coleta e entrega simultânea. *XLIII Simpósio Brasileiro de Pesquisa Operacional*.

[Silva *et al.*, 2011b] Silva, T. C. B., Cruz, R. C., Mine, M. T., Souza, M. J. F., and Santibanez, E. R. (2011b). Um novo algoritmo heurístico para resolução do problema de roteamento de veículos com coleta e entrega simultânea. *Simpósio de Pesquisa Operacional & Logística da Marinha*.

[Silva *et al.*, 2012a] Silva, T. C. B., Cruz, R. C., Souza, M. J. F., and Martins, A. X. (2012a). Genils-ts-cl: Um algoritmo heurístico para resolução do problema de roteamento de veículos com coleta e entrega simultânea. *Simpósio de Pesquisa Operacional & Logística da Marinha*.

[Silva *et al.*, 2012b] Silva, T. C. B., Cruz, R. C., Souza, M. J. F., Martins, A. X., Coelho, V. N., and Mine, M. T. (2012b). Genils-ts-cl-pr: Um algoritmo heurístico para resolução do problema de roteamento de veículos com coleta e entrega simultânea. *XLIV Simpósio Brasileiro de Pesquisa Operacional*.

[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 *Proceeding 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. Dissertação de mestrado, Programa de Pós-Graduação em Ciência da Computação, 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. Disponível em http://www.ic.uff.br/PosGraduacao/RelTecnicos/401.pdf.

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

[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.

[Zachariadis *et al.*, 2010] Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T. (2010). An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. *European Journal of Operational Research*, 202:401–411.