UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS CRATEÚS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FRANCISCO ANTONIO FERREIRA DE ALMEIDA

UMA ABORDAGEM GGVNS APLICADA A RESOLUÇÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEA

CRATEÚS, CEARÁ

2018

FRANCISCO ANTONIO FERREIRA DE ALMEIDA

UMA ABORDAGEM GGVNS APLICADA A RESOLUÇÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Crateús da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientadora: Prof. Msc. Lisieux Marie Marinho dos Santos Andrade

CRATEÚS, CEARÁ

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A446a Almeida, Francisco Antonio Ferreira de.
Uma abordagem GGVNS aplicada a resolução do Problema de Roteamento de Veículos com Coleta e Entrega Simultânea / Francisco Antonio Ferreira de Almeida. – 2018.
61 f. : il.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Crateús, Curso de Ciência da Computação, Crateús, 2018.
Orientação: Profa. Ma. Lisieux Marie Marinho dos Santos Andrade.

1. Otimização Computacional. 2. Problema de Roteamento de Veículos com Coleta e Entrega Simultânea. 3. Greedy Randomized Adaptive Search Procedure. 4. Variable Neighborhood Search. 5. Variable Neighborhood Descent. I. Título.

CDD 004

FRANCISCO ANTONIO FERREIRA DE ALMEIDA

UMA ABORDAGEM GGVNS APLICADA A RESOLUÇÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Crateús da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Msc. Lisieux Marie Marinho dos Santos Andrade (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Msc. Livio Antônio Melo Freire
Universidade Federal do Ceará (UFC)

Prof. Msc. Renê Pereira de Gusmão
Universidade Federal de Sergipe (UFS)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Pai, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir.

AGRADECIMENTOS

À Profa. Msc. Lisieux Marie Marinho dos Santos por sua preciosa orientação e ter me dado forças pra seguir em frente de cabeça erguida.

Aos meus pais, irmãos, que nos momentos de minha ausência dedicados ao estudo superior, sempre fizeram entender que o futuro é feito a partir da contínua dedicação no presente!

Aos professores participantes da banca examinadora pelo tempo, pelas valiosas colaborações e sugestões.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

Aos colegas de turma que acreditaram em mim.

Aos meus amigos por ter me dado esperança para superar as dificuldades.

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

A todos o meu muito obrigado!

"O sonho é que leva a gente para frente. Se a gente for seguir a razão, fica aquietado, acomodado."

(Ariano Suassuna)

RESUMO

No cenário dos processos de coleta e entrega simultânea de produtos e pessoas, uma grande preocupação industrial é o estabelecimento do uso mínimo de recursos e da distância total percorrida pelos veículos. Na área da Otimização Computacional, pesquisas classificam o referido cenário como um Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Dada a complexidade de resolução deste problema e a necessidade de obter resultados melhores do que os existentes na literatura, este trabalho propõe o uso de uma estratégia híbrida heurística composta por *Greedy Adaptive Randomized Search Procedure* + *Variable Neighborhood Search* + *Variable Neighborhood Descent*(GGVNS). O algoritmo foi testado em instâncias consagradas na literatura e se mostrou eficiente para a maioria destas.

Palavras-chave: Otimização Computacional. Problema de Roteamento de Veículos com Coleta e Entrega Simultânea. *Greedy Randomized Adaptive Search Procedure*. *Variable Neighborhood Search*. *Variable Neighborhood Descent* .

ABSTRACT

In the scenario of the processes of pickup and delivery simultaneous of products and people, a major industrial concern is the establishment of the minimum use of resources and the total distance traveled by the vehicles. In the area of Computational Optimization, surveys classify this scenario as a Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD). Given the complexity of solving this problem and the need to obtain better results than existent in the literature, this work proposes the use of a hybrid heuristic strategy composed by Greedy Adaptive Randomized Search Procedure + Variable Neighborhood Search + Variable Neighborhood Descent (GGVNS). The algorithm was tested in consecrated instances in the literature and proved to be efficient for most of these.

Keywords: Computational Optimization. Vehicle Routing with Simultaneous Pickup and Delivery Problem. Greedy Randomized Adaptive Search Procedure. Variable Neighborhood Search. Variable Neighborhood Descent.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O Problema de Roteamento de Veículos(PRV) exerce função de grande importância na cadeia de abastecimento de várias empresas comprometidas com o transporte de bens ou pessoas. A relevância do estudo deste problema é motivada pela grande quantidade de aplicações em situações reais encaradas diariamente no ambiente empresarial.

Toth e Vigo (2002) afirmam que os custos de transporte podem ser reduzidos na ordem de 5% a 20% com a aplicação de técnicas computacionais para a sua resolução. Tais reduções de custos podem ser evidenciadas em diversos estudos de caso apresentados por Golden *et al.* (2002) e Baker (2002). Formalmente, o PRV foi apresentado inicialmente por Dantzig e Ramser (1959) no final dos anos 50 ao tratar o problema de distribuição de gasolina para postos de combustível.

Dentre as variantes do PRV, tem-se o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea(PRVCES), objeto de estudo deste trabalho. Este problema foi proposto primeiramente por Min (1989), um problema clássico da logística reversa a qual um conjunto de procedimentos são utilizados para possibilitar a coleta e entrega de resíduos sólidos do setor empresarial, possuindo grande aplicabilidade na indústria, principalmente em distribuição de bebidas, logística postal, coleta de lixo, entre outras.

Empresas vêm procurando formas eficientes de manipular o retorno de materiais após sua venda e consumo, para um posterior reaproveitamento de suas embalagens. As vantagens dessa prática são: redução de custo, redução dos danos ao meio ambiente e garantia do direito do consumidor de devolução e troca de mercadorias, entre outras. A busca de um procedimento eficiente para definir tais rotas viáveis e otimizadas que satisfaçam as demandas dos consumidores pode reduzir significativamente o custo das grandes empresas e grandes indústrias, além de ser um grande desafio para os estudiosos já que o problema pertence à classe NP-difícil (DETHLOFF, 2001), sendo que o problema pode ser reduzido ao PRV clássico quando nenhum cliente necessita de demanda de coleta. Sabendo da complexidade deste problema, convém implementar algoritmos eficientes para sua resolução em tempo aceitável.

Neste trabalho, propõe-se resolver o PRVCES através da meta-heurística GGVNS que combina o *Greedy Adaptive Randomized Search Procedure*(GRASP), *Variable Neighborhood Search*(VNS) e *Variable Neighborhood Descent*(VND), com o intuito de obter resultados mais competitivos do que os existentes na literatura.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver um procedimento heurístico aplicado ao Problema de Roteamento de Veículos com Coleta e Entrega Simultânea.

1.1.2 Objetivos Específicos

a) Implementar hibridização heurística para o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea;
b) Caracterizar as soluções heurísticas referente a completude do Problema;
c) Avaliar o impacto da hibridização, visto o esforço computacional, frente aos resultados relevantes presentes na literatura;

1.2 Estrutura do trabalho

O presente trabalho está organizado da seguinte maneira: no capítulo 2, são explicados os conceitos necessários para o entendimento da abordagem utilizada. No capítulo 3, é caracterizado o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea.

No capítulo 4, apresenta-se uma breve revisão literária sobre o Problema de Roteamento de Veículos, algumas de suas variações, e trabalhos relacionados, incluindo o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea. No capítulo 5 é apresentada a metodologia utilizada para resolver o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea. Enquanto o capítulo 6 apresenta os resultados obtidos pelo algoritmo. Finalmente, o capítulo 7 apresenta as principais conclusões, bem como os possíveis aprimoramentos do algoritmo desenvolvido.

2 FUNDAMENTAÇÃO TEÓRICA

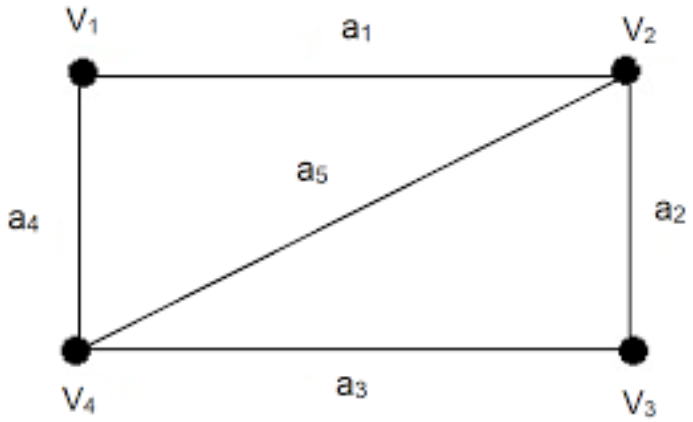
Neste capítulo, apresenta-se uma breve fundamentação teórica de conceitos e termos de Teoria dos Grafos, Otimização Combinatória, Heurísticas e Meta-heurísticas, base importante para compreensão das estratégias utilizadas nesta pesquisa.

2.1 Teoria dos grafos

Teoria dos Grafos é uma área da Matemática que estuda a relação entre os objetos de um conjunto específico sendo aplicada em vários problemas reais. Para tal finalidade, são empregadas estruturas chamadas de grafos.

Segundo Bollobás (1979), um grafo G é um par ordenado $(V(G), E(G))$, consistindo de um conjunto não-vazio $V(G)$ de vértices e um conjunto $E(G)$ de arestas. Cada par não-ordenado de vértices representa uma aresta. Se o par (x, y) está em $E(G)$, escreve-se $xy \in E(G)$ e se afirma que a aresta xy incide sobre os vértices x e y. Por exemplo, na Figura 1 tem-se que $V(G) = \{V_1, V_2, V_3, V_4\}$ e $E(G) = \{a_1, a_2, a_3, a_4, a_5\}$.

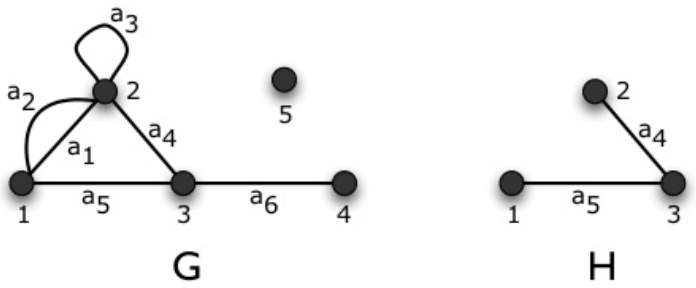
Figura 1 – Exemplo de Grafo.



Fonte: (FAVARO, 2017)

De acordo com West *et al.* (2001), o grafo H é um subgrafo de G se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$. Por $H \subseteq G$, denota-se que H é subgrafo de G, conforme ilustrado na Figura 2.

Figura 2 – Exemplo de subgrafo.



Fonte: (ALCANTARA, 2014)

Chama-se de passeio de tamanho k em um grafo G, uma sequência não-nula $W = v_0e_1v_1e_2...e_kv_k$, cujos termos são vértices e arestas alternados, tal que, para $1 \leq i \leq k$, a aresta e_i incide sobre os vértices v_{i-1} e v_i(BONDY *et al.*, 1976). Um caminho é um passeio W em que os vértices são distintos entre si. Já a trilha é um passeio com as arestas distintas entre si. Na Figura 1, as sequências $V_1a_1V_2a_5V_4a_3V_3a_2V_2$, $V_1a_1V_2a_5V_4$ e $V_4a_5V_2a_2V_3a_3V_4$ são exemplos, respectivamente, de passeio, caminho e trilha.

Em um grafo, circuito é um passeio em que o vértice inicial e o vértice final são iguais e sem repetição de aresta. Já o ciclo é um circuito em que todos os vértices são distintos(com exceção do primeiro e último)(RUOHONEN, 2013).

Um vértice v é adjacente a um outro vértice u, se a aresta uv incidir sobre os vértices v e u. A vizinhança de um vértice v no grafo G é o conjunto de todos os vértices adjacentes a v, denota-se por $N(v)$(BOLLOBÁS, 1979). Utilizamos o grafo da Figura 1 para ilustrar essas definições, o vértice V_1 é adjacente aos vértices V_2,V_4 e não adjacentes aos vértices restantes do grafo. Com isso, $N(V_1) = \{V_2,V_4\}$.

Modelos baseados em grafos são amplamente utilizados em problemas de Otimização Combinatória, desta forma na seção a seguir serão abordados alguns conceitos fundamentais.

2.2 Otimização Combinatória

Na Matemática Aplicada e na Ciência da Computação teórica, a Otimização Combinatória é uma área que consiste em encontrar um objeto ideal a partir de um conjunto finito de objetos (KORTE *et al.*, 2012). Em problemas NP-difíceis, a busca exaustiva não é viável, pois não pode ser processada em tempo polinomial em uma máquina de Turing determinística. A busca opera no domínio desses problemas de otimização, nos quais o conjunto de soluções viáveis é discreto ou pode ser discretizado, e no qual o objetivo é encontrar a melhor solução, de

acordo com uma métrica definida para o problema. Alguns problemas comuns que envolvem Otimização Combinatória são o Problema do Caixeiro Viajante(PCV) (HOFFMAN *et al.*, 2013) e o da Árvore Geradora Mínima(AGM) (NEŠETRIL; NEŠETRILOVÁ, 2012).

A Otimização Combinatória é uma sub-área da Otimização Matemática relacionada a Pesquisa Operacional(PO), a Teoria dos Algoritmos e a Teoria da Complexidade Computacional. Ela tem aplicações importantes em vários campos, incluindo Inteligência Artificial(CZYZŻAK; JASZKIEWICZ, 1998), Aprendizado de Máquina(CASTRO; ZUBEN, 2002) e Engenharia de Software(NIE; LEUNG, 2011). Na Otimização Combinatória, tem-se como principais subáreas: a Programação Linear(PL) e Programação Inteira(PI).

