Programa de Pós-Graduação em Ciência da Computação
Departamento de Ciência da Computação
Instituto de Ciências Exatas e Biológicas
Pró-Reitoria de Pesquisa e Pós-Graduação
Universidade Federal de Ouro Preto

GENILS-TS-CL-PR: Um algoritmo heurístico para resolução do Problema de Roteamento de Veículos com Coleta e Entrega Simultânea

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Ciência da Computação, como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação.

Aluno: Thaís Cotta Barbosa da Silva

Orientador: Marcone Jamilson Freitas Souza

Ouro Preto - MG
Fevereiro de 2012

Um novo algoritmo heurístico para resolução do problema de roteamento de veículos com coleta e entrega simultânea

Thaís Cotta Barbosa da Silva

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

Aprovada por:

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

Prof. Alexandre Xavier Martins, D.Sc. / UFOP

Prof. Haroldo Gambini Santos , D.Sc. / UFOP

Prof. Maurício Cardoso de Souza, D.Sc. / UFMG

Ouro Preto, 29 de fevereiro de 2012.

Dedico este trabalho a meus pais Jailson e Mônica,
e ao meu noivo Jackson por todo apoio e carinho.

Agradecimentos

Aos meus tão queridos pais, Mônica e Jailson, e a minha irmã Letícia, pelo incentivo, carinho, força e por estarem sempre presentes em toda minha vida.

Ao meu noivo Jackson, por todos os gestos de amor e carinho, pela compreensão, pela motivação, e por estar sempre ao meu lado, me ajudando a crescer e superar os obstáculos.

À toda minha família, especialmente à minha avó Terezinha, pelo exemplo e dedicação, à minha prima-irmã Nathália, e às minhas tias Ana, Brígida e Tereza, pelo carinho e força.

Ao professor Marcone, pela preciosa orientação, pelo esforço e comprometimento na realização deste trabalho, além de seus incentivos e ensinamentos que carregarei para o resto da vida.

Ao Raphael, por suas contribuições, e amizade.

Aos meus amigos Alan (Akinase), André (Kurumin), Larissa, Matheus e Samuel, pela ajuda, pelas longas horas de estudo, amizade, carinho e força, fundamentais para que eu conseguisse chegar até aqui.

À minha amiga Júlia, pelo companheirismo e amizade, ajudando a distância de casa parecer menos dolorosa.

À Deus por ter colocado estas pessoas tão especiais na minha vida, que se transformaram em verdadeiros anjos, me guiando e protegendo por toda esta caminhada.

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

A todos o meu muito obrigada!

Resumo

Este trabalho tem seu foco no Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Dada a dificuldade de solução deste problema, é proposto um algoritmo heurístico nomeado GENILS-TS-CL-PR, que combina os procedimentos heurísticos Inserção Mais Barata, Inserção Mais Barata Múltiplas Rotas, GENIUS, *Iterated Local Search* (ILS), *Variable Neighborhood Descent* (VND), Busca Tabu (TS, do inglês *Tabu Search*) e Reconexão por Caminhos (PR, do inglês *Path Relinking*). Os três primeiros procedimentos visam a obtenção de uma 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. As buscas locais fazem uso de uma Lista de Candidatos de forma a evitar o uso de movimentos desnecessários e, assim, reduzir o espaço de busca. O algoritmo proposto foi testado em problemas-teste disponíveis na literatura e se mostrou capaz de gerar soluções de alta qualidade, sendo o algoritmo sequencial mais eficiente da literatura com relação à capacidade de encontrar melhores soluções e com pouca 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.

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 de (Mine et al., 2010)
GENILS-TS	:	Algoritmo GENILS com inserção do módulo de Busca Tabu
GENILS-TS-CL	:	Algoritmo GENILS-TS usando Lista de Candidatos
GENILS-TS-CL-PR	:	Algoritmo GENILS-TS-CL com módulo de Reconexão por Caminhos
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 Tabelas

Lista de Figuras

Capítulo 1

Introdução

Graças à crescente competitividade das empresas mundiais, estas vêm investindo cada vez mais em novos sistemas de informação para apoiar suas decisões, criando impactos positivos na gestão empresarial. A logística é uma das áreas que vem usufruindo desses sistemas não só para reduzir seus custos, mas também para melhorar a satisfação do cliente, que é um elemento fundamental no mercado atual, dando grande importância à disponibilidade do produto, sua eficiência e agilidade na entrega, dentre outros fatores.

Um dos principais problemas de grande aplicabilidade na logística é o Problema de Roteamento de Veículos (PRV), do inglês *Vehicle Routing Problem* (VRP). O PRV, proposto por (Dantzig e 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.

Uma importante variação do PRV, objeto de estudo deste trabalho, é o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES), ou VRPSPD, do inglês *Vehicle Routing Problem with Simultaneous Pickup and Delivery*. Proposto por (Min, 1989), esse 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 coletados, sendo que ambas as operações devem ser realizadas simultâneamente.

O PRV na sua forma original resolve, principalmente, problemas de Logística de Distribuição, preocupando-se apenas em gerenciar a distribuição de produtos, materiais ou pessoas. Já o PRVCES está presente na área conhecida como Logística Reversa, que pode ser definida na sua forma mais ampla como o gerenciamento das operações de distribuição e coleta de materiais, para reciclagem ou descarte especializado. Empresas de reciclagem, tratamento de resíduos, logística postal e distribuição de bebidas são bons exemplos da utilização da Logística Reversa na indústria.

O PRVCES pertence à classe NP-difícil, uma vez que ele pode ser reduzido ao PRV clássico quando nenhum cliente necessita de serviço de coleta. Sendo assim, as principais abordagens presentes na literatura se baseiam em heurísticas como: o algoritmo ILS-RVND de Subramanian et al. (2010), o algoritmo evolutivo de Zachariadis et al. (2010) e o algoritmo GENILS de Mine et al. (2010). O primeiro desses

algoritmos é paralelo, enquando os outros dois são sequenciais. Dentre os sequenciais, o GENILS é o de melhor desempenho. O GENILS combina os procedimentos heurísticos Inserção Mais Barata Rota a Rota, Inserção Mais Barata com Múltiplas Rotas, GENIUS (Gendreau et al., 1992), *Iterated Local Search* – ILS (Stützle e Hoos, 1999) e *Variable Neighborhood Descent* – VND (Mladenović e Hansen, 1997). Os três primeiros são usados para gerar uma solução inicial, enquanto o VND é usado como busca local do ILS.

Neste trabalho é expandido e aprimorado o algoritmo GENILS de (Mine et al., 2010). O algoritmo proposto, denominado GENILS-TS-CL-PR, incorpora ao GENILS três mecanismos de busca: A Busca Tabu como alternativa de procedimento de busca local do ILS após certo número de iterações sem melhora do VND; uma Lista de Candidatos para evitar a avaliação de soluções não promissoras e a Reconexão por Caminhos aplicada a cada ótimo local gerado.

1.1 Objetivos

Os objetivos gerais e específicos estão assim organizados, conforme descrição abaixo.

1.1.1 Objetivo Geral

- Desenvolver um algoritmo eficiente, baseado em metaheurísticas, para resolver o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea.

1.1.2 Objetivos Específicos

- Pesquisar e estudar procedimentos de otimização, em particular aqueles voltados para resolver o problema sob análise;
- Aperfeiçoar o algoritmo GENILS proposto em (Mine et al., 2010), de forma a melhorar seu desempenho em termos da capacidade de gerar soluções de qualidade com pouca variabilidade;
- Avaliar o desempenho do algoritmo desenvolvido frente a outros da literatura.

1.2 Motivação

Recursos computacionais têm se transformado cada vez mais em aliados de áreas como a Logística. O apoio no gerenciamento de seus recursos por meio de novas tecnologias podem trazer grandes benefícios na satisfação de clientes e na redução de gastos. Trabalhos como o de (Toth e Vigo, 2002) afirmam que o uso de métodos computacionais em processos de distribuição frequentemente resulta em economia da ordem de 5% a 20% nos custos de transporte. Assim, (Golden et al., 2002) descrevem vários estudos de caso nos quais a aplicação de algoritmos ao PRV tem conduzido a reduções de custo substanciais.

No caso do PRVCES, sua aplicação é mais comum na Logística Reversa, onde além da Logística de Distribuição, também há a preocupação com a coleta. Com

o aumento da preocupação com o meio ambiente, o uso sustentável de matérias-primas vem sendo valorizado cada vez mais pelas empresas. Sendo assim, a Logística Reversa vem sendo utilizada por muitas empresas para gerenciar a coleta de partes do seu produto para a reciclagem ou tratamento de resíduos.

Porém, a aplicabilidade do PRVCES não é o único motivador para o estudo deste problema. Sendo o PRVCES pertencente à classe de problemas NP-difíceis, é também um grande desafio resolvê-lo de forma eficiente, dada a dificuldade de obter a melhor solução, no caso geral, em tempos aceitáveis. Em vista disso, a abordagem heurística, como a proposta neste trabalho, é a mais comum para a solução dessa classe de problemas.

1.3 Estrutura do trabalho

O restante deste trabalho está organizado como segue. No capítulo 2, o PRVCES é caracterizado.

No capítulo 3 é feita uma revisão dos principais algoritmos desenvolvidos para resolução do problema, bem como das técnicas heurísticas utilizadas ao longo deste trabalho.

No capítulo 4 é apresentada a metodologia utilizada para resolver o PRVCES, enquanto no capítulo 5 são apresentados e discutidos os resultados alcançados pelo algoritmo desenvolvido.

Finalmente, no capítulo 6 são listadas as principais conclusões, bem como apontados os possíveis aperfeiçoamentos para o algoritmo desenvolvido.

Capítulo 2

Caracterização do Problema

O Problema de Roteamento de Veículos (PRV), do inglês *Vehicle Routing Problem* (VRP), foi proposto por (Dantzig e Ramser, 1959), e 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, obter um conjunto de rotas a serem percorridas pelos veículos cujo custo seja mínimo e atenda as seguintes restrições:

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

Graças a sua grande aplicabilidade, diversas variações do PRV vêm sendo propostas com o intuito de atender uma maior gama de problemas reais. As variações se dão pela inserção de novas restrições ao PRV, fazendo com que além da necessidade da demanda, outras características sejam associadas aos clientes. Dentre as principais características estão:

- Tipo da frota: homogênea, quando os veículos possuem as mesmas características ou heterogênea, caso contrário;
- Tamanho da frota: um veículo ou múltiplos;
- Tempo: tempo de serviço em cada cliente e janelas de tempo;
- Natureza da demanda: 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 depósito ou vários.

Vários trabalhos da literatura vêm sendo realizados com o intuito de encontrar soluções para as variações apresentadas.

Trabalhos como os de Belfiore e Yoshizaki (2006) e Choi e Tcha (2007) tratam a variante do PRV em que se considera a frota heterogênea. Neste problema os veículos possuem diferentes capacidades de carga. Assim, as construções das rotas devem levar em conta esse fato, já que a limitação de capacidade do veículo deve ser sempre respeitada.

Outra variante do PRV muito estudada é o Problema de Roteamento de Veículos com Janela de Tempo (VRPTW, do inglês *Vehicle Routing Problem With Time Windows*). O VPRTW inclui ao PRV a restrição de um intervalo de tempo para o cliente ser atendido, e ainda um intervalo para saída e retorno ao depósito. Trabalhos como o de Cordeau et al. (2001) estudam o problema em questão.

O presente trabalho tem o seu foco na variação denominada Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES), ou *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD). 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.

A Figura 2.1 ilustra um exemplo de uma solução para um PRVCES envolvendo 19 clientes e veículos de capacidade $Q = 150$ unidades.

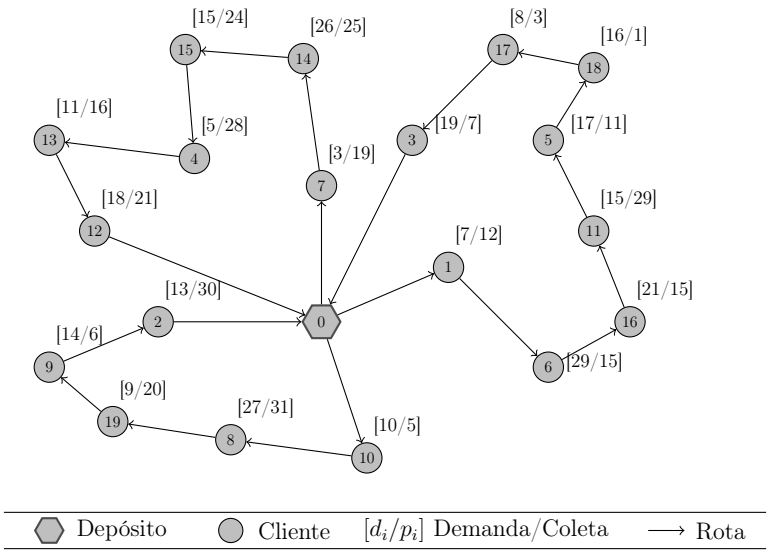

Figura 2.1: Exemplo do PRVCES.

Na rota do veículo do canto superior esquerdo da Figura 2.1, o veículo parte do depósito, passa pelo cliente 7 para entregar 3 unidades do produto e coletar 19 unidades. A seguir, se direciona para os clientes 14, 15, 4, 13, e 12, nesta ordem, entregando e coletando produtos, e retorna ao depósito.

Capítulo 3

Revisão Bibliográfica

3.1 Introdução

Para resolver um problema de planejamento de distribuição de materiais para bibliotecas públicas, Min (1989) propôs uma variação do PRV denominada Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Desde então, o PRVCES vem sendo estudado e resolvido por diversas técnicas que tentam aprimorar e melhorar a qualidade das soluções desse problema. Devido à sua grande complexidade computacional, já que o PRVCES pertence à classe NP-Difícil, diversas abordagens heurísticas vêm sendo propostas. Alguns dos principais algoritmos heurísticos são brevemente comentados a seguir.

Para resolver o PRVCES, Min (1989) apresentou um método que 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 da Inserção Mais Barata (Golden et al., 1984) foi desenvolvido 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 e 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 e 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 e Brandão (2005) por uma técnica híbrida que combina as metaheurísticas Busca Tabu (Glover e Laguna, 1997) e *Variable Neighborhood Descent* - VND (Hansen e 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 e 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 e 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 e 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 e 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.

Uma metodologia reativa da metaheurística Busca Tabu foi utilizada por Wassan et al. (2007) 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 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 metaheurísticas Busca Tabu e *Guided Local Search* (Voudouris e Tsang, 1996). Posteriormente, os mesmos autores propuseram um algoritmo evolucionário em 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, para resolver o PRVCES. Esse 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 de Zachariadis et al. (2010) obteve em 54 casos e o de Subramanian et al. (2008) em 51.

Subramanian et al. (2010) apresentaram um algoritmo paralelo para resolver o PRVCES. 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 método 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.

Em vista da competitividade do algoritmo GENILS, de Mine et al. (2010), ele é aperfeiçoado no presente trabalho com a introdução de um módulo de Busca Tabu substituindo a fase de refinamento do ILS a partir de um determinado número de iterações sem melhora no processo de busca.

3.2 Formulação Matemática

A formulação matemática apresentada no trabalho de Subramanian et al. (2011) 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 (3.1) a (3.20).

$$\text{(P2)} \qquad \text{Minimizar} \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{3.1}$$

$$\text{s.a:} \qquad \sum_{j\in N^*} (d_{ji} - d_{ij}) = 2\,D_i \qquad (i \in N) \tag{3.2}$$

$$\sum_{j\in N} d_{0j} = \sum_{i\in N} D_i \tag{3.3}$$

$$\sum_{j\in N} d_{j0} = K\,Q - \sum_{i\in N} D_i \tag{3.4}$$

$$\sum_{j\in N^*} (p_{ij} - p_{ji}) = 2\,P_i \qquad (i \in N) \tag{3.5}$$

$$\sum_{j\in N} p_{j\bar{0}} = \sum_{i\in N} P_i \tag{3.6}$$

$$\sum_{j\in N} p_{\bar{0}j} = K\,Q - \sum_{i\in N} P_i \tag{3.7}$$

$$\sum_{j\in N^*} (w_{ji} - w_{ij}) = 2\,(D_i - P_i) \qquad (i \in N) \tag{3.8}$$

$$w_{0j} = d_{0j} \qquad (j \in N) \tag{3.9}$$

$$w_{j0} = d_{j0} \qquad (j \in N) \tag{3.10}$$

$$w_{j\bar{0}} = p_{j\bar{0}} \qquad (j \in N) \tag{3.11}$$

$$w_{\bar{0}j} = p_{\bar{0}j} \qquad (j \in N) \tag{3.12}$$

$$d_{ij} + d_{ji} = Q\,x_{ij} \qquad ((i,j) \in A) \tag{3.13}$$

$$p_{ij} + p_{ji} = Q\,x_{ij} \qquad ((i,j) \in A) \tag{3.14}$$

$$w_{ij} + w_{ji} = Q\,x_{ij} \qquad ((i,j) \in A) \tag{3.15}$$

$$\sum_{i\in N^*, i<k} x_{ik} + \sum_{j\in N^*, j>k} x_{kj} = 2 \qquad (k \in N) \tag{3.16}$$

$$\sum_{j\in N} x_{0j} \le K \tag{3.17}$$

$$\sum_{j\in N} x_{j\bar{0}} \le K \tag{3.18}$$

$$x_{ij} \in \{0,1\} \qquad ((i,j) \in A) \tag{3.19}$$

$$p_{ij}, d_{ij}, w_{ij} \ge 0 \qquad ((i,j) \in A) \tag{3.20}$$

Neste modelo, (3.1) representa a função objetivo, que consiste em minimizar o total das distâncias percorridas por todos os veículos; as restrições (3.2) garantem que todas as demandas por entrega sejam satisfeitas; a restrição (3.3) estabelece que a carga do veículo que sai do depósito seja igual ao somatório das demandas (entrega) dos clientes; a restrição (3.4) estabelece que a carga dos veículos que chegam no depósito seja igual à carga residual dos veículos quando saem do depósito; as restrições (3.5) a (3.7) são análogas às restrições (3.2) a (3.4), porém relativas a

demanda por coleta; as restrições (3.8) asseguram que as entregas e coletas sejam realizadas simultaneamente; as restrições (3.9) a (3.15) estabelecem que a capacidade do veículo não seja excedida; as restrições (3.16) definem que o número de arcos incidentes em cada cliente seja igual a dois; as restrições (3.17) e (3.18) limitam a quantidade máxima de veículos utilizados; as restrições (3.20) indicam que as variáveis p_{ij}, d_{ij} e w_{ij} são contínuas e não-negativas e as restrições (3.19) definem as variáveis x_{ij} como binárias.

3.3 Heurísticas

Heurísticas são métodos que exploram o espaço de soluções de forma "inteligente", buscando encontrar soluções de boa qualidade em tempos aceitáveis. Porém, ao contrário de métodos exatos, onde a otimalidade é garantida, heurísticas não são capazes de garantirem que o ótimo local encontrado é também o global. Por outro lado, elas são capazes de produzir, em geral, soluções de qualidade em tempos computacionais reduzidos.

As heurísticas podem ser construtivas ou de refinamento. Heurísticas construtivas são utilizadas para gerar uma solução inicial, onde através de uma função de avaliação, elementos que compõe a solução vão sendo adicionados, até atender todas as restrições do problema, gerando uma solução viável.

Heurísticas de refinamento partem de uma solução inicial, geralmente obtida através das heurísticas construtivas, e realizam uma busca local no espaço de soluções vizinhas, com o intuito de encontrar ótimos locais. 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.

A Subseção 3.3.1 descreve a heurística Descida em Vizinhança Variável, do inglês *Variable Neighborhood Descent* (VND), e na Subseção 3.3.2 apresenta-se a heurística GENIUS.

3.3.1 VND

A Descida em Vizinhança Variável, do inglês *Variable Neighborhood Descent* - VND (Hansen e Mladenović, 2001) é uma heurística de refinamento que explora o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança.

Como pode ser observado no Algoritmo 1, o funcionamento do VND acontece como explicado a seguir. 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.

Algoritmo 1: Descida em Vizinhança Variável

Seja r o número de estruturas de vizinhança distintas;
$k \leftarrow 1$;
enquanto ($k \leq r$) **faça**
  Encontre o melhor vizinho $s' \in N^{(k)}(s)$;
  se ($f(s') < f(s)$) **então**
    $s \leftarrow s'$;
    $k \leftarrow 1$;
  senão
    $k \leftarrow k + 1$;
  fim se
fim enquanto
Retorne s;

3.3.2 GENIUS

A heurística GENIUS foi desenvolvida por (Gendreau et al., 1992) para resolver o Problema do Caixeiro Viajante (PCV), do inglês *Traveling Salesman Problem* (TSP). Essa heurística é composta de uma fase construtiva (GENI - *Generalized Insertion*) e a outra de refinamento (US - *Unstringing and Stringing*). A fase construtiva GENI baseia-se nos métodos de inserção, onde um vértice v não é inserido 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 2 apresenta o pseudocódigo básico da heurística GENIUS. Nesse algoritmo, a *FaseGENI* é a descrita na Seção 3.3.2.1 e a *FaseUS* está descrita na Subseção 3.3.2.2.

Algoritmo 2: GENIUS

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

3.3.2.1 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 3 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 3: 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';

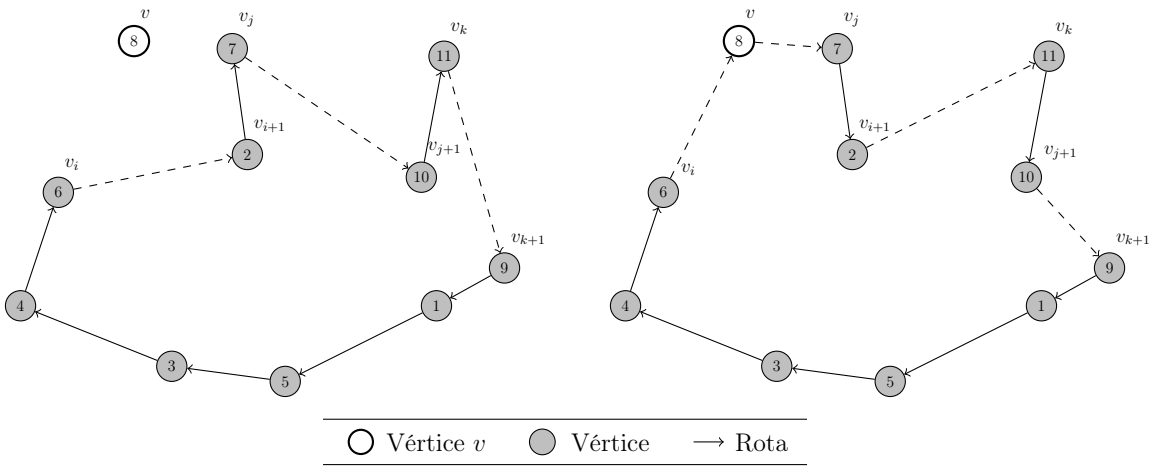

Figura 3.1: Exemplo da Inserção Tipo I da fase GENI.

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 4 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 4: 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';

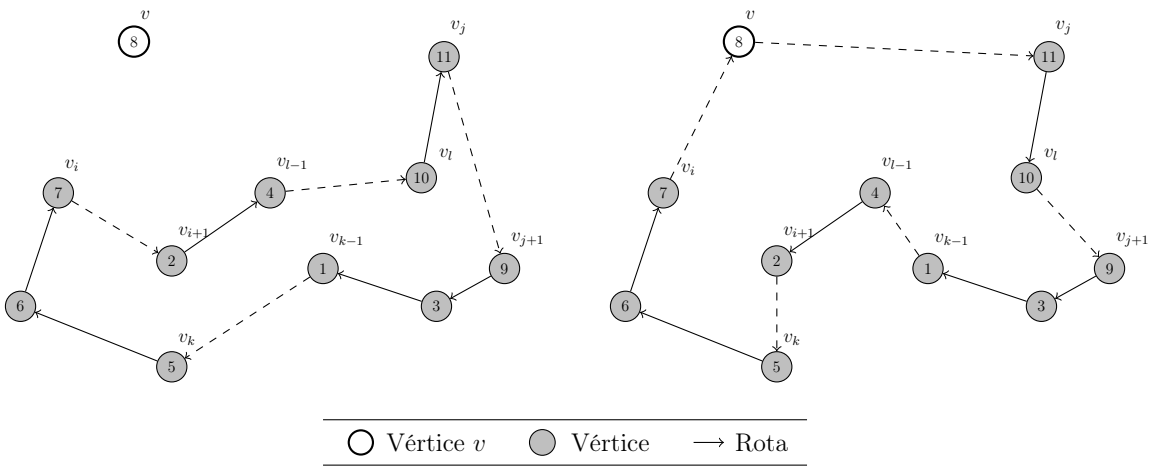

Figura 3.2: Exemplo da Inserção Tipo II da fase GENI.