2.2.1 Programação Linear(PL)

PL é uma sub-área de Otimização Combinatória usada em modelagem computacional para o encontro da melhor solução possível na alocação de recursos limitados (energia, máquinas, materiais, dinheiro, pessoal, espaço, tempo, por exemplo.) para otimizar um determinado objetivo (exemplos: maximizar lucro ou minimizar custo). No entanto, é aplicável somente quando todos os relacionamentos são lineares(ou linearizados) e pode acomodar apenas uma classe limitada de funções de custo.

Um problema de programação linear é formulado de forma a encontrar um conjunto de variáveis positivas x_j que minimizem uma função objetivo linear 2.1 sujeita a um conjunto de restrições lineares 2.2. Em Cabral (1996) definiu-se a forma-padrão de PL para um problema de minimização da seguinte maneira:

$$minimizar \;\; Z = \sum_{j=1}^{n} c_j x_j \tag{2.1}$$

Sujeito a:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad para\ i = 1, ..., m. \tag{2.2}$$

$$x_j \geq 0, \quad para\ j = 1, ..., n. \tag{2.3}$$

Sendo que c_j, a_{ij}, b_i e x_j pertencem aos números reais, tais que representam, respectivamente, o fator de custo, a matriz de valores dos fatores das m restrições, os termos independentes de cada restrição do modelo e a solução do problema.

2.2.2 Programação Inteira(PI)

PI diferencia-se de PL, pelo fato de que algumas ou todas as variáveis possuem valores discretos, ou seja, valores inteiros(CABRAL, 1996). Em Korte *et al.* (2012), definiu-se formalmente PI com a seguinte forma-padrão:

$$minimizar \;\; Z = \sum_{j=1}^{n} c_j x_j \tag{2.4}$$

Sujeito a:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad para\ i = 1, ..., m. \tag{2.5}$$

$$x_j \in \mathbb{Z} \quad para\ j = 1, ..., p(p \leq n). \tag{2.6}$$

$$x_j \geq 0, \quad para\ j = p+1, ..., n. \tag{2.7}$$

Nesse modelo descrito acima, no caso de apenas parte das variáveis terem valores inteiros, ou seja, para $p < n$ tem-se um modelo de Programação Inteira Mista. Agora se todas as variáveis forem inteiras, isto é, se $p = n$ tem-se um modelo de Programação Inteira Pura.

Estes modelos de programação podem ser abordados através de métodos exatos ou aproximativos. Os métodos exatos são procedimentos que garantem encontrar a solução ótima global para o modelo, e os aproximativos entretanto não garantem que a solução obtida seja a melhor possível. Este trabalho aborda os métodos aproximativos, estes são decompostos em duas classes: as heurísticas e as meta-heurísticas.

2.3 Heurísticas

Heurística é uma técnica que surgiu como alternativa aos métodos exatos para resolução de problemas de otimização que não podem ser solucionados em tempo computacional aceitável(FÁVERO; BELFIORE, 2012). O termo heurística vem do grego e significa descobrir. Uma heurística pode ser definida como um procedimento de busca guiada, por regras e ideias, objetivando construir uma boa solução, não necessariamente a ótima global. Na literatura, existem dois tipos principais de heurísticas: as Heurísticas Construtivas e as Heurísticas de Refinamento.

2.3.1 Heurísticas Construtivas

As Heurísticas Construtivas têm o início de seus procedimentos por meio de uma solução vazia e estende iterativamente a solução atual, com o objetivo de obter soluções melhores (GLOVER; KOCHENBERGER, 2006). Com o objetivo de construir uma solução elemento a elemento, a forma de escolha de cada elemento a ser inserido a cada passo varia de acordo com a função de avaliação adotada, que por sua vez depende do problema abordado. Nas heurísticas clássicas, a cada iteração os elementos candidatos são ordenados segundo uma função gulosa que estima o benefício na inserção de cada elemento e somente o que dá resultados satisfatórios à solução é adicionado (SOUZA, 2008). Essas heurísticas são geralmente utilizadas para gerar a solução inicial do problema.

A Figura 3 mostra o pseudocódigo para a construção da solução inicial para um problema de otimização que usa a função gulosa $G(.)$. Nesta figura, t_{melhor} indica o elemento com o valor mais favorável para a função de avaliação G, ou seja, aquele que tenha menor valor de G, no caso de um problema de minimização, ou que tenha o maior valor de G, no caso de um problema de maximização.

Figura 3 – Heurística de construção gulosa de uma solução inicial.

procedimento ConstrucaoGulosa ($G(.)$, s)
1. $s \leftarrow \emptyset$;
2. Inicialize o conjunto C de elementos candidatos;
3. **enquanto** $C \neq \emptyset$ **faça**
4. $\quad G(t_{melhor}) = melhor\{G(t) | t \in C\}$;
5. $\quad s \leftarrow s \cup \{t_{melhor}\}$;
6. $\quad$ Atualize o conjunto C de elemento candidatos;
7. **fim-enquanto**
8. retorna s;

fim ConstrucaoGulosa.

Fonte: (SOUZA, 2008)

Uma outra forma comumente usada para gerar uma solução inicial é escolher os elementos candidatos aleatoriamente. Ou seja, a cada iteração, o elemento a ser inserido na solução é aleatoriamente selecionado. A Figura 4 ilustra o pseudo-código deste procedimento.

Figura 4 – Heurística de construção aleatória de uma solução inicial

procedimento ConstrucaoAleatoria ($G(.)$, s)
1. $s \leftarrow \emptyset$;
2. Inicialize o conjunto C de elementos candidatos;
3. **enquanto** $c \neq \emptyset$ **faça**
4. Escolha aleatoriamente $t_{escolhido} \in C$;
5. $s \leftarrow s \cup \{t_{escolhido}\}$;
6. Atualize o conjunto C de elemento candidatos;
7. **fim-enquanto**
8. retorna s;

fim ConstrucaoAleatoria.

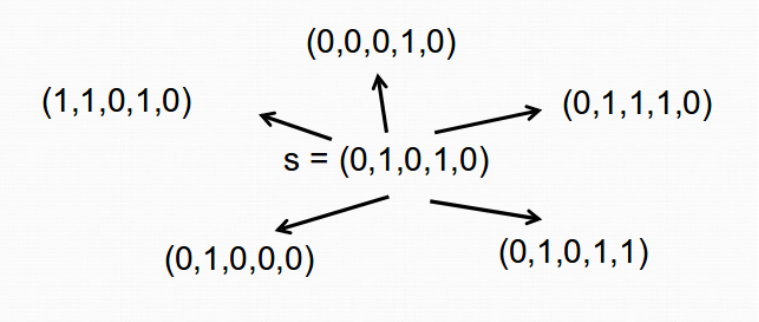
Fonte: (SOUZA, 2008)

Enquanto as Heurísticas Construtivas baseiam-se na ideia da construção de uma solução a cada iteração, as Heurísticas de Refinamento baseiam-se na noção de vizinhança. A seguir define-se com mais detalhes os conceitos de Heurísticas de Refinamento.

2.3.2 Heurísticas de Refinamento

As heurísticas de refinamento em problemas de otimização, também chamadas de técnicas de busca local, constituem uma família de técnicas baseadas na noção de vizinhança. Mais especificamente, seja S o conjunto de todas as soluções possíveis para um dado problema de otimização e f a função objetivo a minimizar. A função N, a qual depende da estrutura do problema tratado, associa a cada solução $s \in S$, sua vizinhança $N(s) \subseteq S$. Cada solução $n \in N(s)$ é chamada de vizinho de s. Denomina-se movimento a modificação m que transforma uma solução s em outra n que esteja em sua vizinhança. Representa-se esta operação por $n \leftarrow s \oplus m$(SOUZA, 2008). Consideremos o Problema da Mochila, que dado o conjunto de n objetos e uma mochila com o benefício de objeto j representado por c_j, peso do objeto j representado por w_j e capacidade W da mochila. O objetivo do problema é determinar quais objetos devem ser colocados na mochila para maximizar o benefício total, de tal forma que o peso da mochila não ultrapasse a sua capacidade. Uma solução s para este problema é um vetor de uns e zeros, tal que se o objeto j estiver na mochila, então $s_j = 1$, caso contrário $s_j = 0$. A Figura 5 ilustra uma exemplificação de vizinhança para este problema.

Figura 5 – Exemplo de vizinhança no Problema da Mochila.



Fonte: Próprio autor.

O conceito de vizinhança permite definir o conceito de solução ótima local. Uma solução máxima local a uma estrutura de vizinhança N é uma solução s', tal que para todo $n \in N(s') : f(s') \geq f(n)$ (BLUM; ROLI, 2003).

Nas seções 2.3.2.1 a 2.3.2.3, apresentam-se as principais heurísticas clássicas de refinamento. Enquanto que na seção 2.3.2.4 apresenta-se uma heurística mais robusta, que explora o espaço de soluções do problema através de trocas sistemáticas.

2.3.2.1 Método da Descida/Subida

Podemos descrever essa técnica partindo de alguma solução inicial, cada passo analisa todos os seus possíveis vizinhos, realizando a movimentação somente para o que represente uma melhora no valor corrente da função de avaliação. Na Figura 6, mostra o pseudocódigo do Método de Descida aplicado à minimização da função f a partir de uma solução inicial s e considerando a busca em uma determinada vizinhança $N(.)$.

Figura 6 – Método da descida

procedimento Descida ($f(.), N(.), s$)
1. $V = \{n \in N(s) | f(n) < f(s)\}$;
2. **enquanto** $|V| > 0$ **faça**
3. Selecione $n \in N(s)$, onde $n = argmin\{f(n) | n \in V\}$;
4. $s \leftarrow n$;
5. $V = \{n \in N(s) | f(n) < f(s)\}$
6. **fim-enquanto**
7. retorna s;

fim Descida.

Fonte: (SOUZA, 2008)

2.3.2.2 Método de Primeira Melhora

Esse método se assemelha com o apresentado acima, com a diferença de que não é necessário explorar toda a vizinhança. A exploração de vizinhança é interrompida quando um vizinho melhor é encontrado. A Figura 7 mostra o pseudocódigo de tal método.

Figura 7 – Método de Primeira Melhora.

procedimento PrimeiraMelhora ($f(.)$, $N(.)$, s)
1. $V = \{n \in N(s) | f(n) < f(s)\}$;
2. **enquanto** $|V| > 0$ **faça**
3. Selecione $n \in N(s)$ o primeiro n tal que $f(n) < f(s)$
4. $s \leftarrow n$;
5. $V = \{n \in N(s) | f(n) < f(s)\}$
6. **fim-enquanto**
7. retorna s;

fim PrimeiraMelhora.

Fonte: (SOUZA, 2008)

2.3.2.3 Método de Descida/Subida Randômica

Esse método consiste em verificar um vizinho qualquer e o aceitar se ele for estritamente melhor que a solução corrente. Não o sendo, a solução atual não é alterada e outro vizinho é gerado. O procedimento é interrompido quando um número fixo de iterações forem executadas sem melhora na solução obtida(SOUZA, 2008).

Figura 8 – Método da Descida Randômica.

procedimento DescidaRandomica ($f(.)$, $N(.)$, s,$IterMax$)
1. $Iter \leftarrow 0$;
2. **enquanto** $Iter < IterMax$ **faça**
3. $Iter = Iter + 1$;
4. Selecione aleatoriamente $n \in N(s)$;
5. **Se** $f(n) < f(s)$ **então**
6. $Iter \leftarrow 0$;
7. $s \leftarrow n$
8. **fim-se**
9. **fim-enquanto**
10. retorna s;

fim DescidaRandomica.

Fonte: (SOUZA, 2008)

2.3.2.4 Variable Neighborhood Descent(VND)

A heurística VND foi proposta por Hansen e Mladenović (2001) a qual representa um método de busca local que explora o espaço de soluções através de trocas sistemáticas de estrutura de vizinhança, e retorna a estrutura de vizinhança que melhora a solução corrente.

No pseudocódigo da Figura 9, apresenta-se o refinamento de uma solução s utilizando uma função de avaliação f a ser minimizada e um conjunto de r diferentes vizinhanças $N = N^{(1)}, N^{(2)}, ..., N^{(r)}$. O VND inicia processando a primeira estrutura de vizinhança $N^{(1)}$. A cada iteração, o procedimento gera o melhor vizinho $s^{'}$ da solução corrente s na vizinhança $N^{(k)}$. Se a solução s' for melhor que s, então s' passa a ser nova solução corrente e volta para a primeira estrutura de vizinhança $N^{(1)}$. Caso contrário, passa-se para a próxima estrutura de vizinhança $N^{(k+1)}$. O procedimento encerra quando não é possível achar uma solução $s^{'} \in N^{(r)}$ melhor que a solução corrente s.

Figura 9 – Algoritmo VND.

procedimento VND $(f(.), N(.), s)$
1. Seja r o número de estruturas de vizinhanças
2. $k \leftarrow 1$;
3. **enquanto** $(k \leq r)$ **faça**
4. Encontre o melhor vizinho $s' \in N^{(k)}(s)$;
5. **se** $f(s') < f(s)$ **então**
6. $s \leftarrow s'$;
7. $k \leftarrow 1$;
8. **senao**
9. $k \leftarrow k+1$;
10. **fim-se**
11. **fim-enquanto**
12. retorna s;

fim VND.

Fonte: (SOUZA, 2008)

Enquanto as heurísticas realizam uma busca limitada no espaço de soluções, as Meta-heurísticas seguem os princípios de intensificação e diversificação.A intensificação visa uma exploração mais detalhada das áreas mais promissoras do espaço de busca, enquanto a diversificação evita que uma solução fique presa em um ótimo local durante o processo de busca(GLOVER; KOCHENBERGER, 2006).

2.4 Meta-heurísticas

Conforme Ribeiro (1996), as meta-heurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico. As heurísticas subordinadas podem ser procedimentos de alto (ou baixo) nível, ou uma simples pesquisa local, ou apenas um método de construção(VOSS *et al.*, 2012).

As meta-heurísticas diferenciam-se entre si pelo mecanismo usado para escapar dos ótimos locais. Elas se dividem em duas categorias de acordo com o princípio usado para explorar a solução: busca local(VANSTEENWEGEN *et al.*, 2009) e busca populacional(BIRBIL *et al.*, 2004).

Nas meta-heurísticas baseadas em busca local, a exploração do espaço de soluções é feita por meio de movimentos, os quais são aplicados a cada passo sobre a solução atual. Os métodos *Iterated Local Search*(ILS), *Variable Neighborhood Search*(VNS) e *Greedy Randomized Adaptative Search Procedure*(GRASP) são exemplares desta categoria. Este trabalho baseia-se no uso das meta-heurísticas GRASP e VNS.

Já os métodos baseados em busca populacional, mantêm as melhores soluções e tenta combiná-las para obter soluções ainda melhores. Exemplos de procedimentos desta categoria são Algoritmos Genéticos, *Ant Colony Optimization(ACO)* e Redes Neurais.

2.4.1 Greedy Randomized Adaptative Search Procedure(GRASP)

Método iterativo proposto em Feo e Resende (1995) que possui duas etapas: uma etapa de construção, na qual uma solução é gerada, elemento a elemento, e de uma fase de busca local, na qual um ótimo local na vizinhança da solução construída é pesquisado. A melhor solução encontrada ao longo de todas as iterações realizadas é retornada como resultado(SOUZA, 2008). O pseudocódigo da Figura 10 ilustra o procedimento GRASP para um problema de minimização.

Figura 10 – Meta-heurística GRASP.

procedimento GRASP $(f(.), g(.), N(.), GRASPmax, s)$
1. $f* \leftarrow \infty$;
2. **para** $Iter = 1, 2, ..., GRASPmax$ **faça**
3. Construcao$(g(.), \alpha, s)$;
4. BuscaLocal$(f(.), N(.), s)$;
5. **se** $f(s) < f^*$ **então**
6. $s^* \leftarrow s$;
7. $f^* \leftarrow f(s)$;
8. **fim-se**
9. **fim-enquanto**
10. $s \leftarrow s^*$
11. retorna s;

fim GRASP.

Fonte: (SOUZA, 2008)

Na fase de construção do GRASP há um subconjunto restrito formado pelo melhores elementos do procedimento. Este conjunto é denominado de Lista de Candidatos Restrita(LCR), conforme a Figura 11 e $\alpha \in [0,1]$ é um parâmetro que controla o nível de gulosidade e aleatoriedade. Sendo que $\alpha = 0$ faz gerar soluções puramente gulosas e $\alpha = 1$ produz soluções totalmente aleatórias. As soluções geradas pela fase de construção certamente não são localmente ótimas quanto à definição de vizinhança aplicada. Com isso, percebe-se a importância da fase de busca local, a qual objetiva melhorar a solução construída. A Figura 12 ilustra o procedimento de Busca Local.

Figura 11 – Fase de construção do algoritmo GRASP.

procedimento Construcao $(g(.), \alpha, s)$
1. $s \leftarrow \emptyset$;
2. Inicialize o conjunto C de candidatos;
3. **enquanto** $C \neq \emptyset$ **faça**
4. $g(t_{min}) = min\{g(t) | t \in C\}$;
5. $g(t_{max}) = max\{g(t) | t \in C\}$;
6. $LCR = \{t \in C | g(t) \leq g(t_{min}) + \alpha(g(t_{max}) - g(t_{min}))\}$;
7. Selecione aleatoriamente, um elemento $t \in LCR$;
8. $s \leftarrow s \cup t$;
9. Atualize o conjunto C de candidatos;
10. **fim-enquanto**
11. retorna s;

fim Construcao.

Fonte: (SOUZA, 2008)

Figura 12 – Fase de Busca Local do algoritmo GRASP.

procedimento BuscaLocal $(f(.), N(.), s)$
1. $V = \{n \in N(s) | f(n) < f(s)\}$;
2. **enquanto** $|V| > 0$ **faça**
4. Selecione $n \in N(s)$;
5. $s \leftarrow n$;
6. $V = \{n \in N(s) | f(n) < f(s)\}$
7. **fim-enquanto**
8. retorna s;

fim BuscaLocal.

Fonte: (SOUZA, 2008)

Em Feo e Resende (1995), discutiu-se o valor de α na qualidade ou diversidade das soluções geradas. O valor de α próximo da escolha gulosa implica em soluções obtidas com baixo esforço computacional. Já um α perto da aleatoriedade, leva a uma grande quantidade de variações de soluções produzidas, entretanto muitas das soluções são de baixa precisão, tornando mais demorado o processo de busca local.

2.4.2 Variable Neighbourhood Search(VNS)

A Meta-heurística VNS é um procedimento de busca baseado em vizinhanças variáveis que explora sistematicamente a troca de estruturas de vizinhança, com o objetivo de escapar de soluções locais e encontrar soluções globais(SOUZA, 2008). Diferente de outras meta-heurísticas baseadas em busca local não segue uma trajetória, entretanto explora vizinhanças mais afastadas da solução atual e objetiva a busca em torno de uma nova solução, se e somente se, um movimento de melhora na solução é feito.

A principal estratégia dos algoritmos de busca em vizinhança variável baseia-se em três hipóteses(HANSEN; MLADENOVIĆ, 2001), que são descritas a seguir:

- Um ótimo local de um determinada estrutura de vizinhança pode não ser um ótimo local em relação às outras estruturas de vizinhança.
- Um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança.
- Para muitos problemas, os ótimos locais em relação a uma ou várias estruturas, são bem próximos uns dos outros.

O algoritmo inicia com uma solução inicial s_0, a cada iteração é selecionado um vizinho $s^{'}$ dentro da vizinhança $N^{(k)}(s)$ da solução atual s. Esta solução é subordinada a um método de busca local. Se a solução local $s^{''}$ for melhor do que a solução corrente s, a busca

continua de s'' recomeçando na primeira estrutura de vizinhança $N^{(1)}(s)$. Senão, prossegue-se a busca a partir da próxima estrutura de vizinhança $N^{(k+1)}(s)$. O procedimento é terminado quando o numero de máximo de iterações é atingido. A Figura 13 ilustra o pseudocódigo deste procedimento.

Figura 13 – Meta-heurística VNS.

procedimento VNS $(f(.), N(.), s_0, MaxIterVNS)$
1. Seja r o número de estruturas de vizinhanças.
2. Selecione um conjunto de vizinhos $N^{(k)}$, para cada $k = 1, ..., r$;
3. $s \leftarrow s_0$;
4. $Iter \leftarrow 1$;
5. **enquanto** $Iter \leq MaxIterVNS$ **faça**
6. $k \leftarrow 1$;
7. **enquanto** $k \leq r$ **faça**
8. Gere um vizinho qualquer $s' \in N^{(k)}(s)$;
9. $s'' \leftarrow BuscaLocal(s')$;
10. **se** $f(s^{(''')} < f(s))$ **então**
11. $s \leftarrow s''$;
12. $k \leftarrow 1$;
13. **senão**
14. $k \leftarrow k+1$;
15. **fim-se**
16. **fim-enquanto**
17. $Iter \leftarrow Iter + 1$;
18. **fim-enquanto**
19. retorna s;

fim VNS.

Fonte: (SOUZA, 2008)

Conhecido os conceitos necessários para a compreensão das estratégias utilizadas nesta pesquisa, no próximo capítulo é apresentada a descrição do problema estudado.

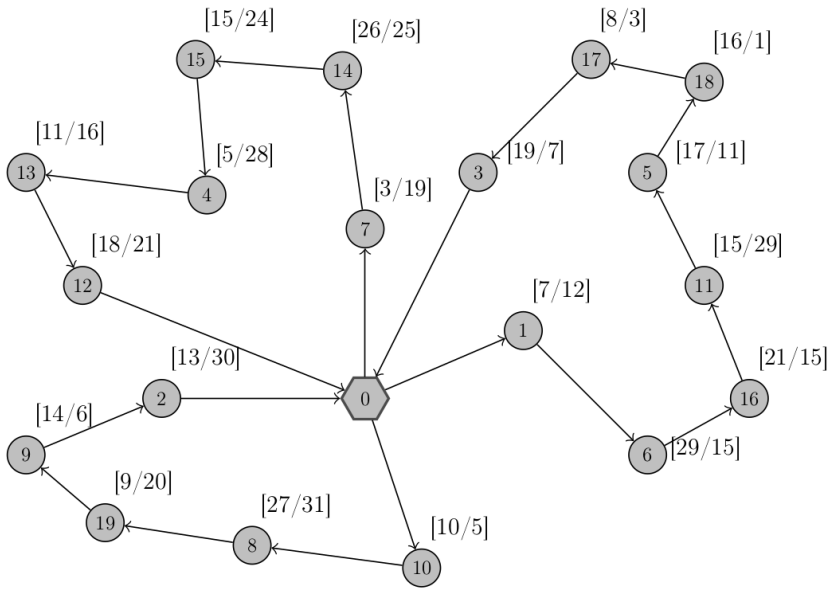
3 PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEA(PRVCES)

Conforme Mine *et al.* (2010), o PRVCES é um problema NP-Difícil que objetiva construir rotas para atender a entrega de produtos e o recolhimento de resíduos dos clientes simultaneamente com o custo mínimo possível. O problema é composto por um depósito que conta com um número de veículos com capacidade definida para atender as demandas de coleta e entrega de clientes. Em CARABETTI *et al.* (2010), constatou-se que um Problema de Roteamento de Veículos com Coleta e Entrega Simultânea deve atender as seguintes condições:

- o início e fim da rota devem se dar no depósito.
- o cliente só deverá ser visitado uma única vez.
- somente um veículo deve ser utilizado para atender a um cliente.
- a resultante das demandas de coleta e entrega da rota não pode ser superior a capacidade do veículo.

Na Figura 14, ilustra-se o PRVCES, sendo que os nós de 1 a 19 representam os clientes e o nó 0 representa o depósito ou centro de distribuição. Cada par $[d_i/p_i]$ indica a demanda de entrega e coleta de um cliente *i*, respectivamente.

Figura 14 – Ilustração do Problema de Roteamento de Veículo com Coleta e Entrega Simultânea.



Fonte: (MINE *et al.*, 2010)

No trabalho de Subramanian *et al.* (2011) foi formulado um modelo de programação linear inteira mista para este problema. Na formulação deste modelo matemático, tem-se os

seguintes parâmetros:

- N é o conjunto dos clientes.
- N^+ é o conjunto dos clientes, incluindo o depósito.
- N^* é o conjunto incluindo o depósito e uma cópia do depósito.
- A é o conjunto de arcos (i, j) com $i, j \in N^*$.
- c_{ij} é a distância entre os clientes i e j.
- D_i é a quantidade de produtos a ser entregue ao cliente i.
- P_i é a quantidade de produtos a ser coletado ao cliente i.
- K é a quantidade de rotas ou veículos disponíveis.
- Q é a capacidade do veículo.
- d_{ij} é a quantidade de produtos a serem entregues a clientes e escoados no arco (i, j).
- p_{ij} é a quantidade de produtos coletados a clientes e escoados no arco (i, j).
- x_{ij} indica se o arco (i, j) está presente na solução $(x_{ij} = 1)$ ou não $(x_{ij} = 0)$
- w_{ij} é a carga do veículo no arco $(i, j) \in A$, referente à entrega e coleta.

O modelo é descrito da seguinte maneira:

$$minimizar \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{3.1}$$

Sujeito a:

$$\sum_{j\in N^*} (d_{ji} - d_{ij}) = 2D_i \ \ (i \in N). \tag{3.2}$$

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

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

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

$$\sum_{j\in N} p_{j0} = \sum_{i\in N} P_i \tag{3.6}$$

$$\sum_{j\in N} p_{0j} = KQ - \sum_{i\in N} P_i \tag{3.7}$$

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

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

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

$$w_{j0} = p_{j0} \ \ (j \in N). \tag{3.11}$$

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

$$d_{ij} + d_{ji} = Qx_{ij} \ \ ((i,j) \in A). \tag{3.13}$$

$$p_{ij} + p_{ji} = Qx_{ij} \ \ ((i,j) \in A). \tag{3.14}$$

$$w_{ij} + w_{ji} = Qx_{ij} \ \ (i \in N). \tag{3.15}$$

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

$$\sum_{j\in N} x_{0j} \leq K \ \ (k \in N). \tag{3.17}$$

$$\sum_{j \in N} x_{j0} \leq K \tag{3.18}$$

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

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

A Equação (3.1) representa a função objetivo do problema que busca minimizar a distância total percorrida pelos veículos. A restrição (3.2) garante 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 de entrega dos clientes. A restrição (3.4) define que a carga dos veículos que chega no depósito seja igual à carga residual dos veículos quando saem do depósito. As restrições (3.5) a (3.7) são similares às restrições (3.2) a (3.4), porém relacionam-se com a demanda de coleta. A restrição (3.8) garante que as entregas e coletas são realizadas simultaneamente. As restrições (3.9) a (3.15) determinam que a capacidade do veículo não seja ultrapassada. A restrição (3.16) define que o número de arcos incidentes em cada cliente seja igual a dois. As restrições (3.17) e (3.18) estabelecem que a capacidade do veículo não seja excedida. As restrições (3.17) e (3.18) estabelecem a quantidade máxima de veículos utilizados. A restrição (3.19) define as variáveis x_{ij} como binárias. Finalmente, a restrição (3.20) define que as variáveis p_{ij}, d_{ij} e w_{ij} são contínuas e positivas.

Compreendido a natureza e característica do PRVCES, é importante e necessário o conhecimento de alguns trabalhos importantes sobre o problema e seus problemas relacionados, à qual são apresentados a seguir.

4 REVISÃO LITERÁRIA

Neste capítulo, apresenta-se a revisão bibliográfica sobre o Problema de Roteamento de Veículos com Coleta e Entrega e as principais técnicas utilizadas para a sua resolução. Na seção 4.1, é feita uma breve descrição do Problema de Roteamento de Veículos e algumas de suas variações para contextualização do problema a ser abordado neste trabalho. Na seção 4.2, são descritas metodologias empregadas em diversos problemas que trabalham com serviços de coleta e entrega. Na seção 4.3, é apresentada uma breve revisão bibliográfica dos problemas relacionados ao problema Problema de Roteamento de Veículos com Coleta e Entrega Simultânea(PRVCES). Por último, a seção 4.4 apresenta as principais técnicas utilizadas na literatura para resolver o PRVCES, que é o foco principal deste trabalho.