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 5, o qual faz uso do procedimento de inserção mostrada no Algoritmo 6. Nesse último algoritmo,

considera-se que $f(s)$ é o custo (distância total) da solução s.

Algoritmo 5: 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 6 }
5: $V^- = V^- \setminus \{v\}$;
6: **fim enquanto**
7: Retorne s;

Algoritmo 6: Inserção GENI

1: Dado um vértice v e uma solução s:
2: $s^* \leftarrow s$;
3: **para** ($s' \in \{ s, \bar{s} \}$) **faça**
4: **para** ($v_i, v_j \in N_p(v)$) **faça**
5: **para** ($v_k \in N_p(v_{i+1})$) **faça**
6: $s'' \leftarrow InserçãoTipoI(s', v, v_i, v_j, v_k)$; { Algoritmo 3 }
7: **se** ($f(s'') < f(s^*)$) **então**
8: $s^* \leftarrow s''$;
9: **fim se**
10: **para** ($v_l \in N_p(v_{j+1})$) **faça**
11: $s'' \leftarrow InserçãoTipoII(s', v, v_i, v_j, v_k, v_l)$; { Algoritmo 4 }
12: **se** ($f(s'') < f(s^*)$) **então**
13: $s^* \leftarrow s''$;
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 e 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.

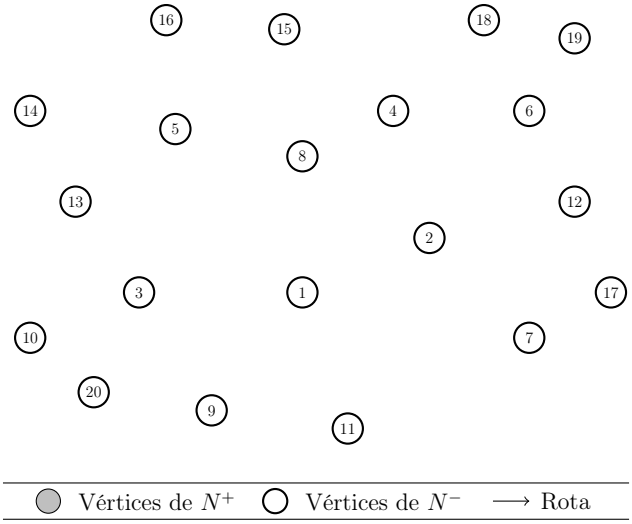

Figura 3.3: Exemplo de um problema do PCV, considerando 20 clientes.

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

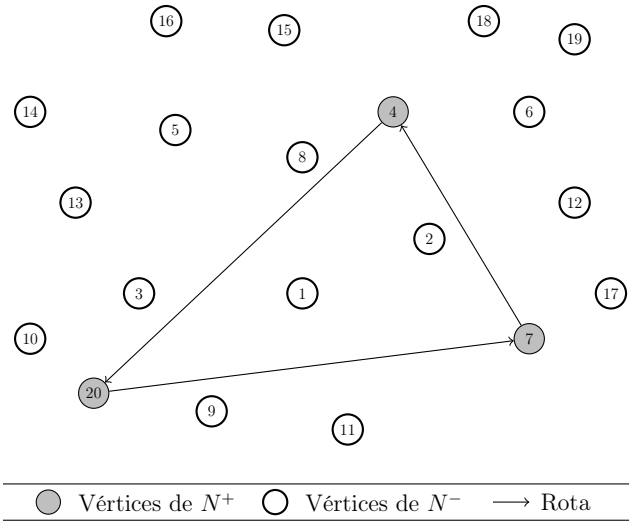

Figura 3.4: Subrota inicial do GENI contendo os vértices 7, 4 e 20.

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.

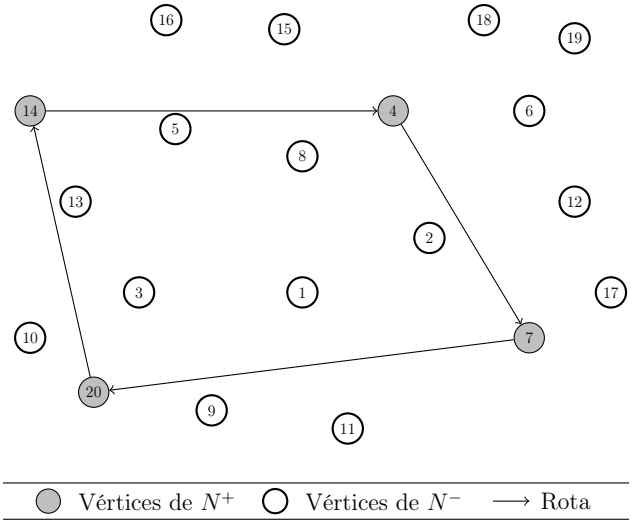

Figura 3.5: Inserção do vértice 14 com a heurística GENI.

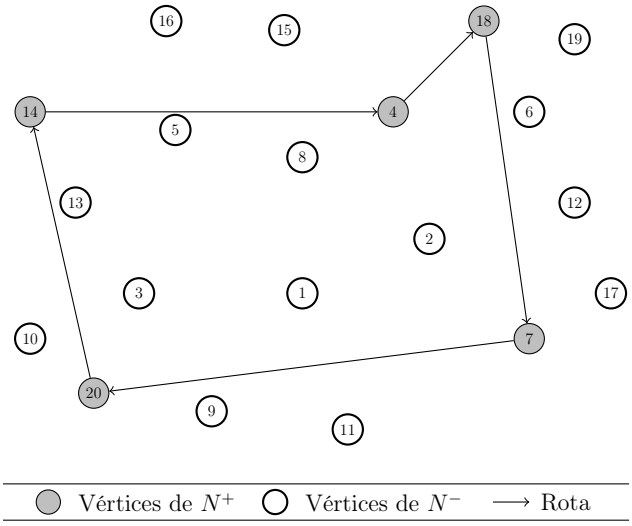

Figura 3.6: Inserção do vértice 18 com a heurística GENI.

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.

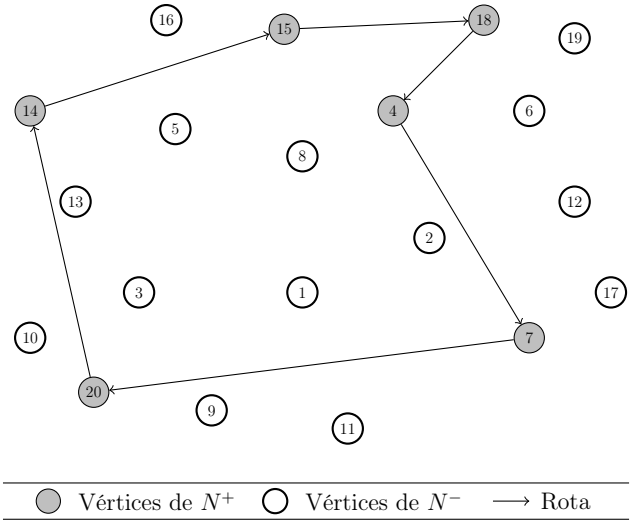

Figura 3.7: Inserção do vértice 15 com a heurística GENI.

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

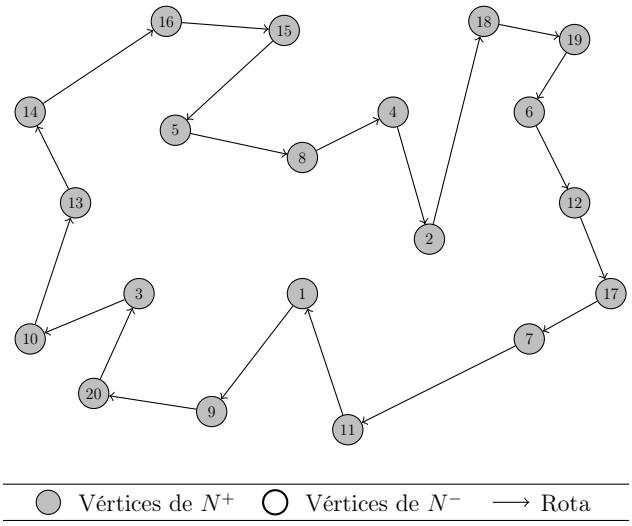

Figura 3.8: Solução inicial gerada pela heurística GENI.

3.3.2.2 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 7 e a Figura 3.9 mostra a exclusão do vértice $\bar{v}_i = 8$, considerando $v_l = 2$ e $v_j = 10$.

Algoritmo 7: 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';

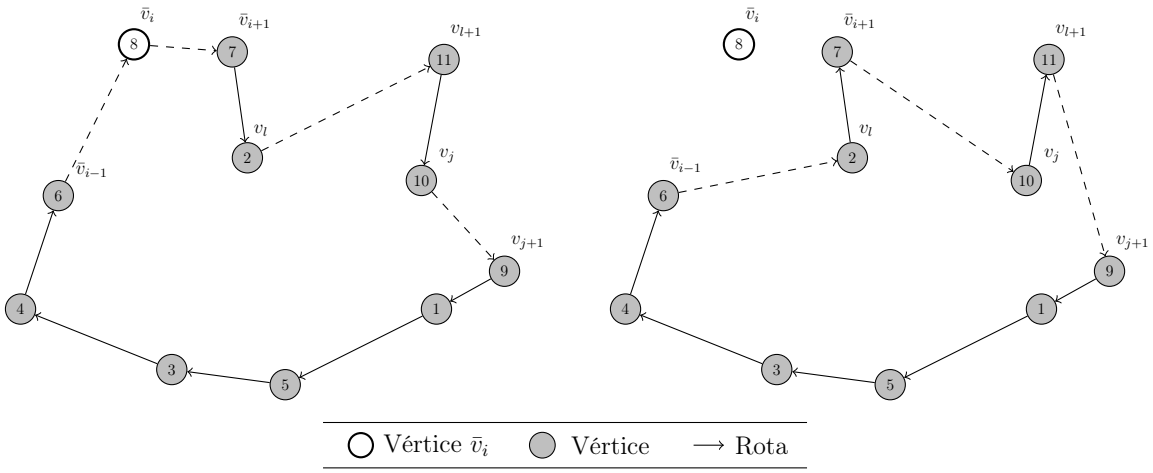

Figura 3.9: Exemplo da Remoção Tipo I da fase US.

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 8 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 8: 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';

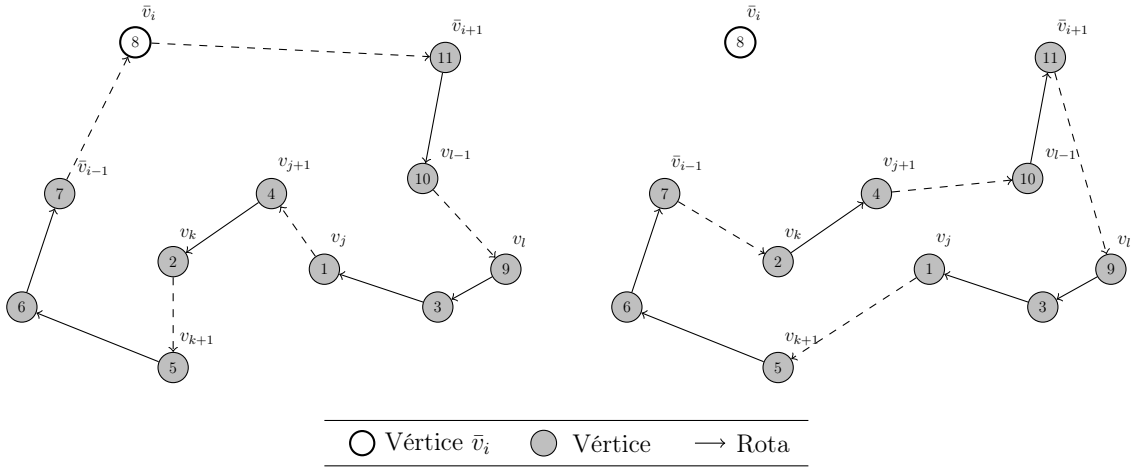

Figura 3.10: Exemplo da Remoção Tipo II da fase US.

O pseudocódigo da fase US é apresentada no Algoritmo 9. 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 na rota. Além disso, considere que $v(p_z, s)$ é o vértice que se encontra na posição p_z da solução s.

Algoritmo 9: Fase de refinamento US

Considere s como sendo uma solução inicial:
$s^* \leftarrow s$;
$p_z \leftarrow p_1$;
enquanto ($p_z \leq p_n$) **faça**
  $v \leftarrow v(p_z, s)$;
  $s' \leftarrow$ *RemoçãoUS(v,s)*; { Algoritmo 10 }
  $s'' \leftarrow$ *InserçãoGENI(v,s')*; { Algoritmo 6 }
  se ($f(s'') < f(s^*)$) **então**
    $s^* \leftarrow s''$;
    $p_z \leftarrow p_1$;
  senão
    $p_z \leftarrow p_{z+1}$;
  fim se
  $s \leftarrow s''$;
fim enquanto
$s \leftarrow s^*$;
Retorne s;

Algoritmo 10: Remoção US

1: Dado um vértice $\bar{v}_i$ e uma solução s:
2: $s^* \leftarrow s$;
3: **para** ($s' \in \{ s, \bar{s} \}$) **faça**
4: **para** ($v_k \in N_p(\bar{v}_{i-1})$) **faça**
5: **para** ($v_j \in N_p(v_{k+1})$) **faça**
6: $s'' \leftarrow RemoçãoTipoI(s', \bar{v}_i, v_j, v_k)$; { Algoritmo 7 }
7: **se** ($f(s'') < f(s^*)$) **então**
8: $s^* \leftarrow s''$;
9: **fim se**
10: **para** ($v_l \in N_p(\bar{v}_{i+1})$) **faça**
11: $s'' \leftarrow RemoçãoTipoII(s', \bar{v}_i, v_j, v_k, v_l)$; { Algoritmo 8 }
12: **se** ($f(s'') < f(s^*)$) **então**
13: $s^* \leftarrow s''$;
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.

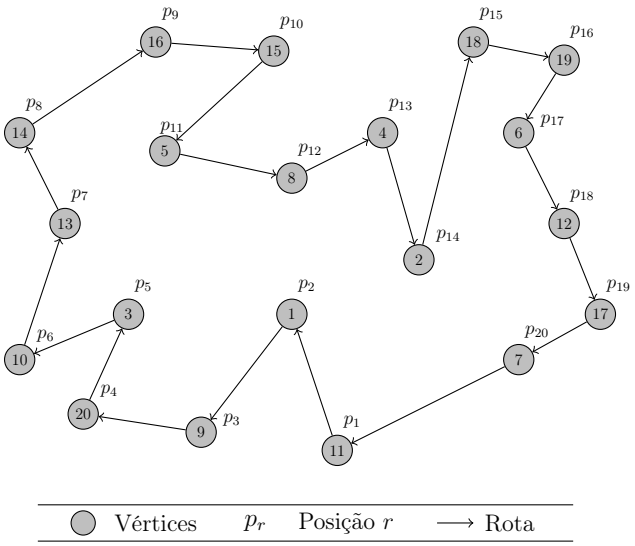

Figura 3.11: Exemplo da fase US.

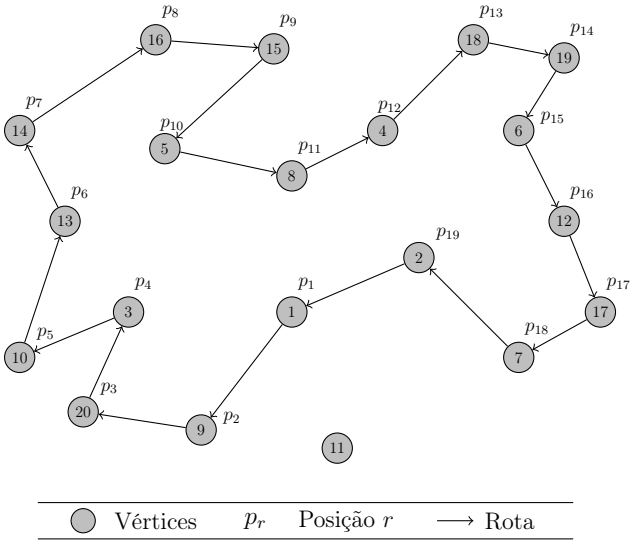

Figura 3.12: Remoção do vértice 11 localizado na posição p_1.

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 6 (*InserçãoGENI*).

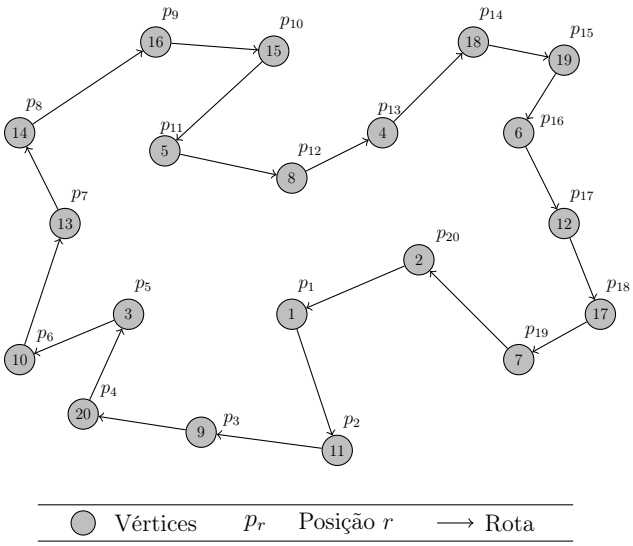

Figura 3.13: Reinserção do vértice 11 com o Algoritmo 6.

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.

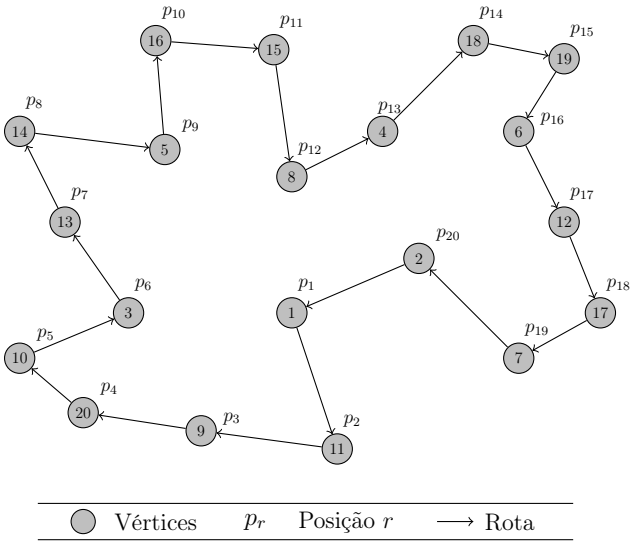

Figura 3.14: Solução gerada pela fase US e pela heurística GENIUS.

3.4 Metaheurística

Assim como as heurísticas, as metaheurísticas são métodos que buscam boas soluções em tempos computacionais aceitáveis. Porém, elas possuem uma característica que as diferem das heurísticas, que é a capacidade de fugir de ótimos locais para alcançar outras regiões no espaço de busca, tentando encontrar ótimos locais melhores, que podem ser ótimos globais.

Neste trabalho foram utilizadas as metaheurísticas *Iterated Local Search* (ILS) e Busca Tabu, as quais são descritas nas Subseções 3.4.2 e 3.4.1, respectivamente.

3.4.1 *Iterated Local Search*

O *Iterated Local Search* (ILS) (Stützle e Hoos, 1999) é uma metaheurística baseada no principio de que o espaço de soluções ótimas locais podem ser alcançados a partir da geração de perturbações em ótimos locais correntes e aplicação de buscas locais a essas soluções intermediárias produzidas.

Como pode ser observado no Algoritmo 11, o ILS parte de uma solução inicial s_0. Esta solução, por sua vez, é refinada através de uma busca local. Para escapar do ótimo local inicial s 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 torna-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.

Algoritmo 11: *Iterated Local Search*

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

3.4.2 Busca Tabu

A Busca Tabu (Glover e Laguna, 1997) é uma metaheurística que caminha no espaço de soluções, movendo-se de um vizinho para o outro, buscando sempre o melhor vizinho da solução corrente. Para auxiliar essa busca é utilizada uma estrutura de memória, chamada Lista Tabu, para armazenar características das soluções geradas, o que permite que o algoritmo não fique preso em um ótimo local.

A Lista Tabu proíbe que soluções com características tabus sejam geradas. Funcionando como uma fila de tamanho determinado, toda vez que um novo movimento tabu é adicionado à lista, o mais antigo sai. Sendo assim, o risco de ciclagem é reduzido, uma vez que é garantido, que pelo tempo em que o movimento permanece tabu, uma solução já visitada anteriormente não será gerada. Porém, a Lista Tabu também pode proibir movimentos para soluções que ainda não foram visitadas. Para minimizar os efeitos desse problema, existe uma função de aspiração, que é um mecanismo que retira, sob certas circunstâncias, o *status* tabu de um movimento.

Como forma de explorar o espaço de soluções de forma mais efetiva, é comum a BT utilizar estratégias conhecidas como intensificação e diversificação. A inten-

sificação é feita, geralmente em regiões consideradas promissoras, concentrando as buscas nessa região, analisando de forma mais eficaz aquele espaço de soluções. Já a diversificação, geralmente é utilizada quando o algoritmo já exauriu a busca naquela região, assim a idéia é diversificar a pesquisa para esplorar outras regiões.

Os parâmetros principais de controle do método de Busca Tabu são a cardinalidade $|T|$ da lista tabu, a função de aspiração A, a cardinalidade do conjunto V de soluções vizinhas testadas em cada iteração e $BTmax$, o número máximo de iterações sem melhora no valor da melhor solução.

Apresenta-se, no Algoritmo 12, o pseudocódigo de uma Busca Tabu básica para o caso de minimização. Neste procedimento, $f_{\min}$ é o valor mínimo conhecido da função f, informação esta que em alguns casos está disponível.

Algoritmo 12: BT $(f(.),N(.),A(.),|V|,f_{min},\ |T|,BTmax,\ s$

1: $s^* \leftarrow s$; {Melhor solução obtida até então}
2: $iter \leftarrow 0$; {Número de iterações}
3: $MelhorIter \leftarrow 0$;{Número de iterações sem melhora na solução}
4: $|T| \leftarrow 0$;{Lista Tabu }
5: **enquanto** ($f(s) \leq f_{\min} e iter - MelhorIter \leq iterMaxTS$) **faça**
6: $iter \leftarrow iter + 1$;
7: Seja $s' \leftarrow s \oplus m$ o 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 ($f(s') \leq A(f(s))$);
8: Atualize a lista tabu T;
9: $s \leftarrow s'$;
10: **se** ($f(s) < f(s^*)$) **então**
11: $s^* \leftarrow s$;
12: $MelhorIter \leftarrow Iter$;
13: **fim se**
14: Atualize a função de aspiração A;
15: **fim enquanto**
16: Retorne s^*;

3.5 Reconexão por Caminhos

A técnica Reconexão por Caminhos, do inglês *Path Relinking*, foi proposta por Glover (1996) 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

Algoritmo Proposto

Neste Capítulo é apresentado o algoritmo proposto para resolver o PRVCES. Na Seção 4.1 mostra-se como uma solução do problema é representada computacionalmente. Na Seção 4.2 apresenta-se o algoritmo proposto, na Seção 4.3 é dedicado a função utilizada para avaliar as soluções do algoritmo, as Seções 4.4 e4.5. apresenta respectivamente, os métodos utilizados para gerar a solução inicial e as estruturas de vizinhança utilizadas neste trabalho. Já nas Seções 4.8, 4.9, 4.10, e 4.11 apresentam o detalhamentos dos procedimentos contidos no ILS.

4.1 Representação de uma solução

A solução do PRVCES é representada computacionalmente por meio de um vetor de números inteiros. Os clientes são representados por números de 1 a n, e o depósito por 0. Sendo assim, como uma rota necessariamente começa e termina no depósito, o 0 é utilizado como separador de rotas, ou seja, a sequência de clientes compreendida entre os 0 representa as rotas da solução.

A Figura 4.1, que considera 19 clientes e 3 veículos disponíveis, representa uma possível solução do PRVCES.

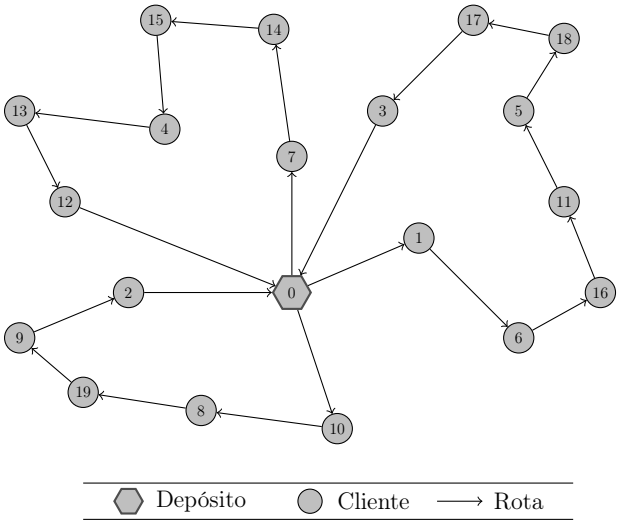

Figura 4.1: Exemplo de uma solução do PRVCES.

Para esta Figura, a solução seria representada por:

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

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.

4.2 Algoritmo GENILS-TS-LC-RC

O algoritmo proposto, nomeado GENILS-TS-LC-RC, é um aperfeiçoamento do algoritmo GENILS de Mine et al. (2010). Esse algoritmo combina as heurísticas ILS, VND, Busca Tabu (TS, do inglês *Tabu Search*), assim como os procedimentos heurísticos Inserção Mais Barata, Inserção Mais Barata Múltiplas Rotas e GENIUS e Reconexão de Caminhos. Seu pseudocódigo é apresentado pelo Algoritmo 14.

Algoritmo 14: *GENILS-TS-LC-RC*

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

Como pode ser observado no Algoritmo 14, os três métodos construtivos utilizados, os quais são descritos na Seção 4.4, produzem as soluções s^A, s^B e s^C, e cada uma delas é refinada por um procedimento VND (Seção 4.8). A melhor solução gerada se transforma na solução inicial s, sendo esta refinada pelo ILS. Para fugir de ótimos locais, se dirigindo para outras regiões do espaço de busca, foram utilizadas perturbações descritas na Seção 4.10. Já os métodos VND e Busca Tabu (descritos nas Seções 4.8) e 4.9, respectivamente) são utilizados como busca local, sendo a Busca Tabu acionada somente após certo número de iterações sem melhora, dado

pelo parâmetro *iterMaxSemMelhora*. A cada iteração do ILS também é aplicada a técnica Reconexão por Caminhos (descrita na Seção 4.11) na solução gerada pela busca local s''.

4.3 Função de Avaliação

Uma solução é avaliada pela função (4.1), a qual deve ser minimizada. 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 da Solução Inicial

Para gerar a solução inicial, foram utilizadas as heurísticas construtivas de Inserção Mais Barata Rota a Rota, Inserção Mais Barata com Múltiplas Rotas e uma adaptação da heurística GENIUS, descritas respectivamente nas Subseções 4.4.1, 4.4.2 e 4.4.3.

4.4.1 Inserção Mais Barata Rota a Rota

A Inserção Mais Barata Rota a Rota, também conhecido como IMB-1R, proposto por Dethloff (2001), consiste basicamente na construção de uma subrota inicial contendo um cliente aleatório, sendo os demais clientes inseridos a cada iteração respeitando-se as restrições do PRVCES. Para determinar qual cliente será incluido na rota, é utilizada a Equação (4.2), de forma que i e j correspondem a clientes já alocados a alguma rota e k representa um potencial cliente a ser inserido na rota.

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

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 distante, de forma a não adicioná-lo tardiamente à rota. Por esta expressão, verifica-se que quanto maior é a distância, maior é a bonificação. 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 4.2. Neste exemplo existem 19 clientes e uma frota ilimitada de veículos de capacidade $Q = 120$.

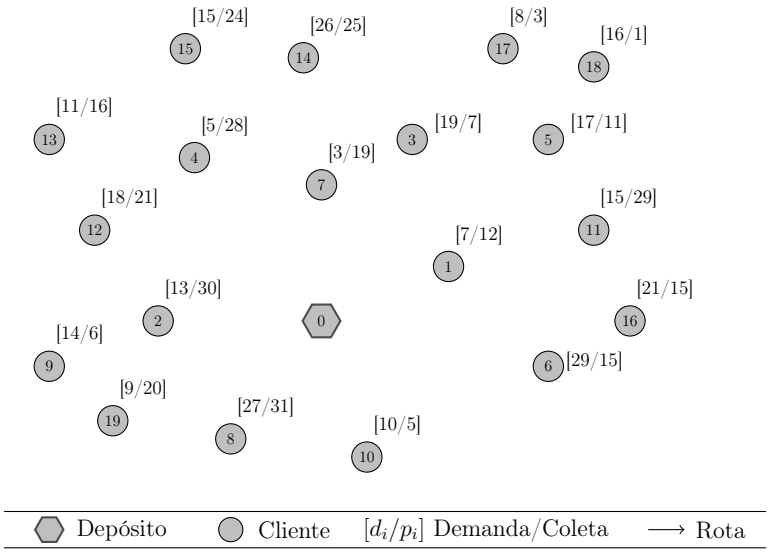

Figura 4.2: Exemplo de um problema envolvendo 19 clientes.

Inicialmente, deve-se gerar uma rota contendo um cliente escolhido aleatoriamente, no caso, o cliente 1 conforme apresentado na Figura 4.3.

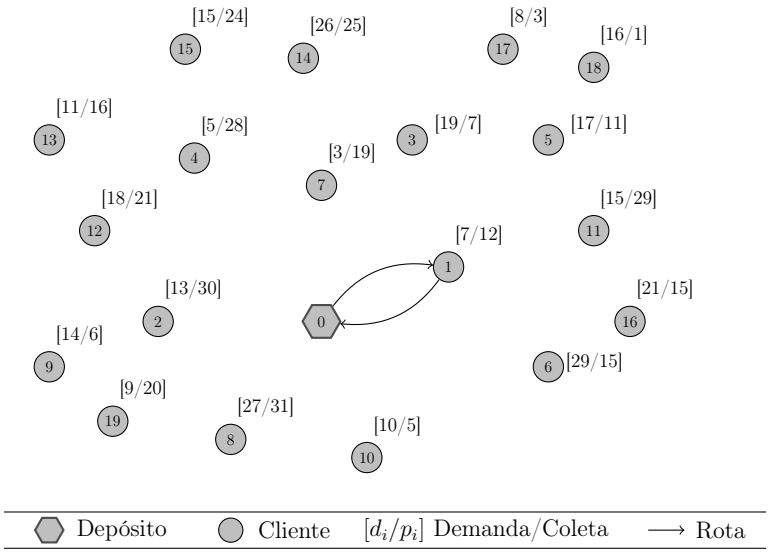

Figura 4.3: Construção de uma rota com o cliente 1.

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 4.4 mostra essa inserção.

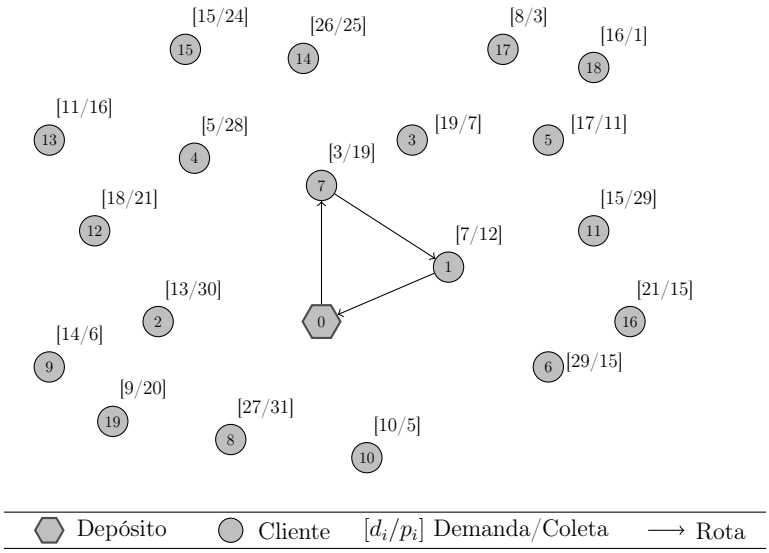

Figura 4.4: Inserção do cliente 7 na rota.

Continuando com a inserção dos clientes, chega-se à situação apresentada na Figura 4.5.

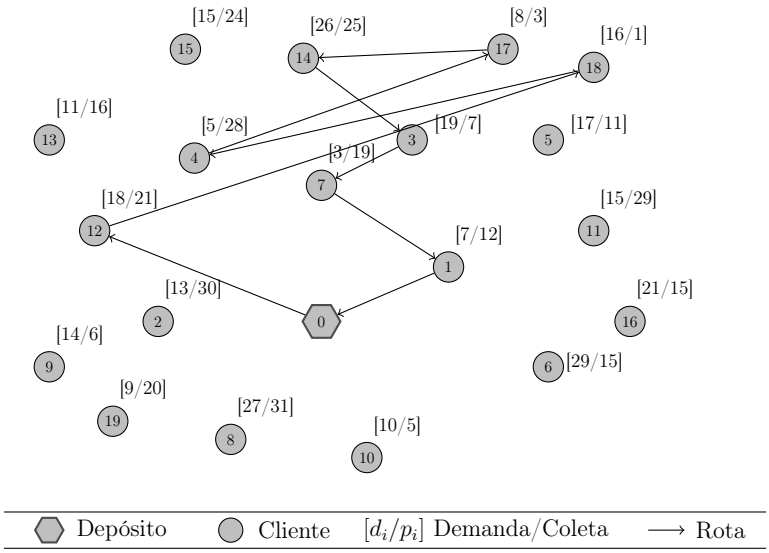

Figura 4.5: Construção completa de uma rota.

Na Figura 4.5 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 4.7 mostra a solução final obtida pelo método.

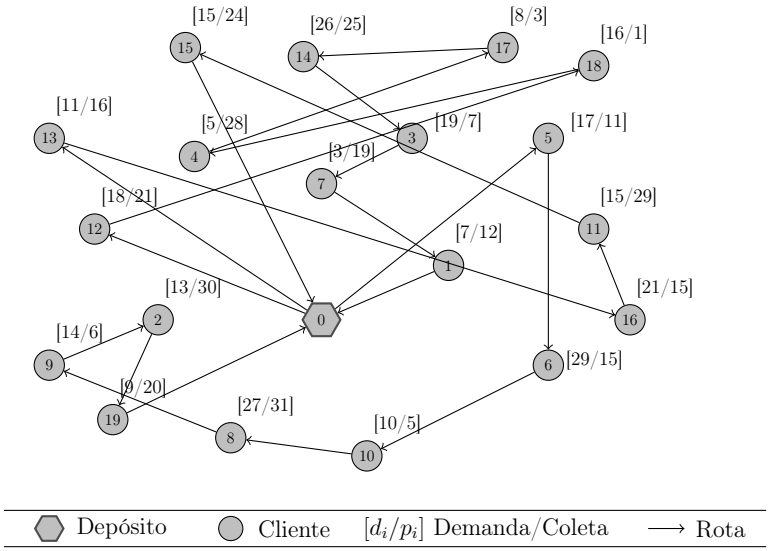

Figura 4.6: Solução gerada pela heurística de inserção mais barata rota a rota.

4.4.2 Inserção Mais Barata com 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.

Para inicializar o IMB-MR, é necessário informar o número inicial de rotas m.

O seu funcionamento ocorre de forma análoga ao algoritmo descrito na Subseção anterior. Porém ao invés de começar com apenas uma rota, m rotas são inicializadas simultaneamente, assim o IMB-MR sorteia aleatoriamente um cliente cada rota. A seguir, a equação (4.2) é utilizada para avaliar qual cliente deve ser inserido a cada rota. Esse procedimento é repetido, até que as rotas não comportem, por restrição de carga, a inserção de nenhum cliente. Se ao final desse procedimento ainda restarem clientes que não pertencem a nenhuma rota, uma nova rota é inicializada e construída como no algoritmo IMB-1R.

Para se obter o parâmetro m requerido pelo algoritmo, esse trabalho utilizou a solução gerada pelo Algoritmo IMB-1R, assim se a solução gerada obtiver três rotas, o valor será $m = 3$.

As figuras a seguir exemplificam o passo a passo do funcionamento do IMB-MR.

Considerando o exemplo da Figura 4.2, após a aplicação do método IMB-IR, obtém-se a solução com três rotas apresentada na Figura 4.7.

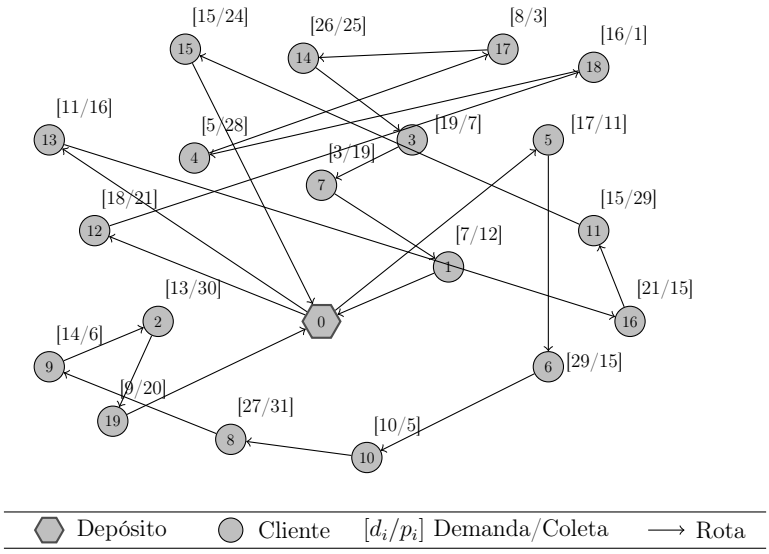

Figura 4.7: Solução gerada pela heurística de inserção mais barata rota a rota.

Em seguida, deve-se construir m rotas contendo um cliente aleatório, neste caso $m = 3$. A Figura 4.8 mostra a etapa inicial do IMB-MR, com os clientes 8, 11 e 13 sorteados. Neste exemplo, considera-se que o valor de γ seja igual a 0.10.

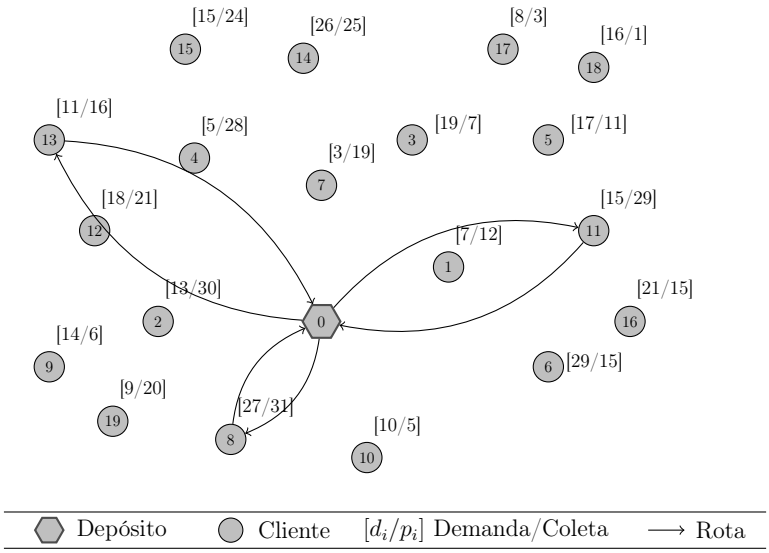

Figura 4.8: Etapa inicial do método, considerando três rotas.

O próximo passo consiste em calcular o custo de inserção de cada cliente, que ainda não esteja presente na solução, em todas as posições de todas as rotas. Neste exemplo, a inserção de menor custo é a do cliente 12 entre o depósito e o cliente 13. A Figura 4.9 ilustra essa inserção.

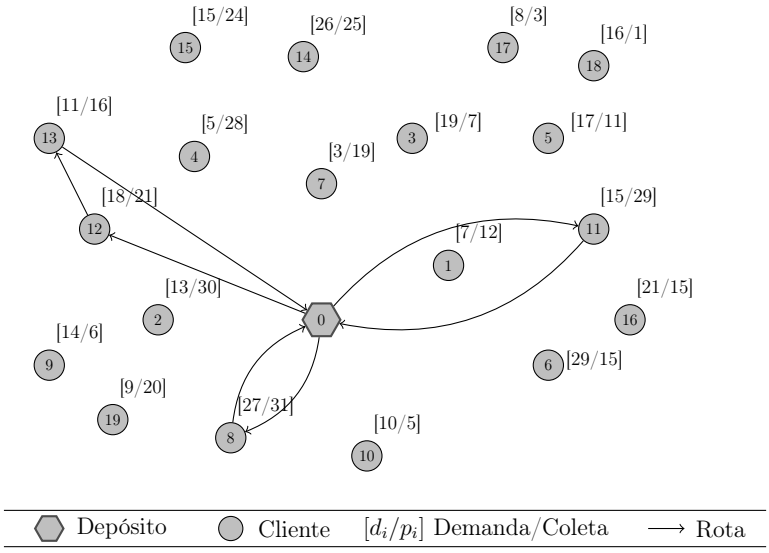

Figura 4.9: Inserção do cliente 12 entre o depósito e o cliente 13.

Após a inserção do cliente 12, deve-se novamente calcular os custos de inserção de cada cliente. Na Figura 4.10 observa-se a inserção do cliente 1 entre o depósito e o cliente 11.

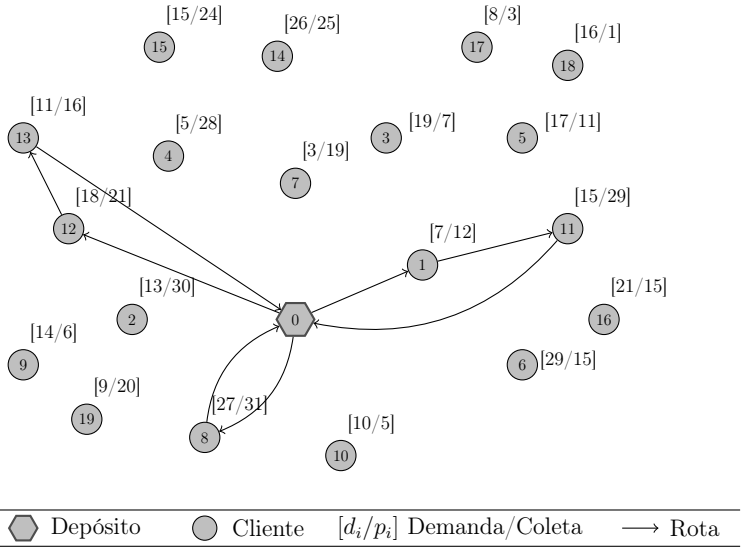

Figura 4.10: Inserção do cliente 1 entre o depósito e o cliente 11.

Continuando com esse procedimento, obtém-se a solução apresentada na Figura 4.11.

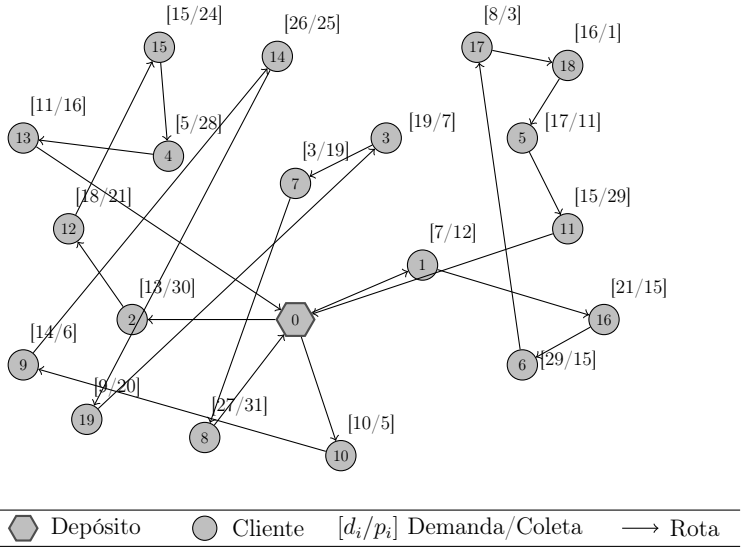

Figura 4.11: Solução gerada pelo método de inserção mais barata com múltiplas rotas.

Como essa heurística pode gerar uma solução incompleta e, portanto, inviável. Isso é devido à dependência do parâmetro m, 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 4.12, no qual foram utilizadas apenas duas rotas, considerando o problema apresentado na Figura 4.2.

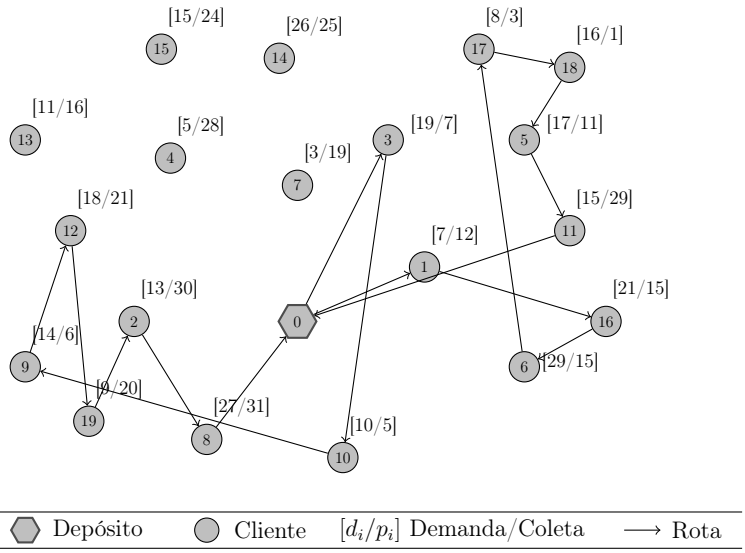

Figura 4.12: Exemplo da geração de uma solução incompleta.

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 4.13 mostra a solução viável gerada pela aplicação desta adaptação à solução incompleta da Figura 4.12.

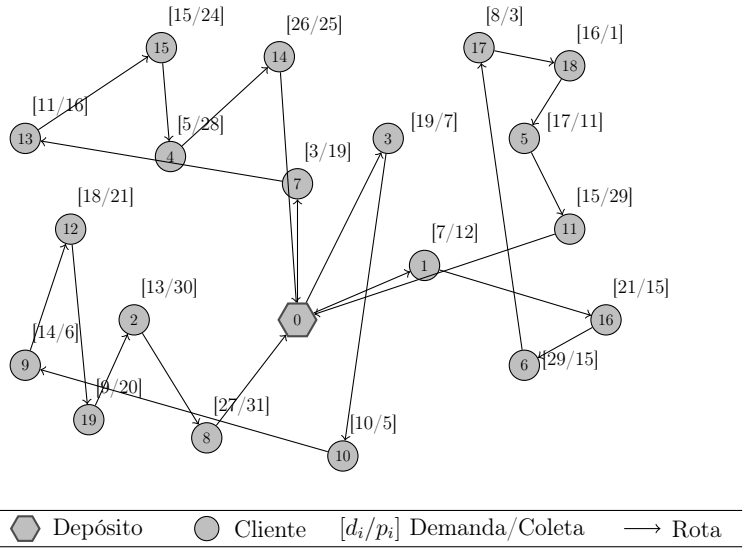

Figura 4.13: Solução gerada pela adaptação do método IMB-MR.

4.4.3 Procedimento construtivo VRGENIUS

O terceiro procedimento utilizado para construir uma solução inicial para o PRV-CES é o VRGENIUS. Este procedimento, desenvolvido por Mine et al. (2010), é

uma adaptação da heurística GENIUS, descrita na Subseção 3.3.2. As principais adaptações desses autores no algoritmo GENIUS foram a inserção da restrição de carga do veículo e a possibilidade de formação de múltiplas rotas. As Subseções 4.4.3.1 e 4.4.3.2 descrevem, respectivamente, as adaptações das fases GENI e US na 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 3.3.2.1. 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 15 apresenta o pseudocódigo da Fase GENI adaptada para o PRV-

CES, denominada VRGENI.