4.1 Problema de Roteamento de Veículos

O Problema de Roteamento de Veículos(PRV), apresentado por Dantzig e Ramser (1959), tem como propósito definir rotas entre um depósito e um conjunto de pontos de entrega que minimize o número de veículos, a distância percorrida ou tempo gasto. As restrições básicas do problema são descritas a seguir:

- Cada cidade é visitada uma única vez por um único veículo.
- Cada rota é iniciada em um depósito e finalizada no mesmo depósito.
- Todas as demandas de todos os consumidores devem ser satisfeitas.

Na literatura, pode-se encontrar vários trabalhos que apresentam diferentes abordagens para os problemas de roteamento de veículos, dentre elas cita-se: a existência de um único depósito (BEASLEY, 1983) (ROUSSEAU *et al.*, 2002), existência de vários depósitos no cenário do problema (SALHI; NAGY, 1999), veículos homogêneos, ou seja, todos os veículos do problema possuem a mesma capacidade (CAMPOS; MOTA, 2000), veículos heterogêneos (BELFIORE; YOSHIZAKI, 2006), problemas de coleta e/ou entrega (MIN, 1989).

4.2 Problemas de Coleta e Entrega

Muitos problemas de roteamento de veículos com coleta e entrega têm sido investigados para tratar várias situações reais. Ambos baseiam-se em um único objetivo: uso eficiente de uma frota de veículos que deve satisfazer demandas de entrega e/ou coleta de um conjunto de consumidores ou clientes. As demandas de cada consumidor devem ser satisfeitas por um único

veículo (XU *et al.*, 2003).

Sabendo-se dessa similaridade, Savelsbergh e Sol (1995) mostra o Problema Geral de Coleta e Entrega (GPDP, do inglês *General Pickup and Delivery Problem*) que combina várias características achadas entre esses problemas práticos de coleta e entrega. O GPDP consiste em especificar um conjunto de rotas com o intuito de satisfazer um conjunto de demandas de transporte.

Todo consumidor possui demandas de coleta e entrega. As demandas de coleta possuem dados sobre a carga total a ser coletada pelo veículo e em quais consumidores essa carga deve ser entregue posteriormente. As demandas de entrega possuem informações sobre a carga total a ser entregue e em quais consumidores ela deve ter sido coletada anteriormente. Nesse caso, existe uma regra de prioridade entre consumidores, pois para satisfazer uma demanda de entrega o veículo deverá satisfazer anteriormente a respectiva demanda de coleta. A posição inicial e a posição final dos veículos também é informada. Com isso, o objetivo é definir rotas otimizadas, que tenham como ponto de partida e ponto de chegada aqueles definidos para os veículos e que satisfaçam todas as demandas dos consumidores sem ultrapassar a capacidade dos veículos.

Em Parragh *et al.* (2008), dividiu-se o Problema Geral de Coleta e Entrega em dois grupos distintos: o de transporte de cargas do depósito para os consumidores e dos consumidores para o depósito. Nesse grupo, todos os veículos saem do depósito contendo a carga necessária para satisfazer as demandas de entrega dos consumidores as cargas recolhidas pelo veículo nos consumidores devem ser entregues no depósito. O segundo grupo é o de transporte de carga entre consumidores. Nesse grupo, os veículos partem de um ponto inicial sem nenhuma carga e percorrem vários pontos de coleta e/ou entrega de itens e então retornam a um ponto final. Nesse caso, demandas de entrega são sempre satisfeitas por objetos coletados anteriormente pelo veículo.

4.3 Problemas Relacionados

4.3.1 Problema do Caixeiro Viajante com Coleta e Entrega(PCVCE)

Este problema se assemelha com o PCV, sendo que cada cliente possui as demanda de coleta e entrega. O PCVCE foi proposto em Mosheiov (1994). Neste trabalho, o autor mostra uma abordagem heurística para o problema e propõe algumas aplicações. Balas *et al.* (1995)

utilizaram restrições de precedência para o PCV e PCVCE que são úteis para outros trabalhos com algoritmos exatos.

Dell'Amico *et al.* (2006) mostraram duas heurísticas para o PCVCE, sendo que a primeira baseia-se numa solução exata e a outra foi a Meta-heurística Busca Tabu. Em Renaud *et al.* (2002), comparou e descreveu 7 heurísticas de pertubação para o PCVCE. Uma pesquisa feita por Beraldi *et al.* (2005) sobre PCVCE mostra procedimentos com complexidade $O(n^3)$, utilizando busca em vizinhança e outras aproximações com complexidade $O(n^5)$.

Hernández-Pérez e Salazar-González (2004) propuseram um modelo de Programação Linear Inteira para a generalização do Problema do Caixeiro Viajante. Neste trabalho utilizaram o algoritmo *branch-and-cut* para extrair as soluções ótimas. Sendo que o modelo proposto pode ser facilmente adaptado para resolver o PCVCE, foram obtidos bons resultados para 75 clientes.

4.3.2 Dial-a-Ride Problem(DARP)

DARP representa um Problema de Coleta e Entrega tal que a carga são pessoas, chamadas de clientes e o tamanho da carga são todas iguais a 1(COSTA *et al.*, 2013). Os clientes são recolhidos em locais preestabelecidos e transportados para pontos definidos. O problema objetiva minimizar o número de veículos, caso no problema possa haver vários veículos, ou a distância trafegada, caso exista um único veículo. Em todo os casos, o problema leva em consideração o tempo consumido no embarque e desembarque dos passageiros.

DARP foi proposto por Psaraftis (1980), na qual o autor criou um algoritmo de programação dinâmica para casos com um único veículo. Na literatura, pode-se encontrar trabalhos que resolveram o problema utilizando heurísticas. Por exemplo, em Ho e Haugland (2004) desenvolveram uma meta-heurística Busca-Tabu e uma hibridização utilizando GRASP e Busca Tabu. No trabalho, foi confirmado que os resultados obtidos pelo primeiro algoritmo foram mais satisfatórios. Em Costa *et al.* (2013), foi apresentado um algoritmo que utiliza a meta-heurística *Iterated Local Search* juntamente com a *Variable Neighborhood Search* para resolver o DARP, em comparação com outros trabalhos relacionados, neste trabalho, foram obtidos melhores resultados com relação à distância percorrida e ao tempo médio de viagem dos clientes.

4.4 Problema de Roteamento de Veículos com Coleta e Entrega Simultânea(PRVCES)

PRVCES foi proposto pela primeira vez em Min (1989). O autor apresentou uma heurística para resolver o problema relacionado à distribuição e coleta de livros de uma biblioteca pública. Esta heurística baseia-se em 3 fases construtivas.

Existem abordagens exatas para a sua resolução na literatura, dentre as existentes, tem-se o algoritmo *Branch-and-price* desenvolvido por Dell'Amico *et al.* (2006) em que duas estratégias distintas foram utilizadas: (i) programação dinâmica exata; e (ii) relaxação do espaço de estados. Os autores conseguiram encontrar soluções ótimas para 40 clientes. Além disso, Angelelli e Mansini (2002) desenvolveram uma abordagem *Branch-and-price* baseada na formulação de cobertura de conjunto, mas para o PRVCES com janelas de restrição de tempo. O sub-problema foi formulado como um caminho mais curto com restrições de recursos, mas sem a condição elementar e foi resolvido aplicando um algoritmo de rotulagem permanente. Em Subramanian *et al.* (2011) foi proposto o algoritmo *branch-and-cut* com separação preguiçosa aplicado ao PRVCES, sendo que o algoritmo foi testado em 87 instâncias no intervalo de 50 a 200 clientes.

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

Nagy e Salhi (2005) propuseram um mecanismo para a resolução do PRVCES com a restrição de limite de tempo para percorrer cada rota. Essa metodologia reúne diferentes heurísticas para resolver o PRV clássico, tais como, *2-opt*, *3-opt*, realocação, troca, além de procedimentos para tornar a solução viável.

Halse (1992) trata o PRVCES através de uma heurística que consiste em, inicialmente, associar os clientes aos veículos e, em seguida, gerar as rotas utilizando um procedimento baseado no método *3-opt*.

Crispim e Brandão (2005) abordam uma técnica híbrida, juntando as meta-heurísticas Busca Tabu (GLOVER; LAGUNA, 2013) e *Variable Neighborhood Descent* (VND) (HANSEN; MLADENOVIĆ, 2001). Na geração da solução inicial, utilizou-se o método de varredura (*sweep method*) e, para refiná-la, usou-se o procedimento de busca local que explora o espaço de soluções com movimentos de troca e realocação.

Métodos construtivos, heurísticas de refinamento e procedimentos baseados na meta-

heurística Busca Tabu são expostos em Bianchessi e Righini (2007). Esses procedimentos utilizam movimentos de troca de nós (*node-exchange-based*) e troca de arcos (*arc-exchange-based*).

Na literatura, pode-se encontrar alguns algoritmos evolucionários usados para a resolução do problema. Vural (2003) propôs dois Algoritmos Genéticos, sendo que o primeiro é baseado no método de chave aleatória, enquanto o segundo consiste em uma heurística de melhoria que se aplica Movimentos opcionais. Gökçe (2004) criou um algoritmo *Ant Colony* dividido em quatro passos: (i) uma lista de candidatos é formada para cada cliente; (ii) uma solução viável é encontrada e inicial trilhas de feromônio em cada arco são calculadas usando-o; (iii) as rotas são construídas e as trilhas de feromônio são modificadas pela atualização local e global das feromonas; (iv) as rotas são melhoradas usando o movimento 2-opt. Gajpal e Abad (2009) também desenvolveram uma heurística de Colônia de Formigas que tem duas etapas principais: (i) as intensidades de trilha e parâmetros são inicializados usando uma solução inicial obtida por meio de uma heurística construtiva de vizinhança mais próxima; (ii) uma solução de formiga é gerada para cada formiga usando as intensidades de trilha, seguidas de uma busca local em cada formiga-solução e atualização das formigas elitistas e intensidades de trilha.

Wassan *et al.* (2008) propuseram uma versão reativa da meta-heurística Busca Tabu. Na geração de uma solução inicial, foi utilizado o método da varredura chamado de *sweep method* e para explorar o espaço de soluções aplicaram os movimentos de realocação, de troca e de inversão do sentido da rota.

Subramanian (2008) criou um algoritmo *Iterated Local Search* (ILS), com o *Variable Neighborhood Descent* (VND) como procedimento de busca local. Para produzir uma solução inicial foi utilizada uma adaptação da heurística de inserção de Dethloff (2001), mas desconsiderando a capacidade residual do veículo. O VND explora o espaço de soluções usando movimentos baseados em realocação, troca e *crossover*. O VND realiza, a cada melhora na solução corrente, uma intensificação nas rotas alteradas, através de técnicas de busca local *Or-opt*, *2-opt*, *exchange* e *reverse*. O procedimento *Or-opt* que foi implementado consiste em trocar um ou mais clientes consecutivos em uma rota. O 2-opt e o *exchange* fazem a permutação de um par de arcos e dois clientes, respectivamente. O movimento *reverse* consiste em inverter o sentido da rota, caso haja diminuição na carga do veículo nos arcos. As técnicas de perturbação utilizadas no ILS foram o *double swap*, o *double bridge* e o *ejection chain*. O *double swap* consiste em realizar duas permutações consecutivas, o *ejection chain* baseia em deslocar um cliente de cada

rota a outra adjacente e o *double bridge* consiste em retirar quatro arcos e adicionar quatro novos arcos. Além disso, Subramanian (2008) apresentou um novo modelo matemático para o PRVCES.

Em Freitas e Montané (2008), foi apresentada uma hibridização combinando GRASP com VND e outra combinação do VNS com o VND, sendo que em ambos foi utilizado o VND como busca local, neste trabalho foram feitos testes com no mínimo 50 e no máximo 199 clientes.

Zachariadis *et al.* (2009) apresentou uma abordagem híbrida para a resolução do PRVCES, combinando as meta-heurísticas Busca Tabu e *Guided Local Search* (VOUDOURIS; TSANG, 1996). Para comparar as abordagens da literatura, Dethloff (2001) propôs um conjunto com 40 problemas envolvendo 50 clientes cada. Em Salhi e Nagy (1999) apresentaram 28 problemas-teste com 50 a 199 clientes, sendo que a metade desses têm restrições de limite de tempo. Finalmente, Montané e Galvao (2006) adaptaram 18 problemas-teste de Ioachim *et al.* (1995) e Gehring e Homberger (1999), envolvendo 100, 200 e 400 clientes. Montané e Galvao (2006) aplicou a meta-heurística Busca Tabu para solucionar o PRVCES. Neste trabalho foi usado as seguintes estratégias: *Best Improvement*, *First Improvement* e os posteriores movimentos: troca, realocação e *crossover*. Em Mine *et al.* (2010) propôs o algoritmo denominado de GENILS, o mesmo baseia-se nos procedimentos *Iterated Local Search*(ILS), *Variable Neighborhood Descent*(VND) e GENIUS; o algoritmo proposto foi testado utilizando três problemas-teste, o Montané e Galvao (2006), Dethloff (2001) e Salhi e Nagy (1999). Chen (2006) aplicou um procedimento baseado na meta-heurística *Simulated Annealing*(SA) e Busca Tabu. Em Subramanian *et al.* (2010), implementou o método heurístico baseado no *Iterated Local Search*(ILS) e *Random Variable Neighborhood Descent*(RVND), sendo que foram utilizados 256 núcleos de processamento para explorar o espaço de soluções. Silva (2012) propôs o algoritmo GENILS-TS-CL-PR que combina as heurísticas de inserção mais barata, inserção mais barata múltiplas rotas, GENIUS, ILS, Busca Tabu e Reconexão por Caminhos.