Algoritmo 15: Fase construtiva VRGENI

Seja m_0 o número inicial de rotas:
$m \leftarrow m_0$;
$V^- \leftarrow V$, onde V é o conjunto de cidades;
enquanto ($|V^-| > 0$) **faça**
  $s \leftarrow \emptyset$;
  $m' \leftarrow 1$;
  enquanto ($m' \leq m$) **faça**
    Selecione, aleatoriamente, duas cidades $v', v'' \in V^-$;
    $r \leftarrow$ rota contendo o depósito e as cidades v' e v'';
    $s \leftarrow s \cup \{r\}$;
    $V^- = V^- \setminus \{v'\}$;
    $V^- = V^- \setminus \{v''\}$;
    $m' \leftarrow m' + 1$;
  fim enquanto
  $V' = \emptyset$;
  enquanto ($|V^-| > 0$) **faça**
    Selecione, aleatoriamente, um vértice $v \in V^-$;
    $s' \leftarrow$ *InserçãoGENI(v,s)*; ;
    { Algoritmo 6 }
    se (*existe um arco* $\in s'$ *tal que sua carga exceda a capacidade do veículo*) **então**
      $V' = V' \cup \{v\}$;
    senão
      $s \leftarrow s'$;
      $V^- = V^- \cup V'$;
      $V' = \emptyset$;
    fim se
    $V^- = V^- \setminus \{v\}$;
  fim enquanto
  se ($|V'| > 0$) **então**
    $s \leftarrow \emptyset$;
    $V^- \leftarrow V$;
    $m \leftarrow m + 1$;
  fim se
fim enquanto
Retorne s;

Basicamente, a fase GENI adaptada ao PRVCES segue a mesma ideia 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. Além disso, o VRGENIUS analisa um espaço restrito de vizinhos de cada vértice, sendo, no máximo, igual a p. No algoritmo implementado p foi fixado no valor 3.

A Figura 4.14 apresenta a solução obtida pela fase VRGENI.

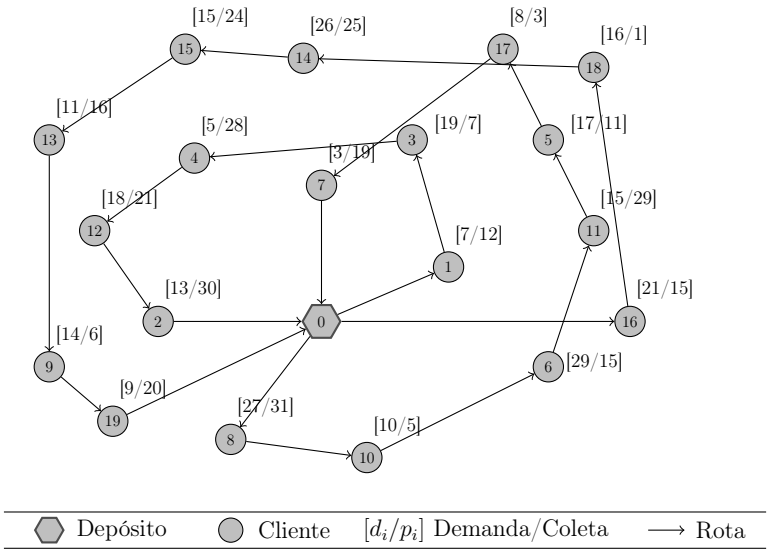

Figura 4.14: Solução gerada pela fase VRGENI.

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 3.3.2.2 e a adição é feita por meio das inserções IT1 e IT2 descritas na Subseção 3.3.2.1.

O pseudocódigo da fase US adaptada ao PRVCES é apresentado no Algoritmo 16. 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 visitada, seguindo a sequência de visitação dos clientes nas rotas. Assim, 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^r_{z+1}. Esse procedimento é realizado até que todos os vértices sejam analisados.

Algoritmo 16: Fase de refinamento VRUS

```
1: Seja s uma solução inicial:
2: s* ← s;
3: para ( r ∈ R(s) ) faça
4:    p^r_z ← p^r_1;
5:    enquanto ( p^r_z ≤ p^r_n ) faça
6:       v ← v(p^r_z, s);
7:       s' ← RemoçãoUS(v,r); ;
         { Algoritmo 10 }
8:       s'' ← InserçãoGENI(v,s'); ;
         { Algoritmo 6 }
9:       se ( f(s'') < f(s*) ) então
10:         s* ← s'';
11:         r ← primeira rota de R(s);
12:         p^r_z ← p^r_1;
13:      senão
14:         p^r_z ← p^r_{z+1};
15:      fim se
16:      s ← s'';
17:   fim enquanto
18: fim para
19: s ← s*;
20: Retorne s;
```

A Figura 4.15 mostra a solução gerada pela fase US adapatada ao PRVCES, considerando o exemplo da solução inicial gerada pela fase VRGENI (Figura 4.14). Como a solução inicial foi gerada pela fase VRGENI, então esta figura representa a solução final obtida pela heurística GENIUS adapatada ao PRVCES.

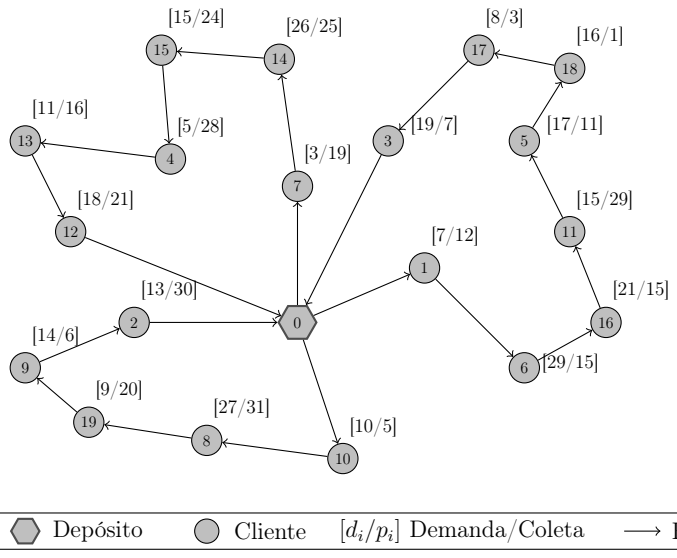

Figura 4.15: Solução gerada pela fase VRUS e pela heurística VRGENIUS.

4.5 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 são permitidos movimentos que conduzam a soluções inviáveis.

4.5.1 Movimento *Shift*

O *Shift* é um movimento de realocação que consiste em transferir um cliente i de uma rota para outra. A Figura 4.24 ilustra um exemplo em que o cliente 6 é transferido da Rota 2 à Rota 3.

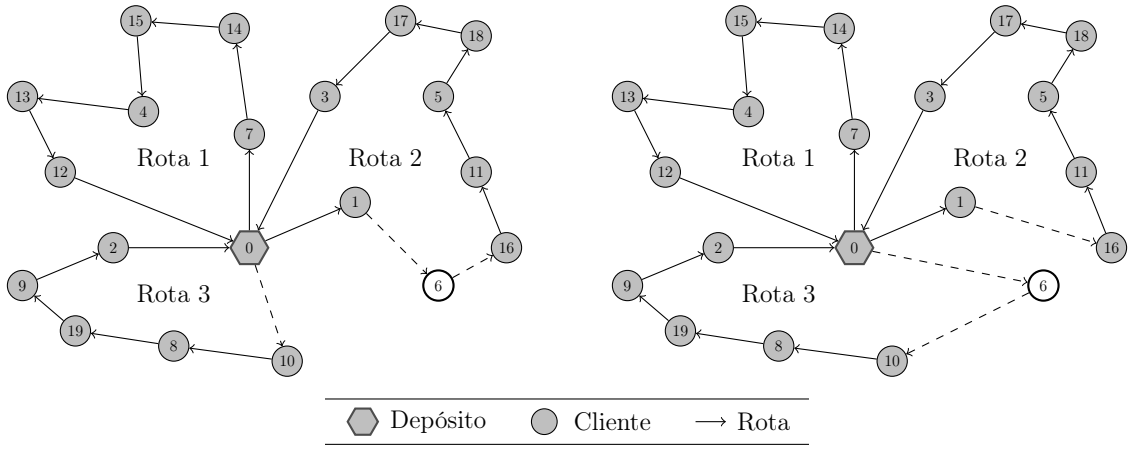

Figura 4.16: Exemplo do movimento *Shift*.

4.5.2 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 4.17 exemplifica a realocação dos clientes 1 e 6 da Rota 2 para a Rota 3.

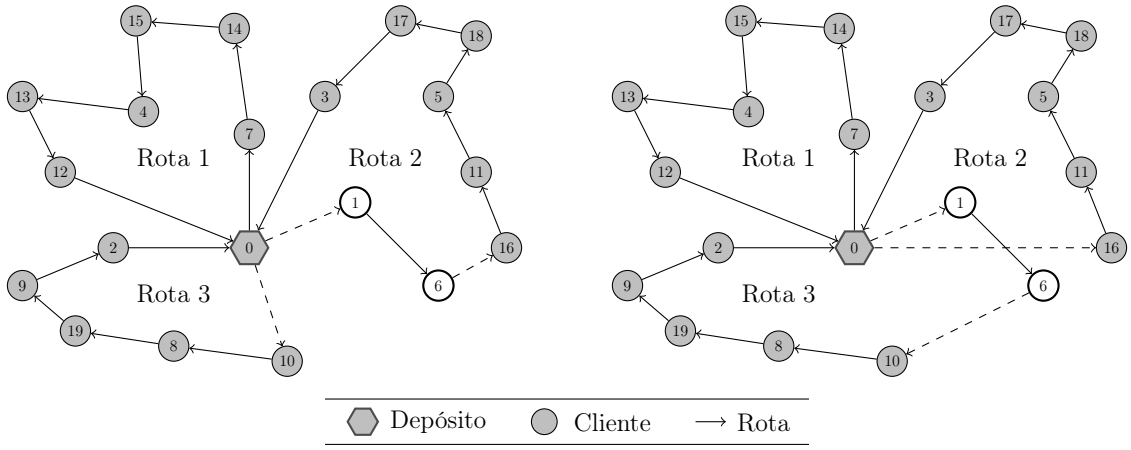

Figura 4.17: Exemplo do movimento *Shift*(2,0).

4.5.3 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 4.18 mostra a aplicação do movimento *Swap* para os clientes 1 e 10 das rotas 2 e 3, respectivamente.

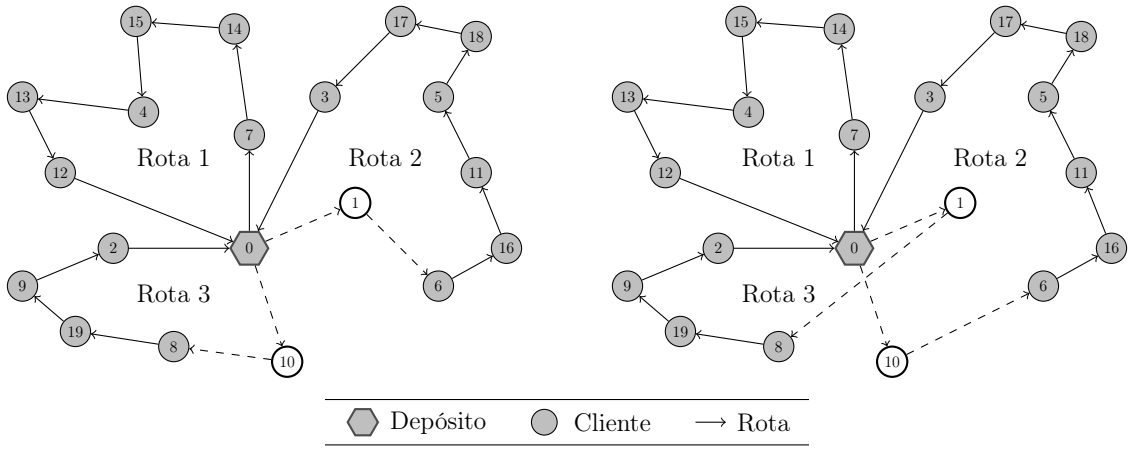

Figura 4.18: Exemplo do movimento *Swap*.

4.5.4 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 4.19 mostra a aplicação do movimento *Swap*(2,1) considerando os clientes 9 e 2 da Rota 3 e o cliente 12 da Rota 1.

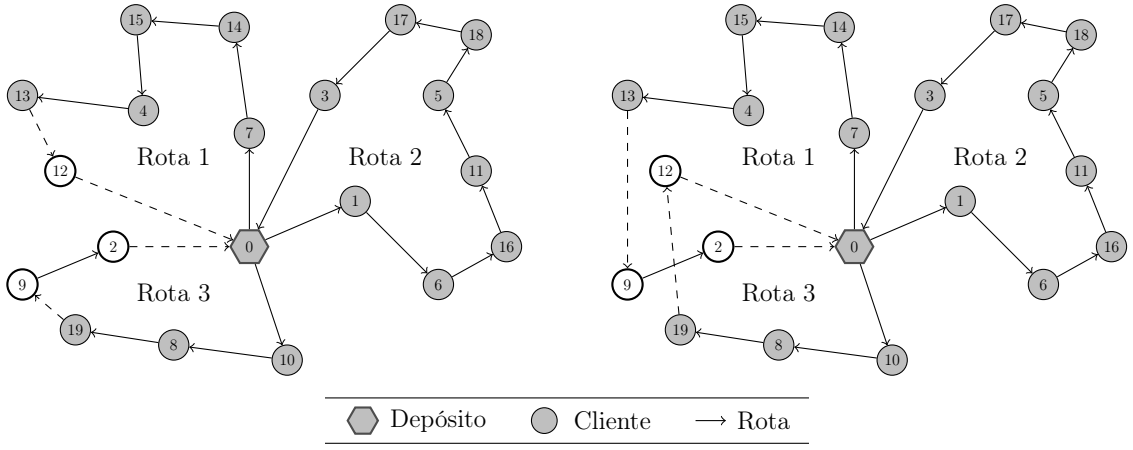

Figura 4.19: Exemplo do movimento *Swap*(2,1).

4.5.5 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 4.20.

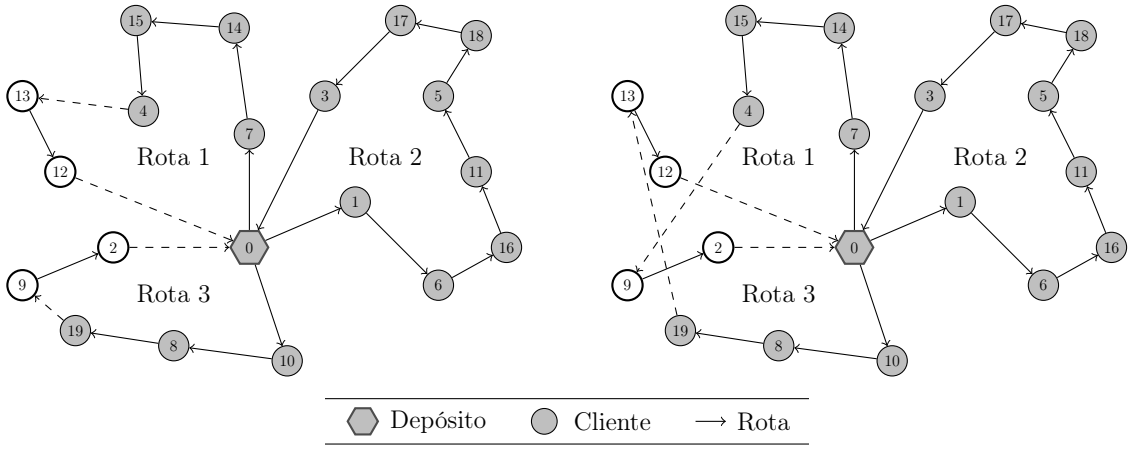

Figura 4.20: Exemplo do movimento *Swap*(2,2).

4.5.6 Movimento *2-Opt*

O *2-Opt* consiste em remover dois arcos e inserir dois novos arcos. A Figura 4.21 mostra a remoção dos arcos $(1, 6)$ e $(0, 10)$ e a inserção dos arcos $(1, 10)$ e $(0, 6)$.

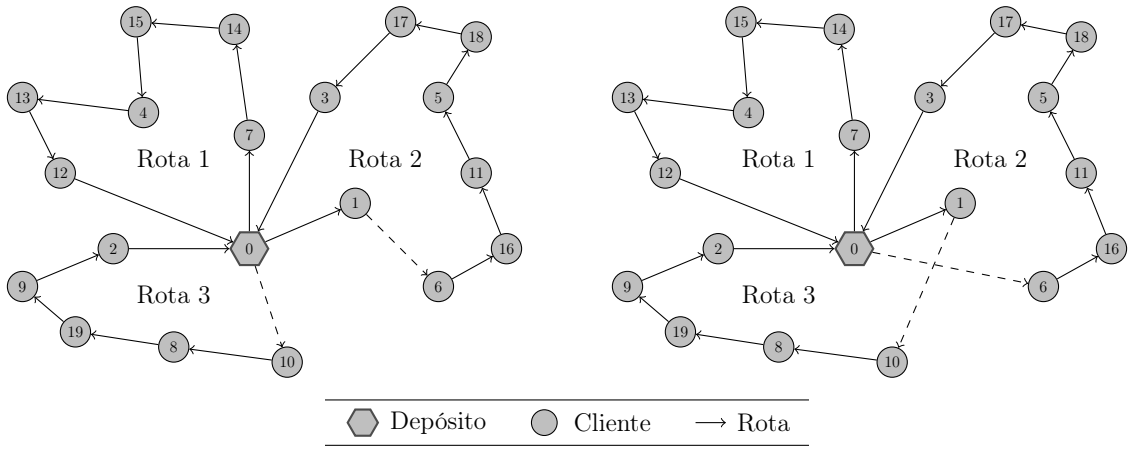

Figura 4.21: Exemplo do movimento *2-Opt*.

4.5.7 Movimento *kOr-Opt*

O movimento *kOr-Opt* consiste em remover k clientes consecutivos de uma rota r e, em seguida, reinseri-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 4.22 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 $(11, 5)$.

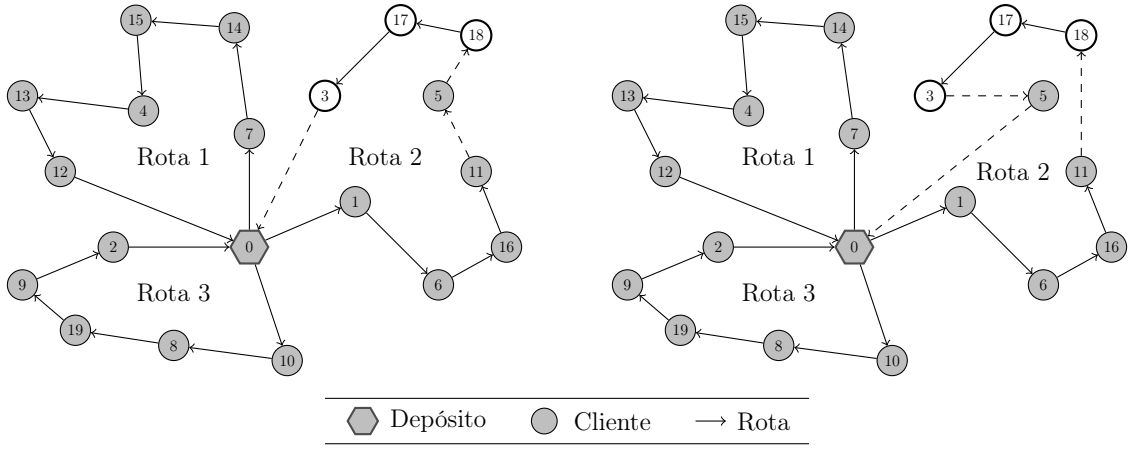

Figura 4.22: Exemplo do movimento *kOrOpt*.

4.6 *G3-opt*, *G4-opt* e *reverse*

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 3.3.2. 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 17.

Algoritmo 17: *G3-Opt*

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

Já o procedimento *G4-Opt* é semelhante ao *G3-Opt*, com a diferença de que, a cada iteração, são removidos os arcos (v_i, v_{i+1}), (v_{l-1}, v_l), (v_j, v_{j+1}) e (v_{k-1}, v_k) e adicionados os arcos (v_i, v_j), (v_l, v_{j+1}), (v_{k-1}, v_{l-1}) e (v_{i+1}, v_k). O Algoritmo 18

apresenta o pseudocódigo do *G4-Opt*.

Algoritmo 18: *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;

4.6.1 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 4.23 mostra a aplicação do *Reverse* na Rota 2.

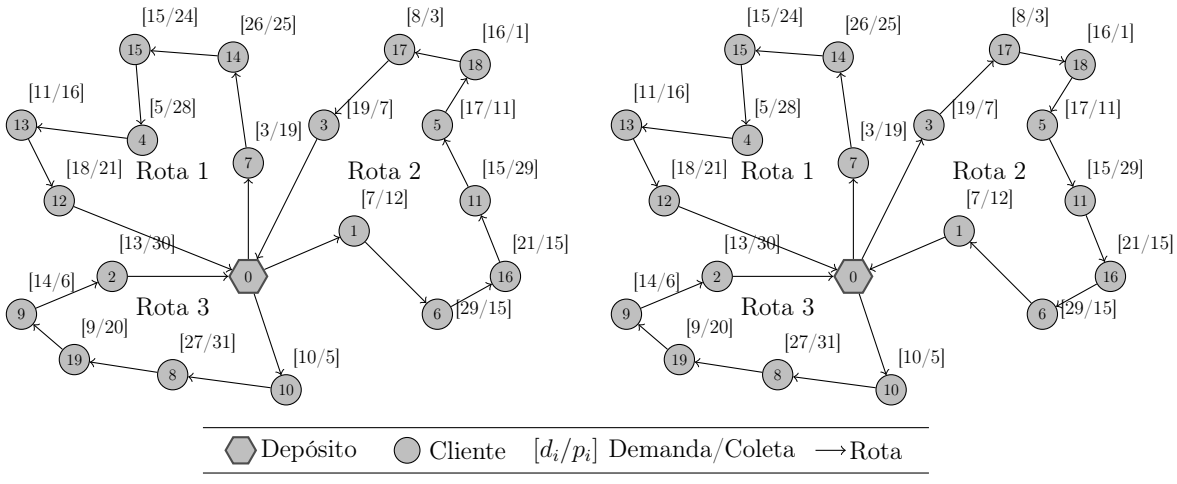

Figura 4.23: Aplicação do procedimento *Reverse* na Rota 2.

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 Lista de Candidatos

Para diminuir a quantidade de possibilidades analisadas pelas estruturas de vizinhança descritas na Seção 4.5, 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 4.24, 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).

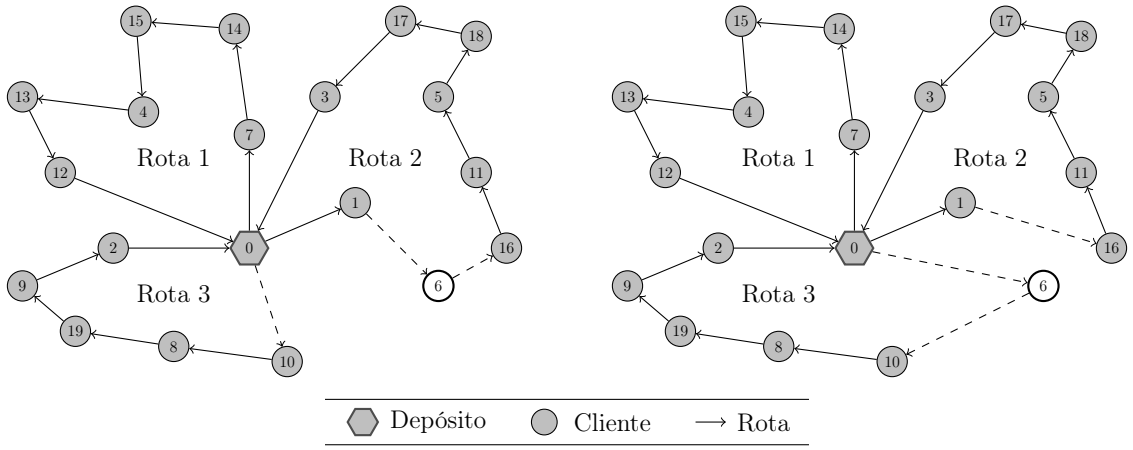

Figura 4.24: Exemplo do movimento *Shift*.

4.8 VND

O VND explora o espaço de soluções por meio de uma mudança sistemática de vizinhanças. No VND clássico considera-se um conjunto de vizinhanças em uma ordem pré-estabelecida, sendo que quando uma solução melhor é encontrada, retorna-se à primeira vizinhança.

O VND implementado é uma adaptação do mesmo método descrito no trabalho de Mine et al. (2010). Tal como esses autores, a cada iteração a ordem das vizinhanças a serem exploradas pode mudar, pois há uma ordenação aleatória das sete vizinhanças descritas na seção 4.5. Entretanto, após cada melhor vizinho escolhido aleatoriamente no passo anterior, é aplicada uma sequência aleatória de procedimentos de otimização, no caso, envolvendo a busca local *Best Improvement* com os sete movimentos descritos na seção 4.5 à página 40, e os procedimentos *G3-opt*, *G4-opt* e *reverse*, apresentados na seção 4.6, página 43. Em Mine et al. (2010), tais procedimentos de otimização são aplicados em uma sequência predeterminada. O pseudocódigo do VND implementado é apresentado no Algoritmo 19.

Algoritmo 19: VND

Seja r o número de estruturas distintas de vizinhanças, no caso, 7;
$\mathcal{RN} \leftarrow$ conjunto das r vizinhanças $\mathcal{N}$, descritas na Seção 4.5, em ordem aleatória;
$k \leftarrow 1$;
enquanto ($k \leq r$) **faça**
  Encontre o melhor vizinho $s' \in \mathcal{RN}^{(k)}(s)$;
  se ($f(s') < f(s)$) **então**
    {Intensificação nas rotas alteradas}
    Seja BL o conjunto de procedimentos de busca (no caso, 7 buscas locais associadas a cada um dos movimentos descritos na Seção 4.5 e os 3 procedimentos descritos na Seção 4.6), em ordem aleatória;
    Seja r_{BL} a cardinalidade do conjunto BL, no caso, 10;
    $s \leftarrow s'$;
    $k \leftarrow 1$;
    para ($j = 1$ até r_{BL}) **faça**
      $s' \in BL^j(s)$;
      $s \leftarrow s'$;
    fim para
  senão
    $k \leftarrow k + 1$;
  fim se
fim enquanto
Retorne s;

4.9 Busca Tabu

Busca Tabu (TS, do termo em inglês *Tabu Search*) é uma metaheurística que utiliza uma estrutura de memória para explorar o espaço de soluções de um problema. A TS caminha no espaço de soluções movendo-se de uma solução para outra que seja seu melhor vizinho, mesmo que este vizinho seja gerado por um movimento de piora. Como estratégia para não retornar a uma solução já gerada anteriormente e, portanto, ciclar, a TS 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 20 apresenta o pseudocódigo da Busca Tabu implementada.

Algoritmo 20: TS (*TamListaTabu*, *iterMaxTS*, *Solução s*)

```
1: s* ← s; {Melhor solução obtida até então}
2: TamOriginal ← TamListaTabu; {Tamanho original da Lista Tabu}
3: iter ← 0; {Número de iterações}
4: MelhorIter ← 0;{Número de iterações sem melhora na solução}
5: enquanto ( iter − MelhorIter ≤ iterMaxTS ) faça
6:    se ( iter − MelhorIter ≤ IterAux ) então
7:       TamListaTabu ← TamListaTabu + 1;
8:    fim se
9:    s' ← Shift(s);
10:   s'' ← Swap(s);
11:   s''' ← Shift(2,0)(s);
12:   s'''' ← Swap(2,1)(s);
13:   s''''' ← Swap(2,2)(s);
14:   s ← argmin{f(s'), f(s''), f(s'''), f(s''''), f(s''''')};
15:   AtualizaListaTabu( TamListaTabu, Lista, Ant, Prox, iter, Δ) {Algoritmo 21 }
16:   se ( f(s) < f(s*) ) então
17:      s* ← s;
18:      MelhorIter ← iter;
19:      TamListaTabu ← TamOriginal;
20:   fim se
21:   iter ← iter + 1;
22: fim enquanto
23: Retorne s*;
```

No Algoritmo 20, cinco soluções são geradas a cada iteração utilizando os movimentos *Shift*, *Swap*, *Shift(2,0)*, *Swap(2,1)* e *Swap(2,2)* descritos na Seção 4.5, 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 4.9.1. O tamanho da Lista Tabu (*TamListaTabu*) é incrementado em uma unidade à medida que o número de iterações sem melhora aumenta; 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.

4.9.1 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. Porém, vale ressaltar que no algoritmo implementado foi utilizado um critério de aspiração, que permite que um movimento tabu seja realizado, caso a solução produzida seja melhor que a melhor solução encontrada até então.

Para se atualizar a Lista Tabu é utilizada a função descrita pelo Algoritmo 21.

Algoritmo 21: *AtualizaListaTabu(TamListaTabu, Lista, Ant, Prox, iter, Δ)*

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

Neste trabalho foi utilizada uma Lista Tabu de tamanho 10, e um valor Δ igual a 3. Desta forma, a lista assume tamanhos diferentes, devido à 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 procura evitar a ocorrência de ciclagem.

Para representar computacionalmente a Lista Tabu, utilizou-se uma matriz quadrada, em que cada célula (i, j) representa até que iteração está proibido o cliente j ficar como sucessor do cliente i na rota. As Figuras 4.25 e 4.26 ilustram essa operação com o valor (variável *DuracaoTabu*) 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.

	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

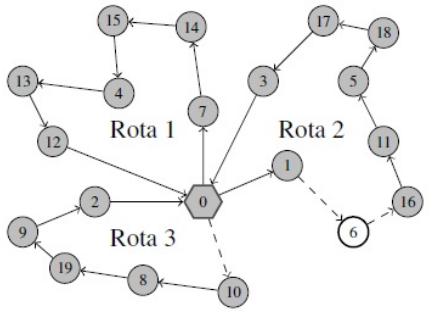

Figura 4.25: Configuração da Matriz antes do Movimento

	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

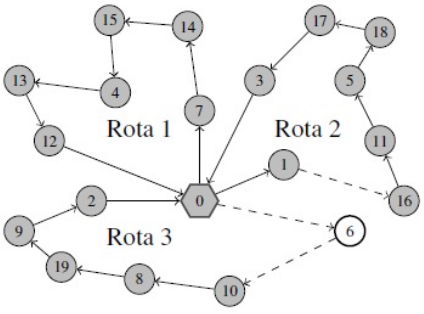

Figura 4.26: Atualização da Matriz após o movimento

Na Figura 4.25, 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 sai do depósito e atende aos clientes 1, 6, 16, 11, 5, 18, 17 e 3, nesta ordem, e retorna ao depósito. Na Figura 4.26 mostra-se o cliente 6 realocado da rota 2 para rota 3 pelo movimento *Shift*. A matriz tabu tem, então, a sua linha 1 e coluna 6 atualizada, 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.

4.10 Perturbações

As perturbações são realizadas por um dos três procedimentos: Múltiplos *Shifts*, Múltiplos *Swaps* e *Ejection chain*, escolhidos aleatoriamente.

Os procedimentos Múltiplos *Shifts* e Múltiplos *Swaps* consistem em k aplicações sucessivas dos movimentos *shift* e *swap*, sendo k um valor inteiro aleatório entre 1 e 3.

O *Ejection chain* foi proposto por (Rego e Roucairol, 1996). Inicialmente, seleciona-se um subconjunto de m rotas $R = r_1, r_2, \cdots, r_m$ de forma arbitrária. Em seguida, transfere-se um cliente da rota r_1 para a rota r_2, um cliente de r_2 para r_3 e assim sucessivamente até que um cliente seja transferido da rota r_m para a primeira rota r_1. Nesse movimento os clientes são escolhidos de forma aleatória.

4.11 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, 1996), 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 4.8 e 4.9. 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.

Capítulo 5

Resultados

O algoritmo GENILS-TS e suas duas variantes (GENILS-TS-CL e GENILS-TS-CL-RC) foram codificados em C++ usando o compilador Visual C++ 2005. Para testá-los, foi usado um microcomputador com processador Intel *Core 2 Quad*, 1,66 GHz e 4 GB de memória RAM e sistema operacional Windows Vista. Apesar de o microcomputador possuir quatro núcleos, o algoritmo proposto não explora multiprocessamento.

Para validá-los, foram usados 40 problemas-teste de Dethloff (2001); 14 de Salhi e Nagy (1999) e 18 de Montané e Galvão (2006). Os parâmetros adotados, obtidos experimentalmente em uma bateria preliminar de testes, foram os seguintes: $TamListaTabu = 10$, $iterMaxTS = 300$, $maxIter = 10000$ e $\Delta = 3$.

Inicialmente, na Seção 5.1, o algoritmo GENILS-TS é comparado com o algoritmo GENILS, de (Mine et al., 2010). A seguir, na seção 5.2 o GENILS-TS é comparado com sua variante GENILS-TS-CL, que usa uma lista de candidatos na exploração do espaço de soluções. Posteriormente, na seção 5.3, o algoritmo GENILS-TS-CL é comparado com sua variante GENILS-TS-CL-RC, que incorpora o procedimento de Reconexão por Caminhos. Finalmente, na seção 5.4, esta última variante do algoritmo é comparada com os principais algoritmos da literatura.

5.1 GENILS × GENILS-TS

Esta seção tem o objetivo de validar a inserção do módulo de Busca Tabu no algoritmo GENILS de Mine et al. (2010). Para tanto, são apresentados os resultados referentes à comparação entre o algoritmo desses autores e o GENILS-TS, assim denotado o algoritmo descrito na seção 4.2, sem os módulos de Lista de Candidatos (vide seção 4.7, página 46) e Reconexão por Caminhos (vide seção 4.11 à página 50). Na Subseção 5.1.1 são apresentados os resultados dos algoritmos utilizando como critério de parada o número de iterações sem melhora ($iterMax$) igual a 10000, que foi o mesmo adotado por Mine et al. (2010). Já na Subseção 5.1.2, são apresentados os resultados referentes a comparações entre o GENILS e o GENILS-TS utilizando como critério de parada o tempo de execução doss algoritmo.

5.1.1 Comparação pelo número de iterações

Nesta Seção são apresentados os resultados referentes à comparação do GENILS com o GENILS-TS tendo como critério de parada o número máximo de iterações sem melhora desses algoritmos. Nesses testes busca-se avaliar a capacidade dos algoritmos encontrarem soluções de qualidade sem levar em consideração o tempo de processamento.

As Tabelas 5.1, 5.2 e 5.3 comparam os resultados dos algoritmos analisados nas instâncias-teste utilizadas. Nestas tabelas, a primeira coluna representa o problema-teste e a segunda o melhor resultado da literatura. As colunas *Melhor* e *Média* apresentam, respectivamente, o melhor resultado de cada algoritmo e a média referente ao resultado encontrado pelo algoritmo nas 30 execuções. Já a coluna $Desv^{Avg}$ apresenta o desvio percentual do valor das soluções médias em relação ao melhor resultado da literatura, calculado pela equação (5.1). A coluna $Desv^{Best}$ apresenta o desvio percentual do melhor resultado encontrado pelo algoritmo em relação ao melhor conhecido na literatura, sendo calculado pela Equação (5.2). Já a coluna *Tempo* apresenta o tempo médio das 30 execuções em segundos.

$$Desv^{Avg} = \frac{Média - Melhor\ Lit.}{Melhor\ Lit.} \times 100 \tag{5.1}$$

$$Desv^{Best} = \frac{Melhor - Melhor\ Lit.}{Melhor\ Lit.} \times 100 \tag{5.2}$$

Ao analisar a Tabela 5.1, onde são comparados os algoritmos GENILS e GENILS-TS nos problemas-teste de Dethloff (2001), observamos que a incorporação do módulo TS em GENILS melhorou o desempenho do algoritmo em termos da capacidade de encontrar a melhor solução. Apesar de os dois algoritmos terem alcançado os melhores resultados conhecidos na literatura, o GENILS não foi capaz de encontrar a melhor solução em todas as suas execuções, já que o $Desv^{Avg}$ chega a ser até de 0,96%. Já o GENILS-TS obteve todos os valores de $Desv^{Avg}$ iguais a 0, o que significa que o melhor resultado foi encontrado em todas as execuções. Quando o ponto de análise passa a ser o tempo médio das execuções, observa-se que o módulo de Busca Tabu fez aumentar o tempo de execução do algoritmo, uma vez que, considerando a média de todas as execuções, o GENILS foi processado em 9,02 segundos, ao passo que o GENILS-TS demandou 13,61 segundos.

Analisando a Tabela 5.2, que contém os resultados da comparação dos algoritmos GENILS e GENILS-TS nos problemas-teste de Salhi e Nagy (1999), verifica-se que o algoritmo GENILS-TS encontrou 6 melhores resultados conhecidos na literatura, enquanto o GENILS só encontrou 4. Constata-se, também, que o GENILS-TS também obteve um melhor desempenho global, já que as suas soluções estão mais próximas das melhores da literatura. Isso pode ser observado analisando-se as colunas $Desv^{Best}$ e $Desv^{Avg}$, que mostram que todos os desvios do GENILS-TS foram menores. Porém, quando observamos o tempo médio de execução, o GENILS-TS se mostrou consideravelmente mais lento.

Na Tabela 5.3, que contém os resultados da comparação de desempenho entre os algoritmos GENILS e GENILS-TS nos problemas-teste de Montané e Galvão (2006), pode-se constatar o que já foi verificado na tabela anterior. Neste conjunto

Tabela 5.1: Comparação do GENILS × GENILS-TS nos problemas-teste de Dethloff (2001) tendo como critério de parada o número de iterações

Prob.	**Melhor**	**GENILS**					**GENILS-TS**				
-	**Lit.**	**Melhor**	**Média**	**Desv^{Avg}** (%)	**Desv^{Best}** (%)	**Tempo(s)** (s)	**Melhor**	**Média**	**Desv^{Avg}** (%)	**Desv^{Best}** (%)	**Tempo(s)** (s)
SCA3-0	**635,62**	**635,62**	638,89	0,51	0,00	9,46	**635,62**	**635,62**	0,00	0,00	10,32
SCA3-1	**697,84**	**697,84**	699,44	0,23	0,00	10,14	**697,84**	**697,84**	0,00	0,00	13,30
SCA3-2	**659,34**	**659,34**	**659,34**	0,00	0,00	9,06	**659,34**	**659,34**	0,00	0,00	9,31
SCA3-3	**680,04**	**680,04**	681,55	0,22	0,00	9,87	**680,04**	**680,04**	0,00	0,00	10,29
SCA3-4	**690,50**	**690,50**	**690,50**	0,00	0,00	9,27	**690,50**	**690,50**	0,00	0,00	12,23
SCA3-5	**659,90**	**659,90**	**659,90**	0,00	0,00	9,45	**659,90**	**659,90**	0,00	0,00	15,21
SCA3-6	**651,09**	**651,09**	**651,09**	0,00	0,00	10,03	**651,09**	**651,09**	0,00	0,00	13,41
SCA3-7	**659,17**	**659,17**	661,09	0,29	0,00	10,11	**659,17**	**659,17**	0,00	0,00	10,25
SCA3-8	**719,48**	**719,47**	720,38	0,13	0,00	11,14	**719,48**	**719,48**	0,00	0,00	21,17
SCA3-9	**681,00**	**681,00**	681,15	0,02	0,00	10,32	**681,00**	**681,00**	0,00	0,00	16,17
SCA8-0	**961,50**	**961,50**	968,02	0,68	0,00	7,54	**961,50**	**961,50**	0,00	0,00	19,65
SCA8-1	**1049,65**	**1049,65**	1057,50	0,75	0,00	7,86	**1049,65**	**1049,65**	0,00	0,00	11,81
SCA8-2	**1039,64**	**1039,64**	1049,67	0,96	0,00	8,83	**1039,64**	**1039,64**	0,00	0,00	9,27
SCA8-3	**983,34**	**983,34**	984,88	0,16	0,00	7,74	**983,34**	**983,34**	0,00	0,00	15,17
SCA8-4	**1065,49**	**1065,49**	1067,32	0,17	0,00	7,40	**1065,49**	**1065,49**	0,00	0,00	10,26
SCA8-5	**1027,08**	**1027,08**	1028,73	0,16	0,00	9,96	**1027,08**	**1027,08**	0,00	0,00	11,21
SCA8-6	**971,82**	**971,82**	972,80	0,10	0,00	8,11	**971,82**	**971,82**	0,00	0,00	18,13
SCA8-7	**1051,28**	**1051,28**	1060,40	0,87	0,00	9,50	**1051,28**	**1051,28**	0,00	0,00	12,26
SCA8-8	**1071,18**	**1071,18**	1077,18	0,56	0,00	7,42	**1071,18**	**1071,18**	0,00	0,00	10,07
SCA8-9	**1060,50**	**1060,50**	1065,10	0,43	0,00	7,25	**1060,50**	**1060,50**	0,00	0,00	10,55
CON3-0	**616,52**	**616,52**	618,89	0,38	0,00	9,46	**616,52**	**616,52**	0,00	0,00	11,49
CON3-1	**554,47**	**554,47**	556,02	0,28	0,00	9,78	**554,47**	**554,47**	0,00	0,00	10,36
CON3-2	**518,00**	**518,00**	522,41	0,85	0,00	11,12	**518,00**	**518,00**	0,00	0,00	15,11
CON3-3	**591,19**	**591,19**	**591,19**	0,00	0,00	9,83	**591,19**	**591,19**	0,00	0,00	20,30
CON3-4	**588,79**	**588,79**	591,60	0,48	0,00	9,88	**588,79**	**588,79**	0,00	0,00	15,62
CON3-5	**563,70**	**563,70**	566,75	0,54	0,00	11,05	**563,70**	**563,70**	0,00	0,00	21,70
CON3-6	**499,05**	**499,05**	502,56	0,70	0,00	10,41	**499,05**	**499,05**	0,00	0,00	14,39
CON3-7	**576,48**	**576,48**	580,48	0,69	0,00	8,70	**576,48**	**576,48**	0,00	0,00	12,51
CON3-8	**523,05**	**523,05**	523,49	0,08	0,00	8,10	**523,05**	**523,05**	0,00	0,00	11,10
CON3-9	**578,25**	**578,24**	583,77	0,96	0,00	11,51	**578,25**	**578,25**	0,00	0,00	21,25
CON8-0	**857,17**	**857,17**	859,97	0,33	0,00	9,71	**857,17**	**857,17**	0,00	0,00	10,43
CON8-1	**740,85**	**740,85**	742,78	0,26	0,00	9,21	**740,85**	**740,85**	0,00	0,00	13,60
CON8-2	**712,89**	**712,89**	713,15	0,04	0,00	9,20	**712,89**	**712,89**	0,00	0,00	10,05
CON8-3	**811,07**	**811,07**	817,91	0,84	0,00	7,93	**811,07**	**811,07**	0,00	0,00	14,24
CON8-4	**772,25**	**772,25**	772,98	0,09	0,00	6,75	**772,25**	**772,25**	0,00	0,00	8,17
CON8-5	**754,88**	**754,88**	757,20	0,31	0,00	7,77	**754,88**	**754,88**	0,00	0,00	11,49
CON8-6	**678,92**	**678,92**	684,38	0,80	0,00	9,37	**678,92**	**678,92**	0,00	0,00	17,08
CON8-7	**811,96**	**811,96**	**811,96**	0,00	0,00	4,55	**811,96**	**811,96**	0,00	0,00	10,03
CON8-8	**767,53**	**767,53**	769,68	0,28	0,00	8,35	**767,53**	**767,53**	0,00	0,00	19,36
CON8-9	**809,00**	**809,00**	810,55	0,19	0,00	7,77	**809,00**	**809,00**	0,00	0,00	16,25
Média	**-**	**-**	**-**	**0,36**	**0,00**	**9,02**	**-**	**-**	**0,00**	**0,00**	**13,61**

de problemas-teste, o GENILS-TS obteve 12 melhores soluções conhecidas da literatura, enquanto o GENILS obteve 11. Em relação ao $Desv^{Avg}$ e $Desv^{Best}$, o GENILS-TS também obteve melhor desempenho, com redução no valor dessas métricas. Em contrapartida, o GENILS executou seu algoritmo em tempos computacionais muito menores.

Foram feitos testes estatísticos, como forma de validar a melhora do algoritmo GENILS-TS frente ao algoritmo GENILS, observada na análise das tabelas de resultados. Para isso, foi realizada uma análise de variância - Teste T (Montgomery, 2007) dos valores de $Desv^{Avg}$ encontrados pelos dois algoritmos. Pelo resultado do Teste T pode-se dizer que existe diferença estatística entre os algoritmos, pois o p resultante da análise foi igual a 0,03 que é menor que o $threshold = 0{,}05$. Sendo assim, pode-se afirmar pelo Teste T que o GENILS-TS é estatisticamente melhor que o GENILS, para esta medida de comparação com 95% de grau de confiança.

Tabela 5.2: Comparação GENILS × GENILS-TS nos problemas-teste de Salhi e Nagy (1999) tendo como critério de parada o número de iterações

Prob.	**Melhor**	**GENILS**					**GENILS-TS**				
-	**Lit.**	**Melhor**	**Média**	**Desv^{Avg} (%)**	**Desv^{Best} (%)**	**Tempo(s) (s)**	**Melhor**	**Média**	**Desv^{Avg} (%)**	**Desv^{Best} (%)**	**Tempo(s) (s)**
CMT1X	**466,77**	**466,77**	472,23	1,17	0,00	11,37	**466,77**	472,23	1,17	0,00	2,25
CMT1Y	**466,77**	**466,77**	472,22	1,17	0,00	10,89	**466,77**	472,23	1,17	0,00	62,20
CMT2X	**668,77**	684,11	693,94	3,76	2,29	19,70	684,11	693,40	3,68	2,29	127,97
CMT2Y	**663,25**	684,11	693,58	4,57	3,15	20,97	684,11	693,40	4,55	3,15	132,64
CMT3X	**721,27**	721,40	726,30	0,70	0,02	60,60	**721,27**	726,34	0,70	0,00	265,77
CMT3Y	**721,27**	**721,27**	730,76	1,32	0,00	61,07	**721,27**	730,56	1,29	0,00	256,88
CMT12X	**644,70**	662,22	677,64	5,11	2,72	56,66	662,22	666,77	3,42	2,72	269,68
CMT12Y	**659,52**	663,50	678,85	2,93	0,60	45,50	662,22	673,91	2,18	0,41	208,54
CMT11X	**833,92**	846,23	875,66	5,01	1,48	84,20	**833,92**	867,96	4,08	0,00	416,26
CMT11Y	**830,39**	836,04	871,93	5,00	0,68	133,12	833,92	856,32	3,12	0,43	488,47
CMT4X	**852,46**	**852,46**	872,68	2,37	0,00	194,59	**852,46**	865,03	1,47	0,00	858,95
CMT4Y	**852,35**	862,28	878,79	3,10	1,17	149,12	855,52	866,06	1,61	0,37	406,01
CMT5X	**1029,25**	1033,51	1066,43	3,61	0,41	410,52	1030,50	1052,67	2,28	0,12	1655,14
CMT5Y	**1029,25**	1036,14	1044,37	1,47	0,67	688,91	1030,50	1053,73	2,38	0,12	790,88
Média	**-**	**-**	**-**	**2,95**	**0,94**	**139,09**	**-**	**-**	**2,36**	**0,69**	**424,40**

Outra comparação de desempenho feita entre os algoritmos GENILS e GENILS-TS foi com relação a gráficos *boxplot*. A Figura 5.1 mostra o gráfico *boxplot* construído com os valores de $Desv^{Avg}$ de cada algoritmo. Por ele pode-se observar a superioridade do GENILS-TS frente ao GENILS, já que o seu limite inferior (50% dos $Desv^{Avg}$) é igual 0, significa que 50% das médias encontradas são iguais aos melhores resultados alcançados pelo algoritmo. Também é importante destacar a inferioridade do GENILS, pois 25% de seus valores de $Desv^{Avg}$ são equivalentes a 50% dos valores $Desv^{Avg}$ do GENILS-TS.