O Subramanian *et al.* (2013) implementou o algoritmo híbrido denominado ILS-RVND-SP, o procedimento baseia-se nos métodos *Iterated Local Search*(ILS), *Random Variable Neighborhood Descent*(RVND) e *Set Partitioning*(SP). Para as instâncias de Salhi e Nagy (1999), o algoritmo ILS-RVND-SP se igualou a 21 das melhores soluções conhecidas na literatura e melhorou outras cinco. Já para as instâncias de Montané e Galvao (2006), encontrou 12 das melhores soluções conhecidas na literatura e melhorou outras seis. Adicionalmente o algoritmo obteve bom desempenho para as instâncias de grande porte. A Tabela 1 sumariza os principais

trabalhos relacionados ao PRVCES mencionados nesta seção.

Tabela 1 – Trabalhos relacionados ao PRVCES.

Trabalho	**Ano**	**Abordagem**
Min (1989)	1989	Procedimento baseado em 3 fases
Halse (1992)	1992	Procedimento baseado no método *3-opt*
Salhi e Nagy (1999)	1999	Heurística baseada em inserção
Dethloff (2001)	2001	Heurística de Inserção Mais Barata
Angelelli e Mansini (2002)	2002	*Branch-and-Price*
Vural (2003)	2003	Algoritmo Genético
Crispim e Brandão (2005)	2005	Busca Tabu e VND
Nagy e Salhi (2005)	2005	Heurística baseada em *2-opt*, *3-opt*, realocação, troca
Dell'Amico *et al.* (2006)	2006	*Branch-and-Price*
Montané e Galvao (2006)	2006	Busca Tabu
Bianchessi e Righini (2007)	2007	Métodos construtivos, busca local e Busca Tabu
Wassan *et al.* (2008)	2008	Busca Tabu reativo
Subramanian (2008)	2008	ILS-VND
Freitas e Montané (2008)	2008	GRASP-VND e VNS-VND
Zachariadis *et al.* (2009)	2009	Busca Tabu e *Guided Local Search*
Mine *et al.* (2010)	2010	GENIUS e ILS
Subramanian *et al.* (2010)	2010	ILS e RVND paralelizado
Subramanian *et al.* (2011)	2011	*Branch-and-Cut*
Silva (2012)	2012	GENILS-TS-CL-PR
Subramanian *et al.* (2013)	2013	ILS-RVND-SP

Fonte: Próprio autor.

5 ALGORITMO PROPOSTO

Neste capítulo é apresentado o algoritmo GGVNS proposto para solucionar o PRVCES. Na seção 5.1, mostra-se como uma solução do problema é representada, na seção 5.2, apresenta-se a função utilizada para avaliar as soluções do algoritmo. Na seção 5.3, mostra-se o algoritmo proposto. A seção 5.4 detalha como a solução inicial é obtida. A seção 5.5 explica as estruturas de vizinhanças utilizadas neste trabalho. Finalmente, a seção 5.6 mostra o algoritmo GVNS, utilizado como busca local no algoritmo GGVNS proposto.

5.1 Representação de uma solução

A solução do PRVCES é representada computacionalmente através de um vetor de inteiros. Os clientes são representados por números de 1 a n, sendo n o número de clientes e 0 o depósito. Como uma rota começa e termina no depósito, então 0 é utilizado como separadores de rotas. Para a Figura 14, a solução é representada por:

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

Tal que $[0,7,14,15,4,13,12,0]$, $[0,1,6,16,11,5,18,17,3,0]$ e $[0,10,8,19,9,2,0]$ representam as rotas desta solução.

5.2 Função de avaliação

Uma solução s é avaliada a partir da função f, descrita em (5.1), que determina o custo total de deslocamento.

$$f(s) = \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{5.1}$$

Tal que, c_{ij} é o custo de locomoção de i a j e x_{ij} é uma variável binária que assume valor 1 se na solução s o arco $(i,j) \in A$ for utilizado, e valor 0, caso contrário.

5.3 Algoritmo GGVNS

O algoritmo proposto, nomeado por GGVNS combina as heurísticas *Greedy Randomized Adaptive Search Procedure*(GRASP), *Variable Neighborhood Search*(VNS) e *Variable Neighborhood Search*(VND), assim como o procedimento de inserção mais barata rota a rota.

No algoritmo GGVNS implementado, a solução inicial do GRASP é gerada por meio da heurística de inserção mais barata e sua busca local é feita pelo GVNS. O GVNS possui como busca local o VND. A Figura 15 ilustra o pseudocódigo do algoritmo GGVNS proposto.

Figura 15 – Meta-heurística GGVNS

procedimento GGVNS ($f(.), N(.), MaxIterGVNS, MaxIterGGVNS, \alpha$)
1. $f^* \leftarrow \infty$;
2. $\lambda \leftarrow$ número aleatório no intervalo $[0,1;0,7]$;
3. **para** $Iter = 1,2,...,MaxIterGGVNS$ **faça**
4. $s \leftarrow InsercaoMaisBarata(f(.), \lambda, s)$;
5. $s' \leftarrow GVNS(f(.), N(.), s, MaxIterGVNS)$;
6. **se** $f(s') < f^*$ **então**
7. $s^* \leftarrow s'$;
8. $f^* \leftarrow f(s')$;
9. **fim-se**
10. **fim-para**
11. $s \leftarrow s^*$
12. retorna s;

fim GGVNS.

Fonte: Próprio autor.

No algoritmo descrito na Figura 15, a cada iteração é utilizado o procedimento construtivo, o qual é detalhado na seção 5.4, este procedimento produz uma solução s, que é refinada pelo procedimento GVNS (seção 5.6), gerando uma solução s', permanecendo sempre a solução que melhore a função de avaliação.

5.4 Construção da solução inicial

Para construir uma solução inicial foi usada a heurística de inserção mais barata rota a rota, proposta por Dethloff (2001). Primeiramente, constrói-se uma rota r contendo um cliente selecionado aleatoriamente. Em seguida, calcula-se o custo de inserção e_{ij}^k, definido pela expressão (5.2) de cada cliente k entre os clientes i e j da rota r.

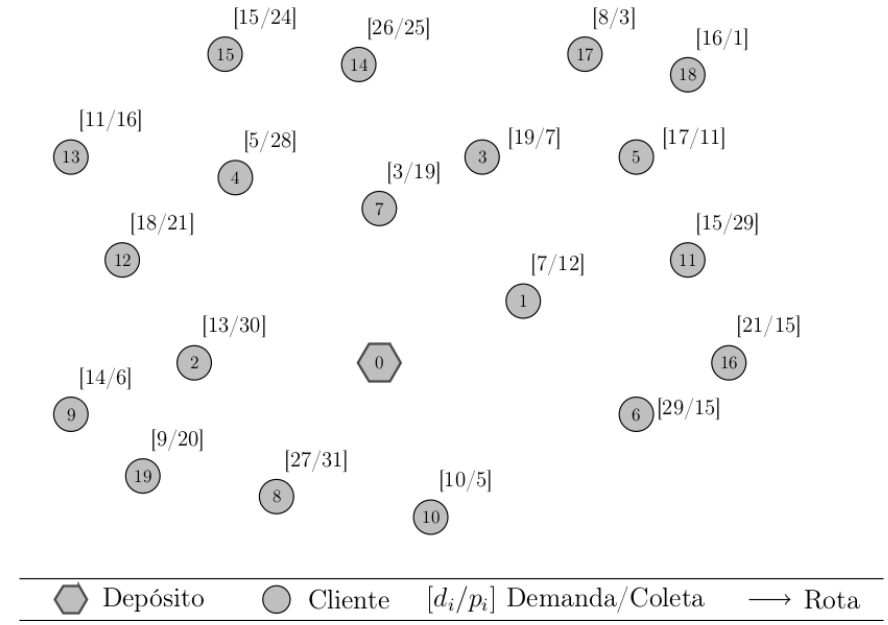
$$e_{ij}^k = (c_{ik} + c_{kj} - c_{ij}) - \lambda(c_{0k} + c_{k0}) \tag{5.2}$$

Nesta equação, a primeira parcela refere-se ao custo do cliente k entre os clientes i e j e a segunda parcela é uma bonificação que se dá um cliente que se encontra distante do

depósito. Essa bonificação é controlada por $\lambda \in [0, 1]$ e favorece a inserção de um cliente distante, de maneira a não adicioná-lo tardiamente à rota.

O cliente com menor custo de inserção é adicionado à rota, desde que a sua inserção não viole a restrição de capacidade do veículo. Caso a inserção de um nó implique em sobrecarga do veículo, então a rota corrente é finalizada e inicia-se a construção em uma nova rota. Este procedimento é repetido até que todos os clientes sejam inseridos à solução. A Figura 16 mostra um exemplo com 19 clientes e um frota ilimitada de veículos com capacidade $Q = 120$.

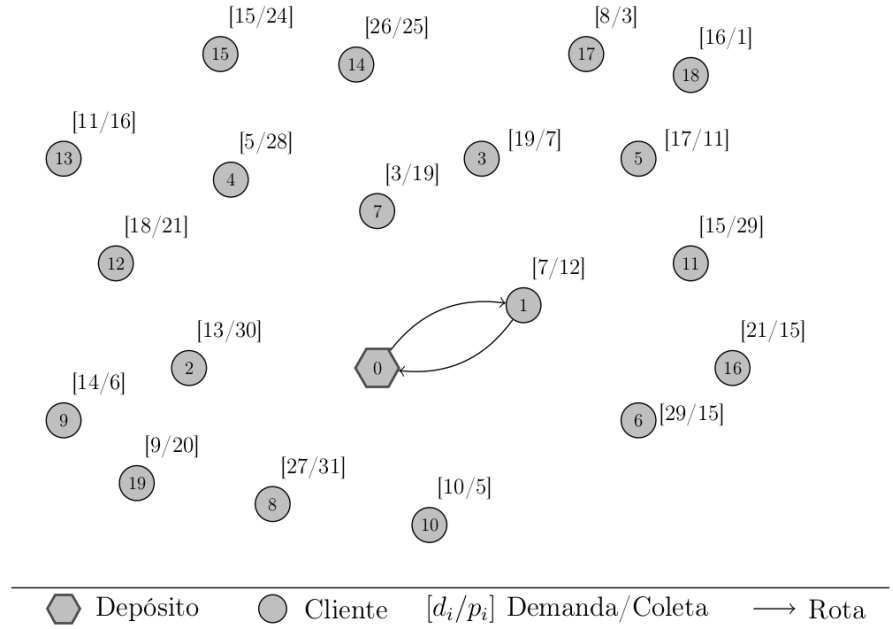
Figura 16 – Exemplo de um problema com 19 clientes.



Fonte: (SILVA, 2012).

Primeiramente gera-se uma rota contendo um cliente escolhido de maneira aleatória, no exemplo da Figura 17, será o cliente 1.

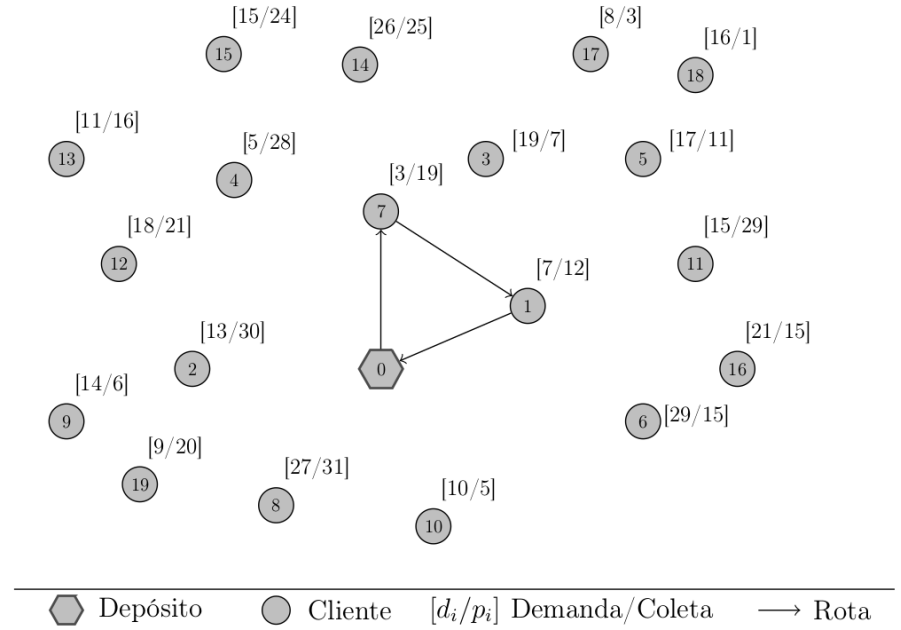
Figura 17 – Construção de uma rota com o cliente 1.



Fonte: (SILVA, 2012).

Depois de escolhido o primeiro cliente, calcula-se o custo de inserção de todos os outros clientes. Nesta figura, o custo de inserção do cliente 7 entre o depósito e o cliente 1 é o menor possível, então deve-se inserir o cliente 7 entre o depósito e o cliente 1. A Figura 18 mostra esta inserção.

Figura 18 – Inserção do cliente 7 na rota.

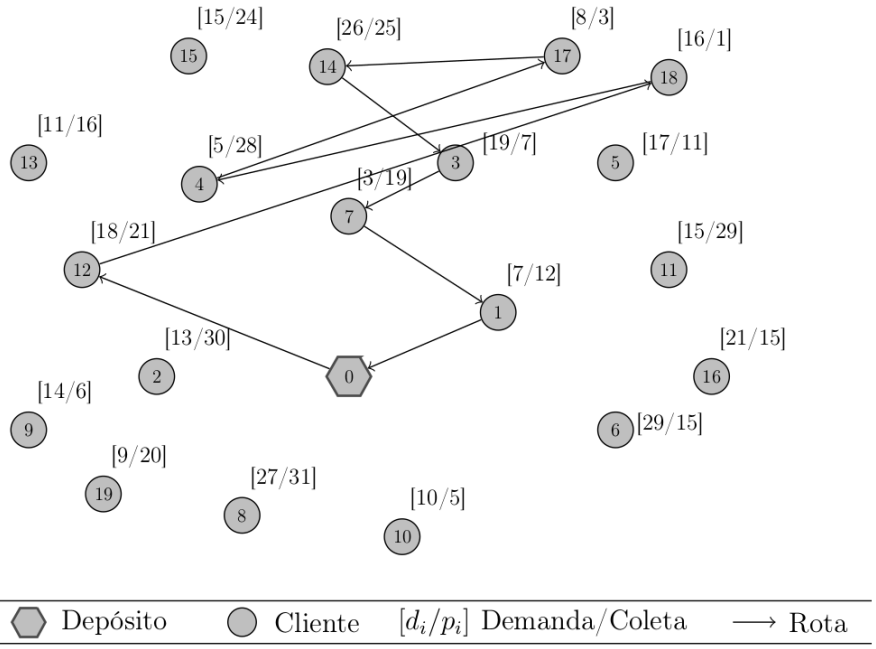


Fonte: (SILVA, 2012).

Prosseguindo com a inserção dos clientes, chega-se ao caso mostrado na Figura 19.

No caso desta Figura, a inserção de qualquer cliente na rota resultará na sobrecarga do veículo.

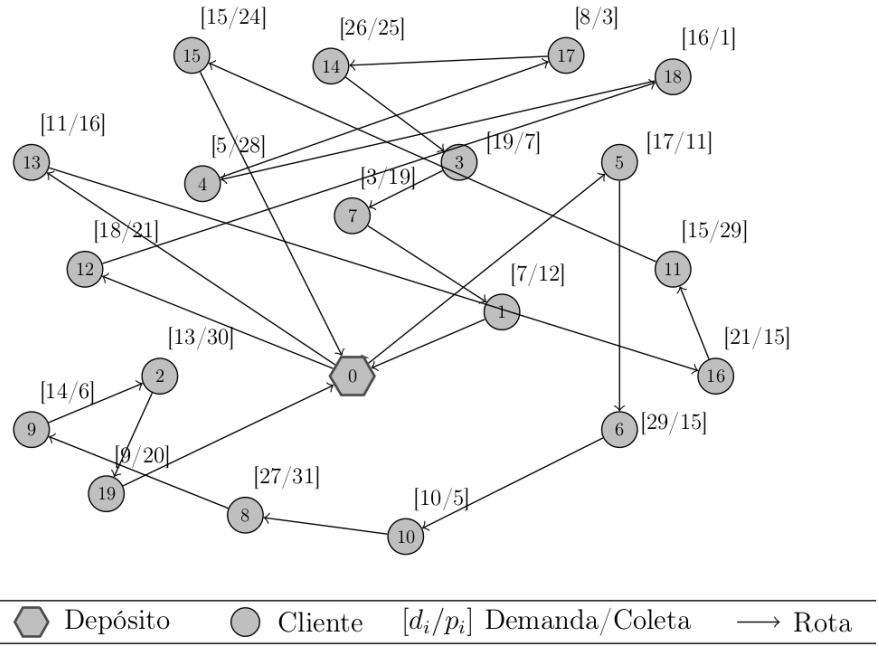
Figura 19 – Inserção de todos os clientes possíveis para a primeira rota.



Fonte: (SILVA, 2012).

Chegando ao caso da Figura 19, a rota é finalizada e passa-se a construir uma nova rota. O procedimento é feito até que todos os clientes sejam adicionados na solução. A Figura 20 mostra a solução atingida pelo procedimento.

Figura 20 – Solução gerada pela heurística de inserção mais barata rota a rota.



Fonte: (SILVA, 2012).

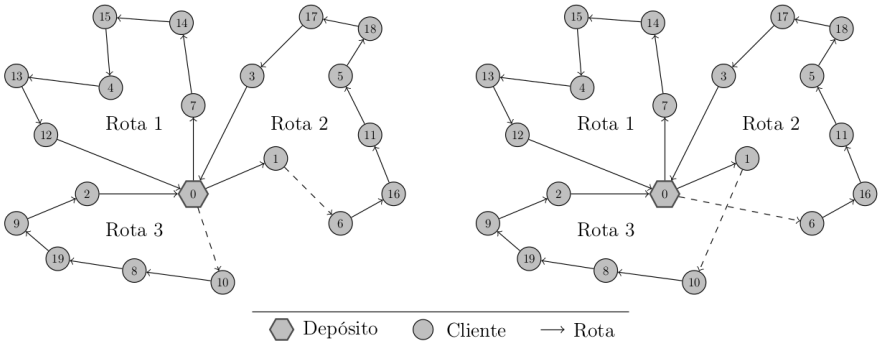
5.5 Estruturas de vizinhança

Na exploração do espaço de vizinhança, foram utilizados seis tipos de estruturas de vizinhança que são descritos a seguir. É interessante destacar que são permitidos apenas os movimentos que não violem as restrições do problema.

5.5.1 2-opt

Este método consiste em retirar dois arcos e adicionar dois novos arcos(SILVA, 2012). A Figura 21 ilustra a remoção dos arcos $(1,6)$ e $(0,10)$ e a adição dos arcos $(1,10)$ e $(0,6)$.

Figura 21 – Exemplo de movimento *2-opt*.

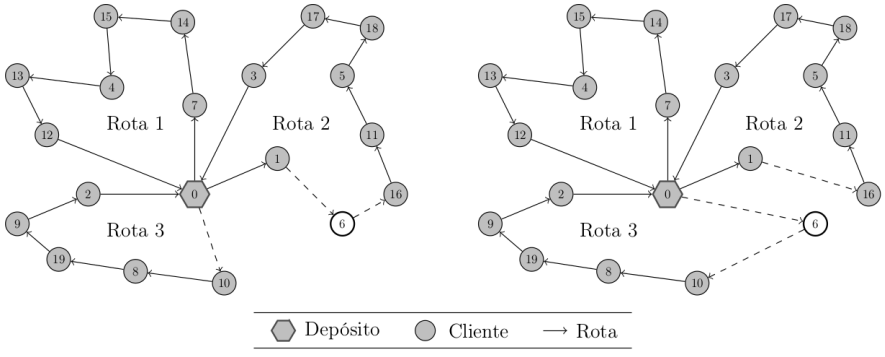


Fonte: (SILVA, 2012).

5.5.2 Shift

O *Shift* é um movimento de realocação que baseia-se em trocar um cliente *i* de uma rota para outra (SILVA, 2012). Na Figura 22, ilustra o cliente 6 sendo transferido da Rota 2 à Rota 3.

Figura 22 – Exemplo de movimento *Shift*.

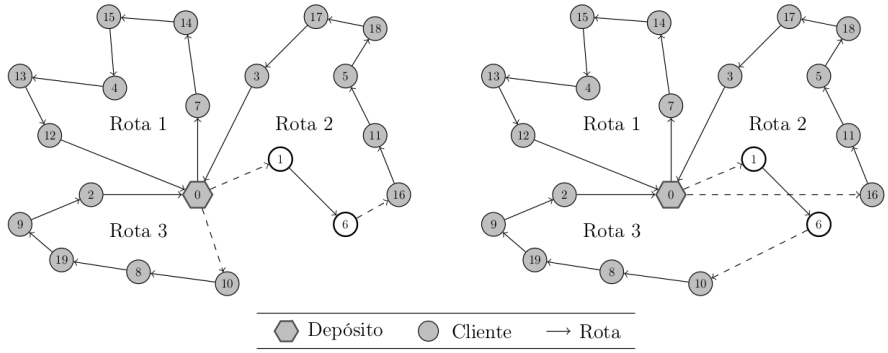


Fonte: (SILVA, 2012)

5.5.3 Shift(2,0)

O *Shift(2,0)* é um movimento similar ao *Shift*, mas realocando dois clientes consecutivos de uma rota para outra. A Figura 23 ilustra a realocação dos clientes 1 e 6 da rota 2 para a rota 3.

Figura 23 – Exemplo de movimento *Shift(2,0)*.

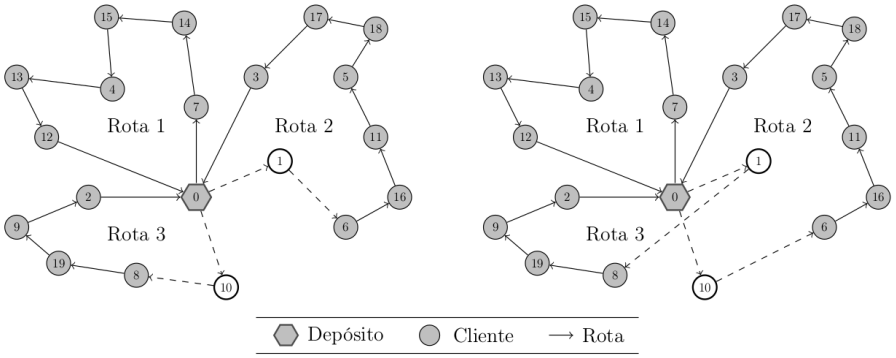


Fonte: (SILVA, 2012)

5.5.4 Swap

O movimento *Swap* consiste em substituir o cliente i de uma rota r_p com outro cliente j de uma rota r_q. A Figura 24 mostra o procedimento aplicado aos clientes 1 e 10 das rotas 2 e 3, respectivamente.

Figura 24 – Exemplo de movimento *Swap*.

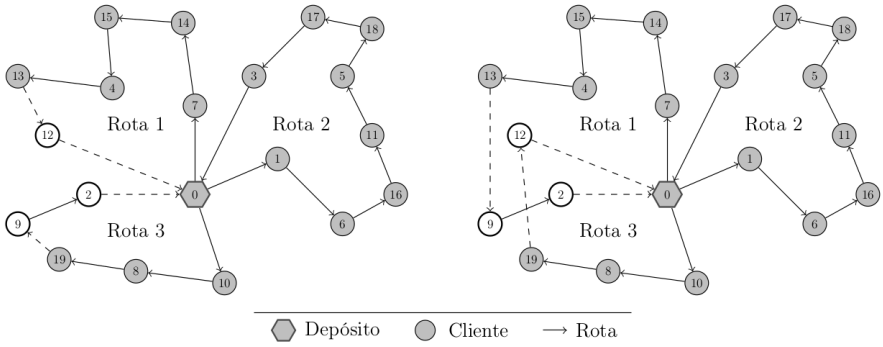


Fonte: (SILVA, 2012)

5.5.5 Swap(2,1)

O *Swap*(2,1) é semelhante ao *Swap*, mas trocando dois clientes consecutivos de uma rota com o cliente de outra. A Figura 25 exemplifica o movimento considerando os clientes 9 e 2 da rota 3 e o cliente 12 da rota 1.

Figura 25 – Exemplo de movimento *Swap(2,1)*.

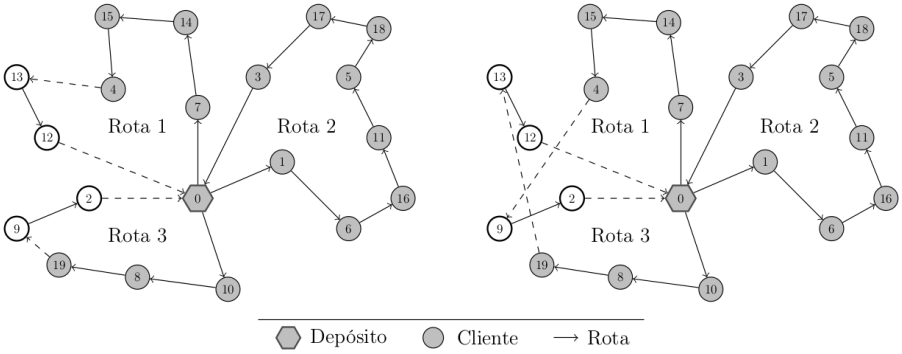


Fonte: (SILVA, 2012)

5.5.6 Swap(2,2)

O *Swap(2,2)* é um movimento parecido com *Swap*, porém com a troca de dois clientes consecutivos de uma rota com dois clientes de outra rota. A Figura 26 apresenta o movimento *Swap*(2,2) aos clientes 13 e 12 da rota 1 e, 9 e 2 da rota 3.

Figura 26 – Exemplo de movimento *Swap(2,2)*.



Fonte: (SILVA, 2012)

5.6 Algoritmo GVNS

A meta-heurística GVNS é um procedimento de busca baseado em vizinhanças variáveis que explora sistematicamente a troca de estruturas de vizinhança, com o objetivo de escapar de soluções locais e encontrar soluções globais(SOUZA, 2008).

A Figura 27 ilustra o pseudocódigo deste procedimento, sendo que o mesmo utiliza os procedimentos de pertubação detalhado na seção 5.6.1 e o método *Variable Neighbourhood Descent*(VND) como busca local, explicado na seção 5.6.2.

Figura 27 – Meta-heurística GVNS proposta.

procedimento GVNS $(f(.), N(.), r, s, MaxIterGVNS)$
1. Selecione um conjunto de vizinhos $N^{(k)}$, para cada $k = 1, ..., r$;
2. $s^* \leftarrow s$;
3. $Iter \leftarrow 1$;
4. **enquanto** $Iter \leq MaxIterGVNS$ **faça**
5. $k \leftarrow 1$;
6. **enquanto** $k \leq r$ **faça**
7. $s \leftarrow Pertubacao()$;
8. Gere um vizinho qualquer $s^{'} \in N^{(k)}(s)$;
9. $s^{''} \leftarrow VND(f(.), N(.), r, s')$;
10. **se** $f(s^{(''')} < f(s)$ **então**
11. $s \leftarrow s^{''}$;
12. $k \leftarrow 1$;
13. **senão**
14. $k \leftarrow k + 1$;
15. **fim-se**
16. **fim-enquanto**
17. $Iter \leftarrow Iter + 1$
18. **fim-enquanto**
19. retorna s;

fim GVNS.

Fonte: Próprio autor.