Tabela 5.3: Comparação GENILS × GENILS-TS nos problemas-teste de Montané e Galvão (2006) tendo como critério de parada o número de iterações

Prob.	Melhor	GENILS					GENILS-TS				
-	Lit.	Melhor	Média	$Desv^{Avg}$ (%)	$Desv^{Best}$ (%)	Tempo(s) (s)	Melhor	Média	$Desv^{Avg}$ (%)	$Desv^{Best}$ (%)	Tempo(s) (s)
r101	**1009,95**	**1009,95**	1016,70	0,67	0,00	24,72	**1009,95**	1013,71	0,37	0,00	208,56
r201	**666,20**	**666,20**	667,72	0,23	0,00	37,23	**666,20**	666,81	0,09	0,00	165,84
c101	**1220,18**	**1220,18**	1224,91	0,39	0,00	22,93	**1220,18**	1222,81	0,22	0,00	203,19
c201	**662,07**	**662,07**	664,00	0,29	0,00	24,04	**662,07**	664,31	0,34	0,00	91,87
rc101	**1059,32**	**1059,32**	1065,97	0,63	0,00	21,26	**1059,32**	1064,49	0,49	0,00	93,86
rc201	**672,92**	**672,92**	680,00	1,05	0,00	32,60	**672,92**	677,49	0,68	0,00	189,50
r1_2_1	**3357,64**	**3357,64**	3433,83	2,27	0,00	196,71	**3357,64**	3450,31	2,76	0,00	890,19
r2_2_1	**1665,58**	**1665,58**	1687,39	1,31	0,00	143,94	**1665,58**	1670,35	0,29	0,00	1038,79
c1_2_1	**3629,89**	3636,74	3658,05	0,78	0,19	188,17	3634,65	3657,16	0,75	0,13	800,47
c2_2_1	**1726,59**	**1726,59**	1751,59	1,45	0,00	116,87	**1726,58**	1745,10	1,07	0,00	773,48
rc1_2_1	**3306,00**	3312,92	3345,80	1,20	0,21	207,58	3312,92	3327,22	0,64	0,21	977,57
rc2_2_1	**1560,00**	**1560,00**	1579,64	1,26	0,00	135,44	**1560,00**	1584,84	1,59	0,00	914,61
r1_4_1	**9605,75**	9627,43	9730,69	1,30	0,23	1599,83	9627,88	9706,91	1,05	0,23	5415,90
r2_4_1	**3551,38**	3582,08	3648,61	2,74	0,86	895,47	3582,08	3612,11	1,71	0,86	4027,96
c1_4_1	**11098,21**	**11098,21**	11147,40	0,44	0,00	1437,12	**11098,21**	11133,59	0,32	0,00	3757,66
c2_4_1	**3546,10**	3596,37	3664,53	3,34	1,42	782,31	3592,71	3617,86	2,02	1,31	3293,82
rc1_4_1	**9535,46**	**9535,46**	9640,17	1,10	0,00	1947,84	**9535,46**	9638,30	1,08	0,00	6004,00
rc2_4_1	**3403,70**	3422,11	3503,29	2,93	0,54	2005,14	3419,76	3502,01	2,89	0,47	3600,39
Média	-	-	-	**1,30**	**0,19**	**545,51**	-	-	**1,02**	**0,18**	**1802,65**

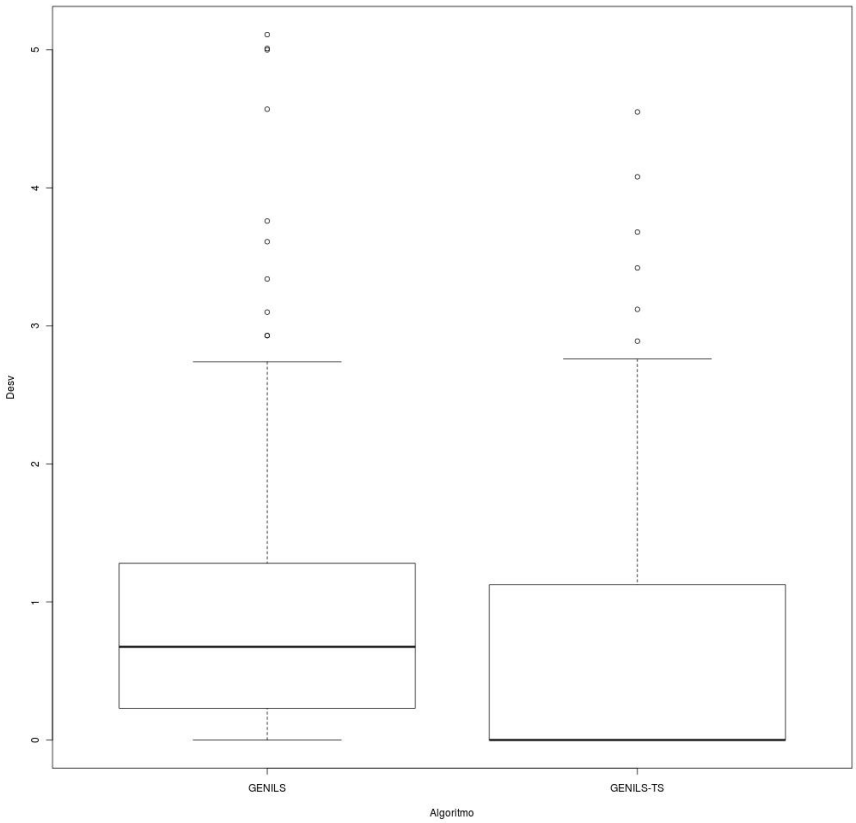

Figura 5.1: Boxplot mostrando os desvios médios dos algoritmos

Desta forma, conclui-se que a inserção do módulo de Busca Tabu no GENILS

influenciou significativamente a qualidade dos resultados. Porém, é importante ressaltar o aumento substancial do tempo de execução do algoritmo.

Como uma alternativa de tentar reduzir os tempos computacionais, sem prejuízo da qualidade das soluções finais, foi adicionado ao algoritmo GENILS-TS o método apresentado na Seção 4.7, página 46, que usa uma Lista de Candidatos na exploração das vizinhanças. Os resultados obtidos podem ser vistos na próxma Seção.

5.1.2 Comparação pelo tempo de processamento

Essa segunda bateria de testes, comparando o GENILS com o GENILS-TS, foi feita como forma de validar as análises feitas na Subseção anterior. Para isso, foi fixado como critério de parada do GENILS o tempo médio de execução do GENILS-TS de cada problema-teste, dado pela coluna Tempo nas Tabelas 5.1, 5.2 e 5.3.

Desta forma, procura-se avaliar a capacidade dos algoritmos encontrarem resultados de qualidade dentro de um determinado tempo de execução.

Os resultados encontrados podem ser visualizados nas Tabelas 5.4, 5.5 e 5.6. Em todas estas Tabelas, pode-se observar conclusões parecidas com as apresentadas na Subseção 5.1.1. Esses resultados mostram que o GENILS-TS obteve uma menor variabilidade nas soluções encontradas, além de alcançar soluções de melhor qualidade, sendo mais próximas ou iguais às das melhores soluções conhecidas na literatura. Essas constatações podem ser obtidas analisando-se os valores de Desv^{Avg} e Desv^{Best} em ambos algoritmos.

Para validar a melhora do algoritmo GENILS-TS frente ao algoritmo GENILS apresentadas nesta Seção, foi realizada uma análise de variância Teste T dos valores de $Desv^{Avg}$ encontrados pelos dois algoritmos. Pelo resultado obtido, pode-se dizer que existe diferença estatística entre os algoritmos, pois o p resultante da análise foi igual a 0,049 que é menor que o *threshold* = 0,05. Sendo assim, pode-se afirmar pelo Teste T que o GENILS-TS é estatisticamente melhor que o GENILS para esta medida de comparação com 95% de grau de confiança.

Para comparar a convergência do algoritmo GENILS-TS frente à do GENILS, foi realizada uma análise de probabilidade empírica, apresentada pela Figura 5.3. Para tanto, foi escolhido o problemas-teste `CON8-7` de (Dethloff, 2001). O critério de parada do algoritmo foi alterado para finalizar quando fosse encontrado o valor alvo, no caso, definido como o melhor valor encontrado na literatura para o respectivo problema-teste.

Para gerar o gráfico de distribuição de probabilidade em relação ao tempo para encontrar o alvo, foi utilizado o método descrito em (Aiex et al., 2002). Esse método consiste, inicialmente, em ordenar os tempos $T = \{t_1, t_2, \ldots, 100\}$ que os algoritmos encontraram o valor alvo, sendo 100 o número de execuções. Em seguida, calculou-se a probabilidade acumulada $p_i = (i - 1/2)/100$ associada ao i-ésimo tempo de execução. Por fim, os 100 pontos $z_i = (t_i, p_i)$ foram plotados.

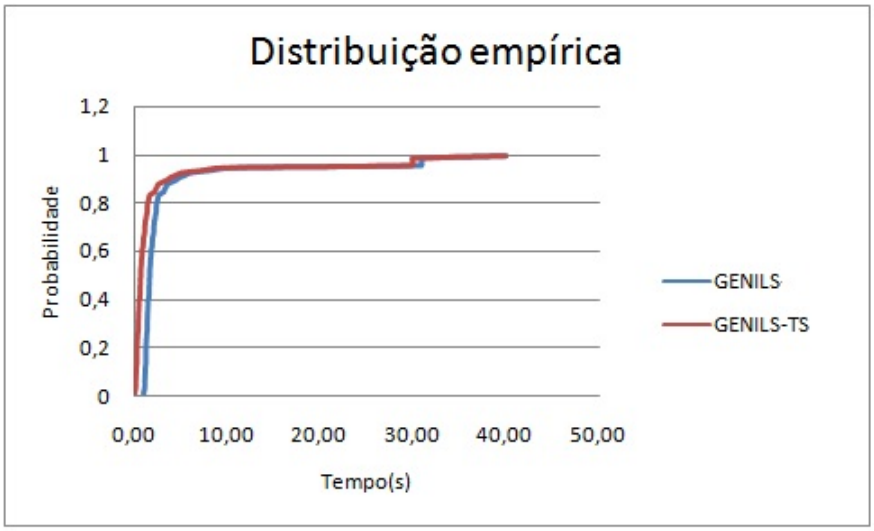

Figura 5.2: Probabilidade acumulada dos algoritmos GENILS-TS e GENILS no problema-teste `CON8-7`

Ao analisar as curvas de probabilidades empíricas pode-se perceber que as curvas de desempenho dos dois algoritmos são bastante semelhantes. Entretanto, observa-se que o GENILS-TS alcança o alvo desejado com um tempo inferior ao GENILS na maioria das vezes.

5.2 GENILS-TS × GENILS-TS-CL

Apresenta-se, nesta Seção, a comparação entre os algoritmos GENILS-TS e GENILS-TS-CL. O segundo é um aprimoramento do primeiro, que incorpora a Lista de Candidatos (Seção 4.7) com o objetivo de tentar diminuir o tempo de execução da versão GENILS-TS.

Nas Tabelas 5.7, 5.8 e 5.9 são apresentados, respectivamente, os resultados desses algoritmos nos problemas-teste de Dethloff (2001), Salhi e Nagy (1999) e Montané e Galvão (2006).

Avaliando os resultados encontrados nas Tabelas 5.7, 5.8 e 5.9, nota-se que o GENILS-TS-CL conseguiu, em todos os problemas-testes, uma melhora no tempo médio de execução do algoritmo, mantendo os mesmos melhores resultados encontrados pelo GENILS-TS.

Porém, quando são analisadas as médias das soluções, nota-se que houve uma pequena piora. Acredita-se, que este resultado se dá não só pelo fato de o algoritmo ser estocástico, mas também por menos soluções terem sido geradas, já que a Lista de Candidatos limita os movimentos da busca local. Entretanto, mesmo assim, os melhores resultados foram encontrados, mostrando que a restrição adicionada não exclui do espaço de soluções a melhor solução.

Para comparar a convergência do algoritmo GENILS-TS-CL frente à do GENILS-TS, foi realizada uma análise de probabilidade empírica, apresentada pela Figura 5.3. Para tanto, foi escolhido o problemas-teste `CON8-7` de (Dethloff, 2001). O critério de parada do algoritmo foi alterado para finalizar quando fosse encontrado o valor

Tabela 5.4: Comparação GENILS × GENILS-TS nos problemas-teste de Dethloff (2001) tendo como critério de parada o tempo de processamento

Prob.	**Melhor**	**GENILS**					**GENILS-TS**				
-	**Lit.**	**Melhor**	**Média**	**Desv^{Avg}** (%)	**Desv^{Best}** (%)	**Tempo(s)** (s)	**Melhor**	**Média**	**Desv^{Avg}** (%)	**Desv^{Best}** (%)	**Tempo(s)** **(s)**
SCA3-0	**635,62**	**635,62**	636,89	0,20	0,00	10,32	**635,62**	**635,62**	0,00	0,00	10,32
SCA3-1	**697,84**	**697,84**	698,44	0,09	0,00	13,30	**697,84**	**697,84**	0,00	0,00	13,30
SCA3-2	**659,34**	**659,34**	**659,34**	0,00	0,00	9,31	**659,34**	**659,34**	0,00	0,00	9,31
SCA3-3	**680,04**	**680,04**	**680,04**	0,00	0,00	10,29	**680,04**	**680,04**	0,00	0,00	10,29
SCA3-4	**690,50**	**690,50**	**690,50**	0,00	0,00	12,23	**690,50**	**690,50**	0,00	0,00	12,23
SCA3-5	**659,90**	**659,90**	**659,90**	0,00	0,00	15,21	**659,90**	**659,90**	0,00	0,00	15,21
SCA3-6	**651,09**	**651,09**	**651,09**	0,00	0,00	13,41	**651,09**	**651,09**	0,00	0,00	13,41
SCA3-7	**659,17**	**659,17**	660,12	0,14	0,00	10,25	**659,17**	**659,17**	0,00	0,00	10,25
SCA3-8	**719,48**	**719,47**	720,09	0,09	0,00	21,17	**719,48**	**719,48**	0,00	0,00	21,17
SCA3-9	**681,00**	**681,00**	**681,00**	0,00	0,00	16,17	**681,00**	**681,00**	0,00	0,00	16,17
SCA8-0	**961,50**	**961,50**	966,25	0,49	0,00	19,65	**961,50**	**961,50**	0,00	0,00	19,65
SCA8-1	**1049,65**	**1049,65**	1049,68	0,00	0,00	11,81	**1049,65**	**1049,65**	0,00	0,00	11,81
SCA8-2	**1039,64**	**1039,64**	1043,74	0,39	0,00	9,27	**1039,64**	**1039,64**	0,00	0,00	9,27
SCA8-3	**983,34**	**983,34**	984,12	0,08	0,00	15,17	**983,34**	**983,34**	0,00	0,00	15,17
SCA8-4	**1065,49**	**1065,49**	1066,87	0,13	0,00	10,26	**1065,49**	**1065,49**	0,00	0,00	10,26
SCA8-5	**1027,08**	**1027,08**	1027,09	0,00	0,00	11,21	**1027,08**	**1027,08**	0,00	0,00	11,21
SCA8-6	**971,82**	**971,82**	972,10	0,03	0,00	18,13	**971,82**	**971,82**	0,00	0,00	18,13
SCA8-7	**1051,28**	**1051,28**	1058,65	0,70	0,00	12,26	**1051,28**	**1051,28**	0,00	0,00	12,26
SCA8-8	**1071,18**	**1071,18**	1072,34	0,11	0,00	10,07	**1071,18**	**1071,18**	0,00	0,00	10,07
SCA8-9	**1060,50**	**1060,50**	1063,50	0,28	0,00	10,55	**1060,50**	**1060,50**	0,00	0,00	10,55
CON3-0	**616,52**	**616,52**	617,16	0,10	0,00	11,49	**616,52**	**616,52**	0,00	0,00	11,49
CON3-1	**554,47**	**554,47**	555,65	0,21	0,00	10,36	**554,47**	**554,47**	0,00	0,00	10,36
CON3-2	**518,00**	**518,00**	519,53	0,30	0,00	15,11	**518,00**	**518,00**	0,00	0,00	15,11
CON3-3	**591,19**	**591,19**	**591,19**	0,00	0,00	20,30	**591,19**	**591,19**	0,00	0,00	20,30
CON3-4	**588,79**	**588,79**	589,20	0,07	0,00	15,62	**588,79**	**588,79**	0,00	0,00	15,62
CON3-5	**563,70**	**563,70**	564,01	0,05	0,00	21,70	**563,70**	**563,70**	0,00	0,00	21,70
CON3-6	**499,05**	**499,05**	500,98	0,39	0,00	14,39	**499,05**	**499,05**	0,00	0,00	14,39
CON3-7	**576,48**	**576,48**	579,14	0,46	0,00	12,51	**576,48**	**576,48**	0,00	0,00	12,51
CON3-8	**523,05**	**523,05**	**523,05**	0,00	0,00	11,10	**523,05**	**523,05**	0,00	0,00	11,10
CON3-9	**578,25**	**578,24**	580,98	0,47	0,00	21,25	**578,25**	**578,25**	0,00	0,00	21,25
CON8-0	**857,17**	**857,17**	857,83	0,08	0,00	10,43	**857,17**	**857,17**	0,00	0,00	10,43
CON8-1	**740,85**	**740,85**	742,02	0,16	0,00	13,60	**740,85**	**740,85**	0,00	0,00	13,60
CON8-2	**712,89**	**712,89**	**712,89**	0,00	0,00	10,05	**712,89**	**712,89**	0,00	0,00	10,05
CON8-3	**811,07**	**811,07**	813,93	0,35	0,00	14,24	**811,07**	**811,07**	0,00	0,00	14,24
CON8-4	**772,25**	**772,25**	**772,25**	0,00	0,00	8,17	**772,25**	**772,25**	0,00	0,00	8,17
CON8-5	**754,88**	**754,88**	756,54	0,22	0,00	11,49	**754,88**	**754,88**	0,00	0,00	11,49
CON8-6	**678,92**	**678,92**	679,98	0,16	0,00	17,08	**678,92**	**678,92**	0,00	0,00	17,08
CON8-7	**811,96**	**811,96**	**811,96**	0,00	0,00	10,03	**811,96**	**811,96**	0,00	0,00	10,03
CON8-8	**767,53**	**767,53**	767,53	0,00	0,00	19,36	**767,53**	**767,53**	0,00	0,00	19,36
CON8-9	**809,00**	**809,00**	810,13	0,19	0,00	16,25	**809,00**	**809,00**	0,00	0,00	16,25
Média	**-**	**-**	**-**	**0,15**	**0,00**	**13,61**	**-**	**-**	**0,00**	**0,00**	**13,61**

alvo, no caso, definido como o melhor encontrado na literatura para o respectivo problema-teste.

Para gerar o gráfico de distribuição de probabilidade em relação ao tempo para encontrar o alvo, foi utilizado o método descrito em (Aiex et al., 2002). Esse método consiste, inicialmente, em ordenar os tempos $T = \{t_1, t_2, \ldots, 100\}$ que os algoritmos encontraram o valor alvo, sendo 100 o número de execuções. Em seguida, calculou-se a probabilidade acumulada $p_i = (i - 1/2)/100$ associada ao i-ésimo tempo de execução. Por fim, os 100 pontos $z_i = (t_i, p_i)$ foram plotados.

Tabela 5.5: Comparação GENILS × GENILS-TS nos problemas-teste de Salhi e Nagy (1999) tendo como critério de parada o tempo de processamento

Prob.	Melhor	GENILS					GENILS-TS				
-	Lit.	Melhor	Média	$Desv^{Avg}$ (%)	$Desv^{Best}$ (%)	Tempo(s) (s)	Melhor	Média	$Desv^{Avg}$ (%)	$Desv^{Best}$ (%)	Tempo(s) (s)
CMT1X	**466,77**	**466,77**	472,23	1,17	0,00	2,25	**466,77**	472,23	1,17	0,00	2,25
CMT1Y	**466,77**	**466,77**	472,22	1,17	0,00	62,20	**466,77**	472,23	1,17	0,00	62,20
CMT2X	**668,77**	684,11	693,94	3,76	2,29	127,97	684,11	693,40	3,68	2,29	127,97
CMT2Y	**663,25**	684,11	693,58	4,57	3,15	132,64	684,11	693,40	4,55	3,15	132,64
CMT3X	**721,27**	**721,27**	726,30	0,70	0,00	265,77	**721,27**	726,34	0,70	0,00	265,77
CMT3Y	**721,27**	**721,27**	730,63	1,30	0,00	256,88	**721,27**	730,56	1,29	0,00	256,88
CMT12X	**644,70**	662,22	673,89	4,53	2,72	269,68	662,22	666,77	3,42	2,72	269,68
CMT12Y	**659,52**	663,50	676,57	2,59	0,60	208,54	662,22	673,91	2,18	0,41	208,54
CMT11X	**833,92**	846,23	875,66	5,01	1,48	416,26	**833,92**	867,96	4,08	0,00	416,26
CMT11Y	**830,39**	836,04	872,64	5,09	0,68	488,47	833,92	856,32	3,12	0,43	488,47
CMT4X	**852,46**	**852,46**	872,68	2,37	0,00	858,95	**852,46**	865,03	1,47	0,00	858,95
CMT4Y	**852,35**	862,28	872,75	2,39	1,17	406,01	855,52	866,06	1,61	0,37	406,01
CMT5X	**1029,25**	1033,51	1059,74	2,96	0,41	1655,14	1030,50	1052,67	2,28	0,12	1655,14
CMT5Y	**1029,25**	1036,14	1056,84	2,68	0,67	790,88	1030,50	1053,73	2,38	0,12	790,88
Média	-	-	-	**2,88**	**0,94**	**424,40**	-	-	**2,36**	**0,69**	**424,40**

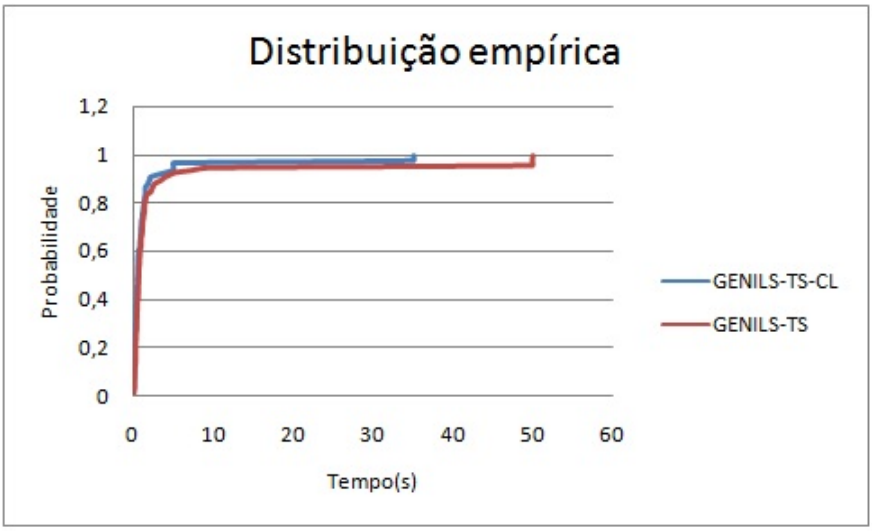

Figura 5.3: Probabilidade acumulada dos algoritmos GENILS-TS e GENILS-TS-CL no problema-teste `CON8-7`

Ao analisar as curvas de probabilidades empíricas pode-se perceber que as curvas de desempenho dos dois algoritmos são bastante semelhantes. Entretanto, observa-se que o GENILS-TS-CL foi o primeiro a alcançar o alvo desejado com uma probabilidade de quase 100% em pouco mais de 35 segundos, enquanto o GENILS-TS levou pouco mais de 50 segundos.

5.3 GENILS-TS-CL-PR

Nesta Seção são mostrados os resultados do algoritmo GENILS-TS-CL-PR, que incorpora o módulo de Reconexão por Caminhos (PR, do inglês *Path Relinking*), conforme descrito na Seção 4.11.

Tabela 5.6: Comparação GENILS × GENILS-TS nos problemas-teste de Montané e Galvão (2006) tendo como critério de parada o tempo de processamento

Prob.	**Melhor**	**GENILS**					**GENILS-TS**				
-	**Lit.**	**Melhor**	**Média**	**Desv**^{Avg} (%)	**Desv**^{Best} (%)	**Tempo(s)** (s)	**Melhor**	**Média**	**Desv**^{Avg} (%)	**Desv**^{Best} (%)	**Tempo(s)** **(s)**
r101	**1009,95**	**1009,95**	1014,98	0,50	0,00	208,56	**1009,95**	1013,71	0,37	0,00	208,56
r201	**666,20**	**666,20**	667,18	0,15	0,00	165,84	**666,20**	666,81	0,09	0,00	165,84
c101	**1220,18**	**1220,18**	1223,93	0,31	0,00	203,19	**1220,18**	1222,81	0,22	0,00	203,19
c201	**662,07**	**662,07**	664,31	0,34	0,00	91,87	**662,07**	664,31	0,34	0,00	91,87
rc101	**1059,32**	**1059,32**	1064,51	0,49	0,00	93,86	**1059,32**	1064,49	0,49	0,00	93,86
rc201	**672,92**	**672,92**	678,63	0,85	0,00	189,50	**672,92**	677,49	0,68	0,00	189,50
r1_2_1	**3357,64**	**3357,64**	3433,83	2,27	0,00	890,19	**3357,64**	3450,31	2,76	0,00	890,19
r2_2_1	**1665,58**	**1665,58**	1684,98	1,16	0,00	1038,79	**1665,58**	1670,35	0,29	0,00	1038,79
c1_2_1	**3629,89**	3636,74	3657,05	0,75	0,19	800,47	3634,65	3657,16	0,75	0,13	800,47
c2_2_1	**1726,59**	**1726,59**	1749,86	1,35	0,00	773,48	**1726,58**	1745,10	1,07	0,00	773,48
rc1_2_1	**3306,00**	3312,92	3339,47	1,01	0,21	977,57	3312,92	3327,22	0,64	0,21	977,57
rc2_2_1	**1560,00**	**1560,00**	1576,58	1,06	0,00	914,61	**1560,00**	1584,84	1,59	0,00	914,61
r1_4_1	**9605,75**	9627,43	9727,90	1,27	0,23	5415,90	9627,88	9706,91	1,05	0,23	5415,90
r2_4_1	**3551,38**	3582,08	3639,14	2,47	0,86	4027,96	3582,08	3612,11	1,71	0,86	4027,96
c1_4_1	**11098,21**	**11098,21**	11144,53	0,42	0,00	3757,66	**11098,21**	11133,59	0,32	0,00	3757,66
c2_4_1	**3546,10**	3596,37	3664,65	3,34	1,42	3293,82	3592,71	3617,86	2,02	1,31	3293,82
rc1_4_1	**9535,46**	**9535,46**	9640,01	1,10	0,00	6004,00	**9535,46**	9638,30	1,08	0,00	6004,00
rc2_4_1	**3403,70**	3422,11	3503,07	2,92	0,54	3600,39	3419,76	3502,01	2,89	0,47	3600,39
Média	**-**	**-**	**-**	**1,21**	**0,19**	**1802,65**	**-**	**-**	**1,02**	**0,18**	**1802,65**

As tabelas 5.10, 5.11 e 5.12 mostram os resultados de sua aplicação nos problemas-teste de Dethloff (2001), Salhi e Nagy (1999) e Montané e Galvão (2006), respectivamente. Nestas tabelas, além das colunas "Prob.", "Melhor Lit.", "Melhor", $Desv^{Avg}$ e $Desv^{Best}$, apresentadas anteriormente, também há duas outras, uma referente à "Mediana" e outra ao "Desvio Padrão" dos resultados encontrados em 30 execuções.