5.6.1 Perturbações

Para escapar dos ótimos locais, direcionando-se para outros espaços de busca, foi utilizada a pertubação de múltiplos *Swaps* que consiste em selecionar um número k no intervalo [1,5], e aplicar k sucessivos movimentos de *Swap*.

5.6.2 VND

O algoritmo VND proposto baseia-se na implementação de Mine *et al.* (2010) e representa uma adaptação do método VND, da seção 2.3.2.4. O procedimento explora o espaço de soluções utilizando trocas sistemáticas de vizinhanças. As vizinhanças são exploradas em uma ordem pré-estabelecida, e quando uma solução de melhora é encontrada, ela é retornada.

A Figura 28 ilustra o procedimento, em que as estruturas de vizinhanças são ordenadas previamente, na ordem $Shift$, $Shift(2,0)$, $Swap$, $2opt$, $Swap(2,1)$, $Swap(2,2)$, na qual são detalhadas na seção 5.5. Posteriormente, ele verifica se a r-ésima vizinhança é melhor em relação à solução corrente, então nas rotas do vizinho é aplicada uma fase de intensificação.

A intensificação tem como intuito realizar buscas locais nas vizinhanças aplicadas, seguida de buscas locais baseados nos procedimentos. O método é finalizado se a solução corrente não for melhorada com a aplicação da intensificação em nenhuma das vizinhanças.

Figura 28 – Algoritmo VND.

procedimento VND $(f(.), N(.), s)$
1. Seja r a quantidade de vizinhança distintas
2. $k \leftarrow 1$;
3. **enquanto** $(k \leq r)$ **faça**
4. Encontre o melhor vizinho $s' \in N^{(k)}(s)$;
5. **se** $f(s') < f(s)$ **então**
6. $s \leftarrow s'$;
7. {fase de intensificação da rotas s}
8. $k \leftarrow 1$;
9. $s \leftarrow Shift()$
10. $s \leftarrow Shift(2,0)$
11. $s \leftarrow Swap()$
12. $s \leftarrow 2opt()$
13. $s \leftarrow Swap(2,1)$
14. $s \leftarrow Swap(2,2)$
15. **senao**
16. $k \leftarrow k+1$;
17. **fim-se**
18. **fim-enquanto**
19. retorna s;

fim VND.

Fonte: (MINE *et al.*, 2010)

Com base na apresentação do algoritmo proposto, o próximo capítulo apresenta os resultados obtidos com as experimentações realizadas nesta pesquisa.

6 RESULTADOS

O algoritmo GGVNS foi desenvolvido em C++ usando o ambiente de desenvolvimento Visual Studio C++. Pra testá-lo, foi utilizado um computador com processador Intel Dual-core, 1,4 GHz e 4GB de memória RAM e Sistema Operacional Windows 10 com arquitetura de 64 bits. O algoritmo não explora multiprocessamento.

Para validar o algoritmo, foram usadas 40 instâncias de Dethloff (2001); 18 instâncias de Montané e Galvao (2006) e 14 de Salhi e Nagy (1999). Os parâmetros adotados para o algoritmo foram os seguintes: MaxIterGGVNS = 10, MaxIterGVNS = 10000.

Primeiramente, na Seção 6.1, o algoritmo GGVNS é comparado com o algoritmo GENILS, de Mine *et al.* (2010). Posteriormente, na seção 6.2 o algoritmo GGVNS é comparado com o ILS-RVND-SP, melhor algoritmo sequencial conhecido na literatura, proposto por Subramanian *et al.* (2013), neste caso compara-se somente em relação à qualidade da solução, devido aos algoritmos terem sido testados em máquinas distintas.

6.1 GENILS x GGVNS

Esta seção tem como objetivo comparar os resultados referentes ao GGVNS com o GENILS, levando-se em consideração o número de iterações, o número máximo de iterações do algoritmo GENILS foi o mesmo utilizado pelo Mine *et al.* (2010), ou seja, 10000 iterações. Nesses experimentos, pretende-se avaliar a capacidade dos algoritmos encontrar soluções de boa qualidade, levando em consideração o tempo gasto de processamento.

As Tabelas 2, 3 e 4 comparam os resultados obtidos nas instâncias utilizadas. Nestas tabelas, a primeira coluna representa a instância, a coluna N mostra o número de clientes pertencentes à instância, coluna Tempo representa o tempo de processamento do algoritmo em segundos, Melhor é o melhor valor para a função objetivo encontrado pelo respectivo algoritmo. A coluna GAP mostra o desvio percentual do melhor resultado encontrado pelo GGVNS em relação ao melhor encontrado pelo GENILS. O GAP é calculado por 100 x (melhorGENILS - melhorGGVNS) / melhorGGVNS, sendo melhorGENILS, o melhor valor encontrado para o algoritmo GENILS e melhorGGVNS o melhor valor para o algoritmo GGVNS.

Na Tabela 2, é comparado os algoritmos GENILS e GGVNS nas instâncias de Dethloff (2001), observa-se que o GAP foi melhor, ou seja, positivo, para 6 instâncias. E tiveram 32 instâncias em que a qualidade da solução foram iguais, destas 24 tiveram tempo de

processamento menor para o GGVNS.

Tabela 2 – Comparação do GENILS x GGVNS nas instâncias de Dethloff (2001).

Instância	N	GENILS		GGVNS		
		Tempo(s)	Valor	Tempo(s)	Valor	GAP(%)
c0530	**50**	**34.62**	**636.06**	**46.85**	**635.62**	**0.07**
c0531	**50**	**6.58**	**697.84**	**0.72**	**697.84**	**0.00**
c0532	**50**	**8.74**	**659.34**	**1.19**	**659.34**	**0.00**
c0533	**50**	**6.17**	**680.60**	**3.32**	**680.04**	**0.08**
c0534	**50**	**3.77**	**690.50**	**4.19**	**690.50**	**0.00**
c0535	**50**	**2.95**	**659.91**	**0.67**	**659.91**	**0.00**
c0536	**50**	**4.35**	**651.09**	**2.27**	**651.09**	**0.00**
c0537	**50**	**319.20**	**659.17**	**0.69**	**659.17**	**0.00**
c0538	**50**	**2.90**	**719.48**	**0.58**	**719.48**	**0.00**
c0539	**50**	**54.18**	**681.00**	**10.84**	**681.00**	**0.00**
c0580	**50**	**14.63**	**961.50**	**4.35**	**961.50**	**0.00**
c0581	50	9.86	1049.65	6.46	1050.38	-0.07
c0582	**50**	**252.30**	**1039.64**	**63.71**	**1039.64**	**0.00**
c0583	**50**	**2.65**	**983.34**	**2.07**	**983.34**	**0.00**
c0584	**50**	**12.83**	**1065.49**	**1.66**	**1065.49**	**0.00**
c0585	**50**	**3.35**	**1027.08**	**12.66**	**1027.08**	**0.00**
c0586	**50**	**1.38**	**972.49**	**10.37**	**971.82**	**0.07**
c0587	**50**	**13.13**	**1051.28**	**6.52**	**1051.28**	**0.00**
c0588	**50**	**2.41**	**1071.18**	**4.08**	**1071.18**	**0.00**
c0589	**50**	**1.76**	**1061.23**	**3.04**	**1060.50**	**0.07**
cc0530	**50**	**30.99**	**616.52**	**1.70**	**616.52**	**0.00**
cc0531	**50**	**33.57**	**554.47**	**13.97**	**554.47**	**0.00**
cc0532	**50**	**314.25**	**518.01**	**132.13**	**518.01**	**0.00**
cc0533	**50**	**1.97**	**591.19**	**2.22**	**591.19**	**0.00**
cc0534	**50**	**16.17**	**588.79**	**6.66**	**588.79**	**0.00**
cc0535	**50**	**25.87**	**563.70**	**0.98**	**563.70**	**0.00**
cc0536	**50**	**15.78**	**499.05**	**1.48**	**499.05**	**0.00**
cc0537	**50**	**6.17**	**576.48**	**6.03**	**576.48**	**0.00**
cc0538	**50**	**7.65**	**523.05**	**25.82**	**523.05**	**0.00**
cc0539	**50**	**14.59**	**578.25**	**40.68**	**578.25**	**0.00**
cc0580	**50**	**3.24**	**857.17**	**11.82**	**857.17**	**0.00**
cc0581	**50**	**8.38**	**740.93**	**6.91**	**740.85**	**0.01**
cc0582	**50**	**2.04**	**713.44**	**8.14**	**712.89**	**0.08**
cc0583	**50**	**3.69**	**811.07**	**0.34**	**811.07**	**0.00**
cc0584	**50**	**1.94**	**772.25**	**4.06**	**772.25**	**0.00**
cc0585	50	13.52	754.88	5.91	754.95	-0.01
cc0586	**50**	**1.51**	**678.92**	**101.91**	**678.92**	**0.00**
cc0587	**50**	**1.96**	**811.96**	**0.38**	**811.96**	**0.00**
cc0588	**50**	**3.11**	**767.53**	**1.93**	**767.53**	**0.00**
cc0589	**50**	**993.93**	**809.00**	**96.25**	**809.00**	**0.00**

Fonte: Próprio autor.

Analisando os resultados da Tabela 3 que compara os algoritmos GENILS e GGVNS às instâncias de Montané e Galvao (2006), observa-se que o algoritmo GGVNS teve melhor desempenho em 8 instâncias com relação à qualidade da solução. E 4 instâncias que empataram, e destas 2 tiveram o tempo de execução do GGVNS menor.

Tabela 3 – Comparação do GENILS x GGVNS nas instâncias de Montané e Galvao (2006).

Instância	**N**	**GENILS**		**GGVNS**		
		Tempo(s)	**Valor**	**Tempo(s)**	**Valor**	**GAP(%)**
c101	**100**	**70.10**	**1220.99**	**34.92**	**1220.18**	**0.07**
c201	**100**	**5.94**	**662.07**	**1.92**	**662.07**	**0.00**
c1_2_1	200	2737.59	3650.60	800.95	3652.94	-0.06
c1_4_1	400	2755.59	11123.21	4203.81	11175.44	-0.47
c2_2_1	200	2737.80	1730.61	1290.49	1731.96	-0.08
c2_4_1	**400**	**2755.45**	**3682.26**	**3666.48**	**3646.08**	**0.99**
r1_2_1	**200**	**2735.37**	**3384.99**	**4805.89**	**3374.52**	**0.31**
r1_4_1	**400**	**2758.55**	**9771.49**	**25265.84**	**9672.49**	**1.02**
r2_2_1	200	2739.22	1693.97	725.30	1751.52	-3.29
r2_4_1	**400**	**3729.57**	**3793.48**	**3729.57**	**3793.48**	**0.00**
r101	**100**	**2736.72**	**1018.95**	**153.36**	**1009.95**	**0.89**
r201	**100**	**2735.83**	**675.51**	**447.85**	**672.39**	**0.46**
rc1_2_1	**200**	**2740.50**	**3348.73**	**1487.17**	**3335.39**	**0.40**
rc1_4_1	400	2757.89	9646.80	2283.48	9724.52	-0.80
rc2_2_1	200	2766.91	1583.87	719.38	1641.23	-3.49
rc2_4_1	**400**	**2746.25**	**3676.36**	**2448.20**	**3593.40**	**2.31**
rc101	**100**	**35.85**	**1059.75**	**15.98**	**1059.75**	**0.00**
rc201	**100**	**579.00**	**672.92**	**866.07**	**672.92**	**0.00**

Fonte: Próprio autor.

A Tabela 4 mostra os resultados dos algoritmos GENILS e GGVNS para as instâncias de Salhi e Nagy (1999), observa-se uma melhora na qualidade da solução de 8 instâncias, e empate em 2, sendo que todos os empates tiveram melhora no tempo de execução do GGVNS.

Tabela 4 – Comparação do GENILS x GGVNS nas instâncias de Salhi e Nagy (1999).

Instância	N	GENILS		GGVNS		
		Tempo(s)	Valor	Tempo(s)	Valor	GAP(%)
sn1x	**50**	**104.49**	**466.77**	**14.92**	**466.77**	**0.00**
sn1y	**50**	**54.22**	**466.77**	**26.48**	**466.77**	**0.00**
sn2x	**75**	**2741.41**	**690.89**	**136.83**	**686.50**	**0.64**
sn2y	**75**	**2745.55**	**688.05**	**149.64**	**684.21**	**0.56**
sn3x	**100**	**148.39**	**721.40**	**221.23**	**721.27**	**0.02**
sn3y	100	2753.52	723.28	193.40	729.13	-0.80
sn4x	**150**	**2761.12**	**858.95**	**322.67**	**857.79**	**0.14**
sn4y	150	2882.39	864.20	314.12	865.14	-0.11
sn5x	199	2826.03	1035.35	476.37	1053.80	-1.75
sn5y	**199**	**2756.61**	**1056.37**	**499.75**	**1054.81**	**0.15**
sn11x	**120**	**2777.49**	**880.81**	**318.00**	**875.06**	**0.66**
sn11y	**120**	**2848.72**	**929.13**	**321.02**	**876.36**	**6.02**
sn12x	100	2742.55	663.50	168.60	673.72	-1.52
sn12y	**100**	**2751.58**	**669.96**	**181.09**	**663.50**	**0.97**

Fonte: Próprio autor.

Com relação aos resultados obtidos, ao comparar os algoritmos GENILS e GGVNS, constatou-se que o algoritmo GGVNS teve bom desempenho, ou seja, empatou ou melhorou a qualidade de sua solução, em 83.3% das instâncias testadas. Das instâncias que empataram, a maioria destas conseguiram melhora com relação ao tempo de processamento.