Tabela 5.7: Comparação GENILS-TS × GENILS-TS-CL nos problemas-teste de Dethloff (2001)

Prob.	**Melhor**	**GENILS-TS**					**GENILS-TS-CL**				
-	**Lit. -**	**Melhor**	**Média**	**Desv^{Avg} (%)**	**Desv^{Best} (%)**	**Tempo(s) (s)**	**Melhor**	**Média**	**Desv^{Avg} (%)**	**Desv^{Best} (%)**	**Tempo(s) (s)**
SCA3-0	**635,62**	**635,62**	**635,62**	0,00	0,00	10,32	**635,62**	**635,62**	0,00	0,00	0,75
SCA3-1	**697,84**	**697,84**	**697,84**	0,00	0,00	13,30	**697,84**	**697,84**	0,00	0,00	0,30
SCA3-2	**659,34**	**659,34**	**659,34**	0,00	0,00	9,31	**659,34**	**659,34**	0,00	0,00	0,01
SCA3-3	**680,04**	**680,04**	**680,04**	0,00	0,00	10,29	**680,04**	**680,04**	0,00	0,00	0,29
SCA3-4	**690,50**	**690,50**	**690,50**	0,00	0,00	12,23	**690,50**	**690,50**	0,00	0,00	0,00
SCA3-5	**659,90**	**659,90**	**659,90**	0,00	0,00	15,21	**659,90**	**659,90**	0,00	0,00	0,00
SCA3-6	**651,09**	**651,09**	**651,09**	0,00	0,00	13,41	**651,09**	**651,09**	0,00	0,00	0,00
SCA3-7	**659,17**	**659,17**	**659,17**	0,00	0,00	10,25	**659,17**	**659,17**	0,00	0,00	0,25
SCA3-8	**719,48**	**719,48**	**719,48**	0,00	0,00	21,17	**719,48**	**719,48**	0,00	0,00	0,17
SCA3-9	**681,00**	**681,00**	**681,00**	0,00	0,00	16,17	**681,00**	**681,00**	0,00	0,00	0,17
SCA8-0	**961,50**	**961,50**	**961,50**	0,00	0,00	19,65	**961,50**	**961,50**	0,00	0,00	0,65
SCA8-1	**1049,65**	**1049,65**	**1049,65**	0,00	0,00	11,81	**1049,65**	**1049,65**	0,00	0,00	0,81
SCA8-2	**1039,64**	**1039,64**	**1039,64**	0,00	0,00	9,27	**1039,64**	**1039,64**	0,00	0,00	1,27
SCA8-3	**983,34**	**983,34**	**983,34**	0,00	0,00	15,17	**983,34**	**983,34**	0,00	0,00	0,17
SCA8-4	**1065,49**	**1065,49**	**1065,49**	0,00	0,00	10,26	**1065,49**	**1065,49**	0,00	0,00	0,26
SCA8-5	**1027,08**	**1027,08**	**1027,08**	0,00	0,00	11,21	**1027,08**	**1027,08**	0,00	0,00	0,21
SCA8-6	**971,82**	**971,82**	**971,82**	0,00	0,00	18,13	**971,82**	**971,82**	0,00	0,00	0,13
SCA8-7	**1051,28**	**1051,28**	**1051,28**	0,00	0,00	12,26	**1051,28**	**1051,28**	0,00	0,00	1,26
SCA8-8	**1071,18**	**1071,18**	**1071,18**	0,00	0,00	10,07	**1071,18**	**1071,18**	0,00	0,00	0,07
SCA8-9	**1060,50**	**1060,50**	**1060,50**	0,00	0,00	10,55	**1060,50**	**1060,50**	0,00	0,00	0,55
CON3-0	**616,52**	**616,52**	**616,52**	0,00	0,00	11,49	**616,52**	**616,52**	0,00	0,00	0,49
CON3-1	**554,47**	**554,47**	**554,47**	0,00	0,00	10,36	**554,47**	**554,47**	0,00	0,00	0,36
CON3-2	**518,00**	**518,00**	**518,00**	0,00	0,00	15,11	**518,00**	**518,00**	0,00	0,00	1,11
CON3-3	**591,19**	**591,19**	**591,19**	0,00	0,00	20,30	**591,19**	**591,19**	0,00	0,00	0,30
CON3-4	**588,79**	**588,79**	**588,79**	0,00	0,00	15,62	**588,79**	**588,79**	0,00	0,00	0,62
CON3-5	**563,70**	**563,70**	**563,70**	0,00	0,00	21,70	**563,70**	**563,70**	0,00	0,00	0,70
CON3-6	**499,05**	**499,05**	**499,05**	0,00	0,00	14,39	**499,05**	**499,05**	0,00	0,00	0,39
CON3-7	**576,48**	**576,48**	**576,48**	0,00	0,00	12,51	**576,48**	**576,48**	0,00	0,00	0,51
CON3-8	**523,05**	**523,05**	**523,05**	0,00	0,00	11,10	**523,05**	**523,05**	0,00	0,00	0,10
CON3-9	**578,25**	**578,25**	**578,25**	0,00	0,00	21,25	**578,25**	**578,25**	0,00	0,00	1,25
CON8-0	**857,17**	**857,17**	**857,17**	0,00	0,00	10,43	**857,17**	**857,17**	0,00	0,00	0,43
CON8-1	**740,85**	**740,85**	**740,85**	0,00	0,00	13,60	**740,85**	**740,85**	0,00	0,00	0,60
CON8-2	**712,89**	**712,89**	**712,89**	0,00	0,00	10,05	**712,89**	**712,89**	0,00	0,00	0,05
CON8-3	**811,07**	**811,07**	**811,07**	0,00	0,00	14,24	**811,07**	**811,07**	0,00	0,00	1,24
CON8-4	**772,25**	**772,25**	**772,25**	0,00	0,00	8,17	**772,25**	**772,25**	0,00	0,00	0,17
CON8-5	**754,88**	**754,88**	**754,88**	0,00	0,00	11,49	**754,88**	**754,88**	0,00	0,00	0,49
CON8-6	**678,92**	**678,92**	**678,92**	0,00	0,00	17,08	**678,92**	**678,92**	0,00	0,00	1,08
CON8-7	**811,96**	**811,96**	**811,96**	0,00	0,00	10,03	**811,96**	**811,96**	0,00	0,00	0,03
CON8-8	**767,53**	**767,53**	**767,53**	0,00	0,00	19,36	**767,53**	**767,53**	0,00	0,00	0,36
CON8-9	**809,00**	**809,00**	**809,00**	0,00	0,00	16,25	**809,00**	**809,00**	0,00	0,00	0,25
Média	**-**	**-**	**-**	**0,00**	**0,00**	**13,61**	**-**	**-**	**0,00**	**0,00**	**0,45**

Tabela 5.8: Comparação GENILS-TS × GENILS-TS-CL nos problemas-teste de Salhi e Nagy (1999)

Prob.	Melhor	GENILS-TS					GENILS-TS-CL				
-	Lit. -	Melhor	Média	$Desv^{Avg}$ (%)	$Desv^{Best}$ (%)	Tempo(s) (s)	Melhor	Média	$Desv^{Avg}$ (%)	$Desv^{Best}$ (%)	Tempo(s) (s)
CMT1X	**466,77**	**466,77**	472,23	1,17	0,00	2,25	**466,77**	472,23	1,17	0,00	2,28
CMT1Y	**466,77**	**466,77**	472,23	1,17	0,00	62,20	**466,77**	472,23	1,17	0,00	41,33
CMT2X	**668,77**	684,11	693,40	3,68	2,29	127,97	684,11	694,4	3,83	2,29	92,19
CMT2Y	**663,25**	684,11	693,40	4,55	3,15	132,64	684,11	695,11	4,80	3,15	83,61
CMT3X	**721,27**	**721,27**	726,34	0,70	0,00	265,77	**721,27**	727,06	0,80	0,00	182,26
CMT3Y	**721,27**	**721,27**	730,56	1,29	0,00	256,88	**721,27**	731,32	1,39	0,00	207,21
CMT12X	**644,70**	662,22	666,77	3,42	2,72	269,68	662,22	666,79	3,43	2,72	193,78
CMT12Y	**659,52**	662,22	673,91	2,18	0,41	208,54	662,22	673,91	2,18	0,41	90,19
CMT11X	**833,92**	**833,92**	867,96	4,08	0,00	416,26	**833,92**	867,95	4,08	0,00	211,57
CMT11Y	**830,39**	833,92	856,32	3,12	0,43	488,47	833,92	856,41	3,13	0,43	399,42
CMT4X	**852,46**	**852,46**	865,03	1,47	0,00	858,95	**852,46**	865,28	1,50	0,00	651,83
CMT4Y	**852,35**	855,52	866,06	1,61	0,37	406,01	855,52	866,11	1,61	0,37	376,54
CMT5X	**1029,25**	1030,50	1052,67	2,28	0,12	1655,14	1030,50	1052,71	2,28	0,12	1134,39
CMT5Y	**1029,25**	1030,50	1053,73	2,38	0,12	790,88	1030,50	1053,98	2,40	0,12	699,12
Média	**-**	**-**	**-**	**2,36**	**0,69**	**424,40**	**-**	**-**	**2,41**	**0,69**	**311,84**

Tabela 5.9: Comparação GENILS-TS × GENILS-TS-CL nos problemas-teste de Montané e Galvão (2006)

Prob.	Melhor	GENILS-TS					GENILS-TS-CL				
-	Lit.	Melhor	Média	$Desv^{Avg}$ (%)	$Desv^{Best}$ (%)	Tempo (s)	Melhor	Média	$Desv^{Avg}$ (%)	$Desv^{Best}$ (%)	Tempo(s) (s)
r101	**1009,95**	**1009,95**	1013,71	0,37	0,00	208,56	**1009,95**	1013,74	0,38	0,00	193,02
r201	**666,20**	**666,20**	666,81	0,09	0,00	165,84	**666,20**	666,88	0,10	0,00	131,91
c101	**1220,18**	**1220,18**	1222,81	0,22	0,00	203,19	**1220,18**	1222,93	0,23	0,00	199,36
c201	**662,07**	**662,07**	664,31	0,34	0,00	91,87	**662,07**	665,16	0,47	0,00	87,12
rc101	**1059,32**	**1059,32**	1064,49	0,49	0,00	93,86	**1059,32**	1064,52	0,49	0,00	57,43
rc201	**672,92**	**672,92**	677,49	0,68	0,00	189,50	**672,92**	677,94	0,75	0,00	121,34
r1_2_1	**3357,64**	**3357,64**	3450,31	2,76	0,00	890,19	**3357,64**	3451,13	2,78	0,00	481,39
r2_2_1	**1665,58**	**1665,58**	1670,35	0,29	0,00	1038,79	**1665,58**	1672,02	0,39	0,00	978,25
c1_2_1	**3629,89**	3634,65	3657,16	0,75	0,13	800,47	3634,65	3657,68	0,77	0,13	673,04
c2_2_1	**1726,59**	**1726,58**	1745,10	1,07	0,00	773,48	**1726,58**	1745,95	1,12	0,00	298,32
rc1_2_1	**3306,00**	3312,92	3327,22	0,64	0,21	977,57	3312,92	3329,01	0,70	0,21	783,56
rc2_2_1	**1560,00**	**1560,00**	1584,84	1,59	0,00	914,61	**1560,00**	1584,18	1,55	0,00	914,67
r1_4_1	**9605,75**	9627,88	9706,91	1,05	0,23	5415,90	9627,88	9706,96	1,05	0,23	2745,48
r2_4_1	**3551,38**	3582,08	3612,11	1,71	0,86	4027,96	3582,08	3612,67	1,73	0,86	1930,87
c1_4_1	**11098,21**	**11098,21**	11133,59	0,32	0,00	3757,66	**11098,21**	11134,64	0,33	0,00	3757,66
c2_4_1	**3546,10**	3592,71	3617,86	2,02	1,31	3293,82	3592,71	3618,91	2,05	1,31	2854,53
rc1_4_1	**9535,46**	**9535,46**	9638,30	1,08	0,00	6004,00	**9535,46**	9638,52	1,08	0,00	5411,00
rc2_4_1	**3403,70**	3419,76	3502,01	2,89	0,47	3600,39	3419,76	3502,68	2,91	0,47	3600,33
Média	**-**	**-**	**-**	**1,02**	**0,18**	**1802,65**	**-**	**-**	**1,05**	**0,18**	**1401,07**

Tabela 5.10: Desempenho do algoritmo GENILS-TS-CL-PR nos problemas-teste de Dethloff (2001)

Problema	Melhor Lit	GENILS-TS-CL-PR								GENILS-TS-CL				
		Melhor	Média	Mediana	Desv. Padrão	$Desv^{Avg}$	$Desv^{Best}$	Tempo	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Tempo
SCA3-0	**635,62**	**635,62**	635,62	635,62	0,00	0,00	0,00	0,74	**635,62**	**635,62**	0,00	0,00	0,75
SCA3-1	**697,84**	**697,84**	697,84	697,84	0,00	0,00	0,00	6,80	**697,84**	**697,84**	0,00	0,00	0,30
SCA3-2	**659,34**	**659,34**	659,34	659,34	0,00	0,00	0,00	5,06	**659,34**	**659,34**	0,00	0,00	0,01
SCA3-3	**680,04**	**680,04**	680,04	680,04	0,00	0,00	0,00	0,29	**680,04**	**680,04**	0,00	0,00	0,29
SCA3-4	**690,50**	**690,50**	690,50	690,50	0,00	0,00	0,00	12,02	**690,50**	**690,50**	0,00	0,00	0,00
SCA3-5	**659,90**	**659,90**	659,90	659,90	0,00	0,00	0,00	1,11	**659,90**	**659,90**	0,00	0,00	0,00
SCA3-6	**651,09**	**651,09**	651,09	651,09	0,00	0,00	0,00	1,12	**651,09**	**651,09**	0,00	0,00	0,00
SCA3-7	**659,17**	**659,17**	659,17	659,17	0,00	0,00	0,00	1,47	**659,17**	**659,17**	0,00	0,00	0,25
SCA3-8	**719,48**	**719,48**	719,48	719,48	0,00	0,00	0,00	1,17	**719,48**	**719,48**	0,00	0,00	0,17
SCA3-9	**681,00**	**681,00**	681,00	681,00	0,00	0,00	0,00	5,18	**681,00**	**681,00**	0,00	0,00	0,17
SCA8-0	**961,50**	**961,50**	961,50	961,50	0,00	0,00	0,00	3,65	**961,50**	**961,50**	0,00	0,00	0,65
SCA8-1	**1049,65**	**1049,65**	1049,65	1049,65	0,00	0,00	0,00	3,81	**1049,65**	**1049,65**	0,00	0,00	0,81
SCA8-2	**1039,64**	**1039,64**	1039,64	1039,64	0,00	0,00	0,00	1,27	**1039,64**	**1039,64**	0,00	0,00	1,27
SCA8-3	**983,34**	**983,34**	983,34	983,34	0,00	0,00	0,00	2,20	**983,34**	**983,34**	0,00	0,00	0,17
SCA8-4	**1065,49**	**1065,49**	1065,49	1065,49	0,00	0,00	0,00	3,26	**1065,49**	**1065,49**	0,00	0,00	0,26
SCA8-5	**1027,08**	**1027,08**	1027,08	1027,08	0,00	0,00	0,00	0,21	**1027,08**	**1027,08**	0,00	0,00	0,21
SCA8-6	**971,82**	**971,82**	971,82	971,82	0,00	0,00	0,00	4,13	**971,82**	**971,82**	0,00	0,00	0,13
SCA8-7	**1051,28**	**1051,28**	1051,28	1051,28	0,00	0,00	0,00	3,26	**1051,28**	**1051,28**	0,00	0,00	1,26
SCA8-8	**1071,18**	**1071,18**	1071,18	1071,18	0,00	0,00	0,00	0,75	**1071,18**	**1071,18**	0,00	0,00	0,07
SCA8-9	**1060,50**	**1060,50**	1060,50	1060,50	0,00	0,00	0,00	0,55	**1060,50**	**1060,50**	0,00	0,00	0,55
CON3-0	**616,52**	**616,52**	616,52	616,52	0,00	0,00	0,00	0,49	**616,52**	**616,52**	0,00	0,00	0,49
CON3-1	**554,47**	**554,47**	554,47	554,47	0,00	0,00	0,00	0,84	**554,47**	**554,47**	0,00	0,00	0,36
CON3-2	**518,00**	**518,00**	518,00	518,00	0,00	0,00	0,00	2,11	**518,00**	**518,00**	0,00	0,00	1,11
CON3-3	**591,19**	**591,19**	591,19	591,19	0,00	0,00	0,00	0,30	**591,19**	**591,19**	0,00	0,00	0,30
CON3-4	**588,79**	**588,79**	588,79	588,79	0,00	0,00	0,00	0,62	**588,79**	**588,79**	0,00	0,00	0,62
CON3-5	**563,70**	**563,70**	563,70	563,70	0,00	0,00	0,00	0,70	**563,70**	**563,70**	0,00	0,00	0,70
CON3-6	**499,05**	**499,05**	499,05	499,05	0,00	0,00	0,00	0,39	**499,05**	**499,05**	0,00	0,00	0,39
CON3-7	**576,48**	**576,48**	576,48	576,48	0,00	0,00	0,00	1,51	**576,48**	**576,48**	0,00	0,00	0,51
CON3-8	**523,05**	**523,05**	523,05	523,05	0,00	0,00	0,00	0,14	**523,05**	**523,05**	0,00	0,00	0,10
CON3-9	**578,25**	**578,25**	578,25	578,25	0,00	0,00	0,00	1,25	**578,25**	**578,25**	0,00	0,00	1,25
CON8-0	**857,17**	**857,17**	857,17	857,17	0,00	0,00	0,00	0,55	**857,17**	**857,17**	0,00	0,00	0,43
CON8-1	**740,85**	**740,85**	740,85	740,85	0,00	0,00	0,00	1,60	**740,85**	**740,85**	0,00	0,00	0,60
CON8-2	**712,89**	**712,89**	712,89	712,89	0,00	0,00	0,00	0,52	**712,89**	**712,89**	0,00	0,00	0,05
CON8-3	**811,07**	**811,07**	811,07	811,07	0,00	0,00	0,00	1,24	**811,07**	**811,07**	0,00	0,00	1,24
CON8-4	**772,25**	**772,25**	772,25	772,25	0,00	0,00	0,00	0,31	**772,25**	**772,25**	0,00	0,00	0,17
CON8-5	**754,88**	**754,88**	754,88	754,88	0,00	0,00	0,00	0,49	**754,88**	**754,88**	0,00	0,00	0,49
CON8-6	**678,92**	**678,92**	678,92	678,92	0,00	0,00	0,00	1,53	**678,92**	**678,92**	0,00	0,00	1,08
CON8-7	**811,96**	**811,96**	811,96	811,96	0,00	0,00	0,00	1,62	**811,96**	**811,96**	0,00	0,00	0,03
CON8-8	**767,53**	**767,53**	767,53	767,53	0,00	0,00	0,00	3,36	**767,53**	**767,53**	0,00	0,00	0,36
CON8-9	**809,00**	**809,00**	809,00	809,00	0,00	0,00	0,00	2,47	**809,00**	**809,00**	0,00	0,00	0,25
Média	-	-	-	-	**0,00**	**0,00**	**0,00**	**2,00**	-	-	**0,00**	**0,00**	**0,45**

Tabela 5.11: Desempenho do algoritmo GENILS-TS-CL-PR nos problemas-teste de Salhi e Nagy (1999)

Problema	Melhor Lit	GENILS-TS-CL-PR							GENILS-TS-CL				
		Melhor	Média	Mediana	Desv. Padrão	$Desv^{Avg}$	$Desv^{Best}$	Tempo	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Tempo
CMT1X	**466,77**	**466,77**	471,08	472,37	4,40	0,92	0,00	7,28	**466,77**	472,23	1,17	0,00	2,28
CMT1Y	**466,77**	**466,77**	472,01	471,53	6,62	1,12	0,00	41,33	**466,77**	472,23	1,17	0,00	41,33
CMT2X	**668,77**	684,11	688,40	696,12	7,28	0,63	2,29	112,03	684,11	694,4	3,83	2,29	92,19
CMT2Y	**663,25**	684,11	695,11	691,67	8,07	1,61	3,15	92,15	684,11	695,11	4,80	3,15	83,61
CMT3X	**721,27**	**721,27**	727,06	726,80	8,67	0,80	0,00	213,87	**721,27**	727,06	0,80	0,00	182,26
CMT3Y	**721,27**	**721,27**	731,32	732,95	7,45	1,39	0,00	207,21	**721,27**	731,32	1,39	0,00	207,21
CMT12X	**644,70**	662,22	666,79	662,23	6,61	0,69	2,72	191,56	662,22	666,79	3,43	2,72	193,78
CMT12Y	**659,52**	662,22	673,91	680,61	7,86	1,76	0,41	90,19	662,22	673,91	2,18	0,41	90,19
CMT11X	**833,92**	**833,92**	867,95	878,91	12,07	4,08	0,00	318,81	**833,92**	867,95	4,08	0,00	211,57
CMT11Y	**830,39**	833,92	856,41	953,70	12,63	2,70	0,43	618,36	833,92	856,41	3,13	0,43	399,42
CMT4X	**852,46**	**852,46**	865,28	864,05	9,69	1,50	0,00	698,12	**852,46**	865,28	1,50	0,00	651,83
CMT4Y	**852,35**	855,52	866,11	878,15	11,51	1,24	0,37	376,54	855,52	866,11	1,61	0,37	376,54
CMT5X	**1029,25**	1030,50	1052,71	1054,41	8,63	2,16	0,12	1136,65	1030,50	1052,71	2,28	0,12	1134,39
CMT5Y	**1029,25**	1030,50	1053,98	1054,57	8,63	2,28	0,12	708,21	1030,50	1053,98	2,40	0,12	699,12
Média	**-**	**-**	**-**	**-**	**8,58**	**1,63**	**0,69**	**343,74**	**-**	**-**	**2,41**	**0,69**	**311,84**

Tabela 5.12: Desempenho do algoritmo GENILS-TS-CL-PR nos problemas-teste de Montané e Galvão (2006)

Problema	Melhor Lit	GENILS-TS-CL-PR								GENILS-TS-CL				
		Melhor	Média	Mediana	Desv. Padrão	$Desv^{Avg}$	$Desv^{Best}$	Tempo	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Tempo
r101	**1009,95**	**1009,95**	1012,71	1013,79	4,05	0,27	0,00	199,98	**1009,95**	1013,74	0,38	0,00	193,02
r201	**666,20**	**666,20**	666,81	668,28	2,42	0,09	0,00	182,67	**666,20**	666,88	0,10	0,00	131,91
c101	**1220,18**	**1220,18**	1222,81	1223,49	5,16	0,22	0,00	234,43	**1220,18**	1222,93	0,23	0,00	199,36
c201	**662,07**	**662,07**	664,31	662,07	3,07	0,34	0,00	96,31	**662,07**	665,16	0,47	0,00	87,12
rc101	**1059,32**	**1059,32**	1062,97	1059,32	9,35	0,34	0,00	88,12	**1059,32**	1064,52	0,49	0,00	57,43
rc201	**672,92**	**672,92**	677,49	677,21	3,86	0,68	0,00	213,67	**672,92**	677,94	0,75	0,00	121,34
r1_2_1	**3357,64**	**3357,64**	3459,12	3453,30	7,24	3,02	0,00	610,83	**3357,64**	3451,13	2,78	0,00	481,39
r2_2_1	**1665,58**	**1665,58**	1667,44	1670,46	3,17	0,11	0,00	1087,32	**1665,58**	1672,02	0,39	0,00	978,25
c1_2_1	**3629,89**	3634,65	3652,16	3651,00	11,94	0,48	0,13	789,04	3634,65	3657,68	0,77	0,13	673,04
c2_2_1	**1726,59**	**1726,58**	1745,10	1741,23	6,32	1,07	0,00	854,12	**1726,58**	1745,95	1,12	0,00	298,32
rc1_2_1	**3306,00**	3312,92	3327,22	3326,12	5,54	0,43	0,21	872,98	3312,92	3329,01	0,70	0,21	783,56
rc2_2_1	**1560,00**	**1560,00**	1584,84	1576,27	6,77	1,59	0,00	109,54	**1560,00**	1584,18	1,55	0,00	914,67
r1_4_1	**9605,75**	9627,88	9706,91	9710,47	5,79	0,82	0,23	2897,99	9627,88	9706,96	1,05	0,23	2745,48
r2_4_1	**3551,38**	3582,08	3599,11	3602,89	7,54	0,48	0,86	2160,21	3582,08	3612,67	1,73	0,86	1930,87
c1_4_1	**11098,21**	**11098,21**	11133,59	11126,02	9,29	0,32	0,00	3876,82	**11098,21**	11134,64	0,33	0,00	3757,66
c2_4_1	**3546,10**	3592,71	3617,86	3604,88	9,97	0,70	1,31	2790,21	3592,71	3618,91	2,05	1,31	2854,53
rc1_4_1	**9535,46**	**9535,46**	9638,30	9651,37	8,62	1,08	0,00	5789,21	**9535,46**	9638,52	1,08	0,00	5411,00
rc2_4_1	**3403,70**	3419,76	3477,01	3476,56	2,94	1,67	0,47	3976,76	3419,76	3502,68	2,91	0,47	3600,33
Média	**-**	**-**	**-**	**-**	**6,28**	**0,76**	**0,18**	**1490,57**	**-**	**-**	**1,05**	**0,18**	**1401,07**

Comparando os resultados obtidos com os apresentados anteriormente pelas outras versões do algoritmo, constata-se que o GENILS-TS-CL-PR alcançou os mesmos melhores resultados que os demais, porém houve melhora nas médias e, consequentemente, nos $Desv^{Avg}$. Observa-se, também, que houve uma pequena piora no tempo quando comparado com o GENILS-TS-CL. Entretanto, quando comparado com o GENILS-TS, os tempos apresentados se mostram significativamente menores.

Para mostrar a robustez do algoritmo, foram analisados as medianas e o desvio padrão dos resultados obtidos em todas as execuções do algoritmo. Nota-se, através destes parâmetros, que o algoritmo produz soluções finais com pouca variabilidade, já que tem valores de medianas próximas ao melhor valor encontrado e desvio padrão em média inferior a 8,58. Mostra-se, assim, que todas as vezes que o GENILS-TS-CL-PR é executado, o valor encontrado terá uma boa qualidade.

5.4 GENILS-TS-CL-PR× algoritmos da literatura

Nesta Seção são mostrados os resultados da comparação entre o algoritmo GENILS-TS-CL-PR e outros três algoritmos da literatura: o algoritmo evolutivo de Zachariadis et al. (2010), o algoritmo ILS-RVND paralelo de Subramanian et al. (2010) (que usa 256 núcleos de processamento) e o GENILS de Mine et al. (2010). Destaca-se que os algoritmos de Zachariadis et al. (2010) e Mine et al. (2010) são, de nosso conhecimento, os melhores algoritmos sequenciais conhecidos, enquanto o algoritmo de Subramanian et al. (2010) é o melhor algoritmo encontrado para o PRVCES.

As Tabelas 5.13, 5.14 e 5.15 apresentam os melhores resultados encontrados pelo GENILS-TS-CL-PR e os comparam com os melhores resultados 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 conhecido na literatura.

Como pode ser observado na Tabela 5.13, o algoritmo GENILS-TS-CL-PR foi capaz de alcançar as melhores soluções conhecidas em todos os problemas-teste de Dethloff (2001).

Analisando-se a Tabela 5.14, que contém os resultados nos problemas-teste de Salhi e 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%. 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 5.15, em que são apresentados os resultados nos problemas-teste de (Montané e 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

Tabela 5.13: GENILS-TS-CL-PR × melhores algoritmos da literatura nos problemas-teste de Dethloff (2001)

Prob.	Melhor	Zachariadis *et al.* (2010)		Subramanian *et al.* (2010)		Mine *et al.*(2010)		GENILS-TS-CL-PR	
-	Lit.	**Melhor**	**Desv^{Best}** (%)	**Melhor**	**Desv^{Best}** (%)	**Melhor**	**Desv^{Best}** (%)	**Melhor**	**Desv^{Best}** (%)
SCA3-0	**635,62**	**635,62**	0,00	**635,62**	0,00	**635,62**	0,00	**635,62**	0,00
SCA3-1	**697,84**	**697,84**	0,00	**697,84**	0,00	**697,84**	0,00	**697,84**	0,00
SCA3-2	**659,34**	**659,34**	0,00	**659,34**	0,00	**659,34**	0,00	**659,34**	0,00
SCA3-3	**680,04**	**680,04**	0,00	**680,04**	0,00	**680,04**	0,00	**680,04**	0,00
SCA3-4	**690,50**	**690,50**	0,00	**690,50**	0,00	**690,50**	0,00	**690,50**	0,00
SCA3-5	**659,90**	**659,90**	0,00	**659,90**	0,00	**659,90**	0,00	**659,90**	0,00
SCA3-6	**651,09**	**651,09**	0,00	**651,09**	0,00	**651,09**	0,00	**651,09**	0,00
SCA3-7	**659,17**	**659,17**	0,00	**659,17**	0,00	**659,17**	0,00	**659,17**	0,00
SCA3-8	**719,48**	**719,48**	0,00	**719,48**	0,00	**719,48**	0,00	**719,48**	0,00
SCA3-9	**681,00**	**681,00**	0,00	**681,00**	0,00	**681,00**	0,00	**681,00**	0,00
SCA8-0	**961,50**	**961,50**	0,00	**961,50**	0,00	**961,50**	0,00	**961,50**	0,00
SCA8-1	**1049,65**	**1049,65**	0,00	**1049,65**	0,00	**1049,65**	0,00	**1049,65**	0,00
SCA8-2	**1039,64**	**1039,64**	0,00	**1039,64**	0,00	**1039,64**	0,00	**1039,64**	0,00
SCA8-3	**983,34**	**983,34**	0,00	**983,34**	0,00	**983,34**	0,00	**983,34**	0,00
SCA8-4	**1065,49**	**1065,49**	0,00	**1065,49**	0,00	**1065,49**	0,00	**1065,49**	0,00
SCA8-5	**1027,08**	**1027,08**	0,00	**1027,08**	0,00	**1027,08**	0,00	**1027,08**	0,00
SCA8-6	**971,82**	**971,82**	0,00	**971,82**	0,00	**971,82**	0,00	**971,82**	0,00
SCA8-7	**1051,28**	**1051,28**	0,00	**1051,28**	0,00	**1051,28**	0,00	**1051,28**	0,00
SCA8-8	**1071,18**	**1071,18**	0,00	**1071,18**	0,00	**1071,18**	0,00	**1071,18**	0,00
SCA8-9	**1060,50**	**1060,50**	0,00	**1060,50**	0,00	**1060,50**	0,00	**1060,50**	0,00
CON3-0	**616,52**	**616,52**	0,00	**616,52**	0,00	**616,52**	0,00	**616,52**	0,00
CON3-1	**554,47**	**554,47**	0,00	**554,47**	0,00	**554,47**	0,00	**554,47**	0,00
CON3-2	**518,00**	**518,00**	0,00	**518,00**	0,00	**518,00**	0,00	**518,00**	0,00
CON3-3	**591,19**	**591,19**	0,00	**591,19**	0,00	**591,19**	0,00	**591,19**	0,00
CON3-4	**588,79**	**588,79**	0,00	**588,79**	0,00	**588,79**	0,00	**588,79**	0,00
CON3-5	**563,70**	**563,70**	0,00	**563,70**	0,00	**563,70**	0,00	**563,70**	0,00
CON3-6	**499,05**	**499,05**	0,00	**499,05**	0,00	**499,05**	0,00	**499,05**	0,00
CON3-7	**576,48**	**576,48**	0,00	**576,48**	0,00	**576,48**	0,00	**576,48**	0,00
CON3-8	**523,05**	**523,05**	0,00	**523,05**	0,00	**523,05**	0,00	**523,05**	0,00
CON3-9	**578,25**	**578,25**	0,00	**578,25**	0,00	**578,25**	0,00	**578,25**	0,00
CON8-0	**857,17**	**857,17**	0,00	**857,17**	0,00	**857,17**	0,00	**857,17**	0,00
CON8-1	**740,85**	**740,85**	0,00	**740,85**	0,00	**740,85**	0,00	**740,85**	0,00
CON8-2	**712,89**	**712,89**	0,00	**712,89**	0,00	**712,89**	0,00	**712,89**	0,00
CON8-3	**811,07**	**811,07**	0,00	**811,07**	0,00	**811,07**	0,00	**811,07**	0,00
CON8-4	**772,25**	**772,25**	0,00	**772,25**	0,00	**772,25**	0,00	**772,25**	0,00
CON8-5	**754,88**	**754,88**	0,00	**754,88**	0,00	**754,88**	0,00	**754,88**	0,00
CON8-6	**678,92**	**678,92**	0,00	**678,92**	0,00	**678,92**	0,00	**678,92**	0,00
CON8-7	**811,96**	**811,96**	0,00	**811,96**	0,00	**811,96**	0,00	**811,96**	0,00
CON8-8	**767,53**	**767,53**	0,00	**767,53**	0,00	**767,53**	0,00	**767,53**	0,00
CON8-9	**809,00**	**809,00**	0,00	**809,00**	0,00	**809,00**	0,00	**809,00**	0,00
Média	**-**	-	0,00	-	0,00	-	0,00	-	0,00

e variabilidade das soluções, supera 6 resultados 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 5.14: GENILS-TS-CL-PR × melhores algoritmos da literatura nos problemas-teste de Salhi e Nagy (1999)

Prob.	Melhor	Zachariadis *et al.* (2010)		Subramanian *et al.* (2010)		Mine *et al.*(2010)		GENILS-TS-CL-PR	
-	Lit.	**Melhor**	**Desv**^{Best} (%)	**Melhor**	**Desv**^{Best} (%)	**Melhor**	**Desv**^{Best} (%)	**Melhor**	**Desv**^{Best} (%)
CMT1X	**466,77**	469,8	0,65	**466,77**	0,00	**466,77**	0,00	**466,77**	0,00
CMT1Y	**466,77**	469,8	0,65	**466,77**	0,00	**466,77**	0,00	**466,77**	0,00
CMT2X	**668,77**	684,21	2,31	684,21	2,31	684,11	2,29	684,11	2,29
CMT2Y	**663,25**	684,21	3,16	684,21	3,16	684,11	3,15	684,11	3,15
CMT3X	**721,27**	**721,27**	0,00	**721,27**	0,00	**721,40**	0,02	**721,27**	0,00
CMT3Y	**721,27**	**721,27**	0,00	**721,27**	0,00	**721,27**	0,00	**721,27**	0,00
CMT12X	**644,7**	662,22	2,72	662,22	2,72	662,22	2,72	662,22	2,72
CMT12Y	**659,52**	662,22	0,41	662,22	0,41	663,50	0,60	662,22	0,41
CMT11X	**833,92**	**833,92**	0,00	**833,92**	0,00	846,23	1,48	**833,92**	0,00
CMT11Y	**830,39**	833,92	0,43	833,92	0,43	836,04	0,68	833,92	0,43
CMT4X	**852,46**	**852,46**	0,00	**852,46**	0,00	**852,46**	0,00	**852,46**	0,00
CMT4Y	**852,35**	852,46	0,01	**852,46**	0,01	862,28	1,17	855,52	0,37
CMT5X	**1029,25**	1030,55	0,13	**1029,25**	0,00	1033,51	0,41	1030,5	0,12
CMT5Y	**1029,25**	1030,55	0,13	**1029,25**	0,00	1036,14	0,67	1030,5	0,12
Média	**-**	-	0,76	-	0,65	-	0,94	-	0,69

Tabela 5.15: GENILS-TS-CL-PR × melhores algoritmos da literatura nos problemas-teste de Montané e Galvão (2006)

Prob.	Melhor	Zachariadis *et al.* (2010)		Subramanian *et al.* (2010)		Mine *et al.*(2010)		GENILS-TS-CL-PR	
-	Lit.	**Melhor**	**Desv**^{Best} (%)	**Melhor**	**Desv**^{Best} (%)	**Melhor**	**Desv**^{Best} (%)	**Melhor**	**Desv**^{Best} (%)
r101	**1009,95**	**1009,95**	0,00	**1009,95**	0,00	**1009,95**	0,00	**1009,95**	0,00
r201	**666,20**	**666,20**	0,00	**666,20**	0,00	**666,20**	0,00	**666,20**	0,00
c101	**1220,18**	1220,99	0,07	**1220,18**	0,00	**1220,18**	0,00	**1220,18**	0,00
c201	**662,07**	**662,07**	0,00	**662,07**	0,00	**662,07**	0,00	**662,07**	0,00
rc101	**1059,32**	**1059,32**	0,00	**1059,32**	0,00	**1059,32**	0,00	**1059,32**	0,00
rc201	**672,92**	**672,92**	0,00	**672,92**	0,00	**672,92**	0,00	**672,92**	0,00
r1_2_1	**3357,64**	3376,30	0,55	3360,02	0,07	**3357,64**	0,00	**3357,64**	0,00
r2_2_1	**1665,58**	**1665,58**	0,00	**1665,58**	0,00	**1665,58**	0,00	**1665,58**	0,00
c1_2_1	**3629,89**	3643,82	0,38	**3629,89**	0,00	3636,74	0,19	3634,65	0,13
c2_2_1	**1726,59**	**1726,59**	0,00	**1726,59**	0,00	**1726,59**	0,00	**1726,58**	0,00
rc1_2_1	**3306,00**	3323,56	0,53	**3306,00**	0,00	3312,92	0,21	3312,92	0,21
rc2_2_1	**1560,00**	**1560,00**	0,00	**1560,00**	0,00	**1560,00**	0,00	**1560,00**	0,00
r1_4_1	**9605,75**	9691,60	0,89	**9605,75**	0,00	9627,43	0,23	9627,88	0,23
r2_4_1	**3551,38**	3572,38	0,59	**3551,38**	0,00	3582,08	0,86	3582,08	0,89
c1_4_1	**11098,21**	11179,36	0,73	11099,54	0,01	**11098,21**	0,00	**11098,21**	0,00
c2_4_1	**3546,10**	3549,27	0,09	**3546,10**	0,00	3596,37	1,42	3592,71	1,31
rc1_4_1	**9535,46**	9645,27	1,14	9536,77	0,01	**9535,46**	0,00	**9535,46**	0,00
rc2_4_1	**3403,70**	3423,62	0,58	**3403,70**	0,00	3422,11	0,54	3419,76	0,47
Média	**-**	-	0,31	-	0,01	-	0,19	-	0,18

Capítulo 6

Conclusões

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.

O PRVCES é um problema da classe NP-difícil, fazendo com que seja difícil resolver instâncias maiores na otimalidade em tempos computacionais aceitáveis. Devido a este fato, abordagens heurísticas são mais utilizadas para resolvê-lo. Sendo assim, este trabalho se baseou em uma importante referência na literatura, proposta em Mine et al. (2010), que desenvolveram uma heurística denominada GENILS para a resolução do PRVCES. O algoritmo GENILS combina os procedimentos heurísticos Inserção Mais Barata, Inserção Mais Barata Múltiplas Rotas, GENIUS, *Iterated Local Search* (ILS) e *Variable Neighborhood Descent* (VND). Os três primeiros 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, 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, 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 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%. Como ponto negativo, o GENILS-TS exige um tempo computacional muito maior, quando o critério de parada é o número máximo de iterações sem melhora do ILS.

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 de Subramanian et al. (2010), Mine et al. (2010): 57 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 de melhor desempenho foi o ILS-RVND paralelo de Subramanian et al. (2010). Entretanto, esse algoritmo usa 256 núcleos de processamento para explorar o espaço de soluções, enquanto os demais usam apenas um. Tendo em vista a estrutura computacional sofisticada requerida para aplicação desse algoritmo, 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.

Como trabalhos futuros, sugere-se estudar outras estratégias de Lista de Candidatos, que excluam do espaço de busca um maior de soluções sem prejuízo para a obtenção da solução ótima, assim como paralelizar o algoritmo, aproveitando os núcleos de processamento existentes nas máquinas atuais.

Referências Bibliográficas

Aiex, R. M.; Resende, M. G. C. e Ribeiro, C. C. (2002). Probability distribution of solution time in grasp: An experimental investigation. *Journal of Heuristics*, v. 8, n. 3, p. 343–373. ISSN 1381-1231. doi: http://dx.doi.org/10.1023/A:1015061802659.

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

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

Belfiore, P. P. e Yoshizaki, H. T. Y. (2006). Scatter search para problemas de roteirização de veículos com frota heterogênea, janelas de tempo e entregas fracionadas. *Produção*, v. 16, p. 455– 469.

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

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

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

Choi, E. e Tcha, D.-W. (2007). (2007). A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers and Operations Research*, v. 34(7), p. 2080–2095.

Cordeau, J-F; Laporte, G. e Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *The Journal of the Operational Research Society*, v. 52, p. 52, No. 8.

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

Dantzig, George B. e Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, v. 6, p. 80–91.

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

Dorigo, Marco; Maniezzo, Vittorio e Colorni, Alberto. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, v. 26, p. 29–41.

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

Gendreau, M.; Hertz, A. e Laporte, G. (1992). New insertion and post optimization procedures or the traveling salesman problem. *Operations Research*, v. 40, p. 1086–1094.

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, F. (1996). Tabu search and adaptive memory programming - advances, applications and challenges. *Interfaces in Computer Sciences and Operations Research*, p. 1–75.

Glover, Fred e Laguna, Manuel. (1997). *Tabu Search*. Kluwer Academic Publisher, Boston.

Goldberg, David E. January(1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional. ISBN 0201157675. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0201157675`.

Golden, B.; Assad, A.; Levy, L. e Gheysens, F. (1984). The fleet size and mix vehicle routing. *Computers & Operations Research*, v. 11, p. 49–66.

Golden, B.L.; Assad, A. A. e Wasil, E. A. (2002). Routing vehicles in the real world: Applications in the solid waste, beverage, food, dairy, and newspaper industries. *SIAM Monographs on Discrete Mathematics and Applications 9*, p. 245–286.

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

Hansen, Pierre e Mladenović, Nenad. May(2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, v. 130, n. 3, p. 449–467.

Kirkpatrick, S.; Gelatt, C. D. e Vecchi, M. P. May(1983). Optimization by simulated annealing. *Science*, v. 4598, n. 13, p. 671–680. URL `citeseer.ist.psu.edu/kirkpatrick83optimization.html`.

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

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

Mladenović, N. e Hansen, P. (1997). Variable neighboorhood search. *Computers and Operations Research*, v. 24, p. 1097–1100.

Montané, Fermín Alfredo Tang e Galvão, Roberto Diéguez. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers and Operations Research*, v. 33, n. 3, p. 595–619. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2004.07.009.

Montgomery, D. (2007). *Design and Analysis of Experiments*. John Wiley & Sons, New York, NY, fifth edição.

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

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

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

Ribeiro, C. C.; Uchoa, E. e Werneck, R. F. (2002). A hybrid grasp with perturbations for the steiner problem in graphs. *INFORMS Journal on Computing*, v. 14, p. 228–246.

Röpke, S. e Pisinger, D. (2006). A unified heuristic for a large class of vehicle routing problems with backhauls. Relatório Técnico 2004/14, University of Copenhagen.

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

Shaw, Paul. (1998). Using constraint programming and local search methods to solve vehicle routing problems. *CP '98: Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*, p. 417–431, London, UK. Springer-Verlag. ISBN 3-540-65224-8.

Steiglitz, K. e Weiner, P. (1968). Some improved algorithms for computer solution of the traveling salesman problem. *Proceedings of the Sixth Allerton Conference on Circuit Theory*, p. 814–821, Monticello, EUA.

Stützle, Thomas e Hoos, Holger H. (1999). Analyzing the run-time behaviour of iterated local search for the tsp. *Proceeding of the Third Metaheuristics International Conference*, p. 449–453, Angra dos Reis, Rio de Janeiro.

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

Subramanian, A.; Uchoa, E.; A.Pessoa, e Ochi, L. (2011). Branch-and-cut with lazy separation for the vehicle routing problem with simultaneous pickup and delivery. *Operations Research Letters*, v. 39, p. 338–341.

Subramanian, Anand. (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, UFPB, João Pessoa.

Subramanian, Anand; Cabral, Lucídio A. F. e Ochi, Luiz Satoru. (2008). An efficient ils heuristic for the vehicle routing problem with simultaneous pickup and delivery. Relatório Técnico 07/2008, Universidade Federal Fluminense. Disponível em http://www.ic.uff.br/PosGraduacao/RelTecnicos/401.pdf.

Topcuoglu, Haluk e Sevilmis, Can. (2002). Task scheduling with conflicting objectives. Yakhno, Tatyana M., editor, *ADVIS*, volume 2457 of *Lecture Notes in Computer Science*, p. 346–355. Springer, (2002). ISBN 3-5... URL `http://dblp.uni-trier.de/db/conf/advis/advis2002.html#TopcuogluS02`.

Toth, P. e Vigo, D. (2002). VRP with backhauls. *SIAM Monographs on Discrete Mathematics and Applications*, v. 9, p. 195–224.

Voudouris, Chris e Tsang, Edward. (1996). Partial constraint satisfaction problems and guided local search. *In The Second International Conference on the Practical Application of Constraint Technology (PACT'96)*, p. 337–356, London, UK.

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, N. A.; Wassan, A. H. e Nagy, G. (2007). A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimization*, v. 15, n. 4, p. 368–386.

Zachariadis, Emmanouil E.; Tarantilis, Christos D. e Kiranoudis, Chris T. (2009). A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications*, v. 36, n. 2, p. 1070–1081. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2007.11.005.

Zachariadis, Emmanouil E.; Tarantilis, Christos D. e Kiranoudis, Chris T. (2010). An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. *European Journal of Operational Research*, v. 202, p. 401–411.

Apêndice A

Produtos

São listados, a seguir, os trabalhos oriundos desta pesquisa que foram publicados em anais de eventos científicos e periódicos.

Trabalhos publicados em periódicos

Título: O problema de roteamento de veículos com coleta e entrega simultânea: uma abordagem via Iterated Local Search e GENIUS
Autores: Marcone Jamilson Freitas Souza, Marcio Tadayuki Mine, Matheus de Souza Alves Silva, Luiz Satoru Ochi, Thais Cotta Barbosa da Silva
Periódico: Transportes (Rio de Janeiro)
ISSN: 2237-1346
Volume: 18 **Páginas**: 60-71
Ano: 2010

Trabalhos apresentados em eventos nacionais

Título: Um Algoritmo Heurístico para Resolução do Problema de Roteamento de Veículos com Coleta e Entrega Simultânea
Co-autores: Thais Cotta Barbosa da Silva, Raphael Carlos Cruz, Marcio Tadayuki Mine, Marcone Jamilson Freitas Souza, Ernesto Del Rosario Santibanez
Evento: XLIII Simpósio Brasileiro de Pesquisa Operacional (XLIII SBPO)
Local: Ubatuba (SP)
Período: 15 a 18 de Agosto de 2011

Título: GENILS-TS: Um algoritmo heurístico para resolução do problema de roteamento de veículos com coleta e entrega simultânea
Co-autores: Thais Cotta Barbosa da Silva, Raphael Carlos Cruz, Marcio Tadayuki Mine, Marcone Jamilson Freitas Souza, Ernesto Del Rosario Santibanez
Evento: Simpósio de Logística e Pesquisa Operacional da Marinha (SPOLM 2011)
Local: Rio de Janeiro (RJ)
Período: 15 a 16 de Setembro de 2011