6.2 ILS-RVND-SP x GGVNS

Esta seção tem o objetivo de comparar o algoritmo ILS-RVND-SP de Subramanian *et al.* (2013), no caso leva-se em consideração apenas a qualidade das soluções obtidas, pois os algoritmos foram testados em máquinas distintas. O algoritmo ILS-RVND-SP foi testado em uma máquina com o processador Intel Core i7 com 2.93GHz e 8GB de RAM no sistema operacional *Ubuntu Linux* 64 bits. As Tabelas 5 e 6 representam, respectivamente, os resultados dos dois algoritmos nos problemas-testes de Montané e Galvao (2006) e Salhi e Nagy (1999). A coluna GAP destas tabelas, é calculado por 100 x (melhorILS_RVND_SP - melhorGGVNS) / melhorGGVNS, sendo melhorILS_RVND_SP, o melhor valor encontrado para o algoritmo ILS-RVND-SP e melhorGGVNS o melhor valor para o algoritmo GGVNS.

Ao analisar a Tabela 5, percebe-se que nenhuma das instâncias obtiveram melhoras na qualidade da solução, entretanto 4 instâncias tiveram o GAP igual a zero, ou seja, os dois algoritmos empataram, levando-se em consideração a qualidade da solução obtida.

Tabela 5 – Comparação do ILS-RVND-SP x GGVNS nas instâncias de Montané e Galvao (2006).

Instância	N	ILS-RVND-SP		GGVNS		
		Tempo(s)	Valor	Tempo(s)	Valor	GAP(%)
c101	**100**	**12.93**	**1220.18**	**34.92**	**1220.18**	**0.00**
c201	**100**	**9.77**	**662.07**	**1.92**	**662.07**	**0.00**
c1_2_1	200	2874.50	3629.89	800.95	3652.94	-0.63
c1_4_1	400	8016.83	11047.19	4203.81	11175.44	-1.15
c2_2_1	200	1365.93	1726.59	1290.49	1731.96	-0.31
c2_4_1	400	10691.30	3539.50	3666.48	3646.08	-2.92
r1_2_1	200	1142.00	3353.00	4805.89	3374.52	-0.64
r1_4_1	400	9177.90	9519.45	25265.84	9672.49	-1.58
r2_2_1	200	1425.88	1665.58	725.30	1751.52	-4.91
r2_4_1	400	2874.50	3628.51	3729.57	3793.48	-4.35
r101	**100**	**65.42**	**1009.95**	**153.36**	**1009.95**	**0.00**
r201	100	15.71	666.20	447.85	672.39	-0.92
rc1_2_1	200	1293.53	3303.70	1487.17	3335.39	-0.95
rc1_4_1	400	10867.10	9447.53	2283.48	9724.52	-2.85
rc2_2_1	200	1361.87	1560.00	719.38	1641.23	-4.95
rc2_4_1	400	8326.18	3403.70	2448.20	3593.40	-5.28
rc101	100	16.89	1059.32	15.98	1059.75	-0.04
rc201	**100**	**11.42**	**672.92**	**866.07**	**672.92**	**0.00**

Fonte: Próprio autor.

Ao analisar a Tabela 6, observa-se que 4 empataram com relação à qualidade de sua solução. E as outras instâncias desta tabela, teve piora na solução gerada.

Tabela 6 – Comparação do ILS-RVND-SP x GGVNS nas instâncias de Salhi e Nagy (1999).

Instância	N	ILS-RVND-SP		GGVNS		
		Tempo(s)	Valor	Tempo(s)	Valor	GAP(%)
sn1x	**50**	**2.08**	**466.77**	**14.92**	**466.77**	**0.00**
sn1y	**50**	**1.97**	**466.77**	**26.48**	**466.77**	**0.00**
sn2x	75	12.79	684.21	136.83	686.50	-0.33
sn2y	**75**	**10.83**	**684.21**	**149.64**	**684.21**	**0.00**
sn3x	**100**	**17.69**	**721.27**	**221.23**	**721.27**	**0.00**
sn3y	100	17.61	721.27	193.40	729.13	-1.08
sn4x	150	98.03	852.46	322.67	857.79	-0.62
sn4y	150	80.63	852.46	314.12	865.14	-1.47
sn5x	199	1786.74	1029.25	476.37	1053.80	-2.33
sn5y	199	1726.18	1029.25	499.75	1054.81	-2.42
sn11x	120	51.82	846.23	318.00	875.06	-3.29
sn11y	120	48.63	846.23	321.02	876.36	-3.44
sn12x	100	9.07	662.22	168.60	673.72	-1.71
sn12y	100	9.34	662.22	181.09	663.50	-0.19

Fonte: Próprio autor.

Ao analisar os resultados obtidos pela comparação dos algoritmos ILS-RVND-SP e GGVNS, descobre-se que 25% das instâncias empataram com relação a qualidade da solução. Entretanto este algoritmo foi testado apenas nos problemas-testes de Montané e Galvao (2006) e Salhi e Nagy (1999).

7 CONCLUSÕES E TRABALHOS FUTUROS

O PRVCES é um problema NP-difícil, tornando-se difícil solucionar instâncias grandes em tempo computacionalmente aceitável. Com isso, abordagens heurísticas são mais usadas para resolvê-lo. Desta forma, implementou-se o algoritmo heurístico híbrido GGVNS que combina os métodos heurísticos: *Greedy Randomized Adaptative Search Procedure*(GRASP), *Variable Neighbourhood Search*(VNS) e *Variable Neighbourhood Descent*(VND). Na fase de construção do GRASP, uma solução do problema é gerada utilizando o método de inserção mais barata e, na fase de busca local é aplicado o procedimento GVNS. Neste procedimento, aplica-se na fase de busca local o método heurístico VND.

O algoritmo proposto foi testado em um conjunto de instâncias consagradas na literatura. Foram realizadas duas análises, na primeira (seção 6.1) testou-se o algoritmo GENILS, de Mine *et al.* (2010), com o algoritmo proposto, neste caso, o algoritmo GGVNS melhorou em 22 instâncias e empatou com 38 instâncias de 72. Na segunda análise, testou-se o algoritmo ILS-RVND-SP, proposto por Subramanian *et al.* (2013), com o algoritmo abordado nesta pesquisa, neste caso, o algoritmo GGVNS empatou em 8 instâncias de 32.

Como trabalho futuro, sugere-se aplicar o *framework Iterated Racing for Automatic Algorithm Configuration*(IRACE)(LÓPEZ-IBÁÑEZ *et al.*, 2016) para calibrar os parâmetros do procedimento, tal como paralelizar o algoritmo aproveitando-se do número de núcleos de processamento existentes nos computadores atuais. Além disso, torna-se plausível utilizar outras estruturas de vizinhança e outras heurísticas de refinamento dentro do algoritmo GGVNS, como também outros mecanismos de pertubação para o mesmo.

REFERÊNCIAS

ALCANTARA, N. L. J. L. **Introdução à Teoria dos Grafos**. 2014. Disponível em: <https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwj4uZen5N_bAhUGf5AKHQQkAtMQjRx6BAgBEAQ&url=https%3A%2F%2Fpt.slideshare.net%2Fnathannlucas%2F09-grafos&psig=AOvVaw0ZwYyNnxWWKO4AHHetBKzv&ust=1529499637531047>.

ANGELELLI, E.; MANSINI, R. **Quantitative Approaches to Distribution Logistics and Supply Chain Management, chapter A branch-and-price algorithm for a simultaneous pick-up and delivery problem**. [S.l.]: Springer, Berlin-Heidelberg, 2002.

BAKER, E. K. Evolution of microcomputer-based vehicle routing software: Case studies in the united states. In: **The vehicle routing problem**. [S.l.]: SIAM, 2002. p. 353–361.

BALAS, E.; FISCHETTI, M.; PULLEYBLANK, W. R. The precedence-constrained asymmetric traveling salesman polytope. **Mathematical Programming**, Springer, v. 68, n. 1-3, p. 241–265, 1995.

BEASLEY, J. Route first—cluster second methods for vehicle routing. **Omega**, v. 11, n. 4, p. 403 – 408, 1983. ISSN 0305-0483. Disponível em: <http://www.sciencedirect.com/science/article/pii/0305048383900336>.

BELFIORE, P. P.; YOSHIZAKI, H. T. Y. Scatter search para problemas de roteirização de veículos com frota heterogênea, janelas de tempo e entregas fracionadas. **Production**, SciELO Brasil, v. 16, n. 3, p. 455–469, 2006.

BERALDI, P.; GHIANI, G.; LAPORTE, G.; MUSMANNO, R. Efficient neighborhood search for the probabilistic pickup and delivery travelling salesman problem. **Networks**, Wiley Online Library, v. 45, n. 4, p. 195–198, 2005.

BIANCHESSI, N.; RIGHINI, G. Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. **Computers & Operations Research**, Elsevier, v. 34, n. 2, p. 578–594, 2007.

BIRBIL, Ş. İ.; FANG, S.-C.; SHEU, R.-L. On the convergence of a population-based global optimization algorithm. **Journal of global optimization**, Springer, v. 30, n. 2-3, p. 301–318, 2004.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM computing surveys (CSUR)**, ACM, v. 35, n. 3, p. 268–308, 2003.

BOLLOBÁS, B. Graph theory: An introductory course. 1979. **Springer-Verlag, New York**, 1979.

BONDY, J. A.; MURTY, U. S. R. *et al.* **Graph theory with applications**. [S.l.]: Citeseer, 1976. v. 290.

CABRAL, L. Introdução à pesquisa operacional(notas de aula). 1996.

CAMPOS, V.; MOTA, E. Heuristic procedures for the capacitated vehicle routing problem. **Computational Optimization and Applications**, Springer, v. 16, n. 3, p. 265–277, 2000.

CARABETTI, E. G.; SOUZA, S.; FRAGA, M. C. Metaheurística colônia de formiga aplicada ao problema de roteamento de veículos com coleta e entrega e janela de tempo. **Anais do XIV Simpósio de Pesquisa Operacional e Logística da Marinha, Escola de Guerra Naval, Urca–RJ**, 2010.

CASTRO, L. N. D.; ZUBEN, F. J. V. Learning and optimization using the clonal selection principle. **IEEE transactions on evolutionary computation**, IEEE, v. 6, n. 3, p. 239–251, 2002.

CHEN, J.-F. Approaches for the vehicle routing problem with simultaneous deliveries and pickups. **Journal of the Chinese Institute of Industrial Engineers**, Taylor & Francis, v. 23, n. 2, p. 141–150, 2006.

COSTA, D. L. V. *et al.* Uma abordagem heurística para o problema de roteamento dial-a-ride. Universidade Federal da Paraíba, 2013.

CRISPIM, J.; BRANDÃO, J. Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. **Journal of the Operational Research Society**, Springer, v. 56, n. 11, p. 1296–1302, 2005.

CZYZŻAK, P.; JASZKIEWICZ, A. Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. **Journal of Multi-Criteria Decision Analysis**, Wiley Online Library, v. 7, n. 1, p. 34–47, 1998.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. **Management science**, Informs, v. 6, n. 1, p. 80–91, 1959.

DELL'AMICO, M.; RIGHINI, G.; SALANI, M. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. **Transportation science**, INFORMS, v. 40, n. 2, p. 235–247, 2006.

DETHLOFF, J. Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. **OR-Spektrum**, Springer, v. 23, n. 1, p. 79–96, 2001.

FAVARO, F. F. A teoria dos grafos e sua abordagem na sala de aula com recursos educacionais digitais. Universidade Estadual Paulista (UNESP), 2017.

FÁVERO, L.; BELFIORE, P. **Pesquisa Operacional para Cursos de Administração**. Elsevier Editora Ltda., 2012. ISBN 9788535249118. Disponível em: <https://books.google.com.br/books?id=spbIUnIPwn8C>.

FEO, T. A.; RESENDE, M. G. Greedy randomized adaptive search procedures. **Journal of global optimization**, Springer, v. 6, n. 2, p. 109–133, 1995.

FREITAS, L. M. B. de; MONTANÉ, F. A. T. Metaheurísticas vns-vnd e grasp-vnd para problemas de roteamento de veículos com coleta e entrega simultânea. **XI Simpósio de Pesquisa Operacional e Logística da Marinha, Rio de Janeiro**, 2008.

GAJPAL, Y.; ABAD, P. An ant colony system (acs) for vehicle routing problem with simultaneous delivery and pickup. **Comput. Oper. Res.**, Elsevier Science Ltd., Oxford, UK, UK, v. 36, n. 12, p. 3215–3223, dez. 2009. ISSN 0305-0548. Disponível em: <http://dx.doi.org/10.1016/j.cor.2009.02.017>.

GEHRING, H.; HOMBERGER, J. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: CITESEER. **Proceedings of EUROGEN99**. [S.l.], 1999. v. 2, p. 57–64.

GLOVER, F.; LAGUNA, M. Tabu search. In: **Handbook of combinatorial optimization**. [S.l.]: Springer, 2013. p. 3261–3362.

GLOVER, F. W.; KOCHENBERGER, G. A. **Handbook of metaheuristics**. [S.l.]: Springer Science & Business Media, 2006. v. 57.

GÖKÇE, E. İ. A revised ant colony system approach to vehicle routing problems. **Masther's Thesis, Graduate School of Engineering and Natural Sciences, Sabanci University, Turkey**, 2004.

GOLDEN, B. L.; ASSAD, A. A.; WASIL, E. A. Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries. In: **The vehicle routing problem**. [S.l.]: SIAM, 2002. p. 245–286.

HALSE, K. **Modeling and solving complex vehicle routing problems**. Tese (Doutorado) — Technical University of Denmark, 1992.

HANSEN, P.; MLADENOVIĆ, N. Variable neighborhood search: Principles and applications. **European journal of operational research**, Elsevier, v. 130, n. 3, p. 449–467, 2001.

HERNÁNDEZ-PÉREZ, H.; SALAZAR-GONZÁLEZ, J.-J. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. **Discrete Applied Mathematics**, Elsevier, v. 145, n. 1, p. 126–139, 2004.

HO, S. C.; HAUGLAND, D. Local search heuristics for the probabilistic dial-a-ride problem. Citeseer, 2004.

HOFFMAN, K. L.; PADBERG, M.; RINALDI, G. Traveling salesman problem. In: **Encyclopedia of operations research and management science**. [S.l.]: Springer, 2013. p. 1573–1578.

IOACHIM, I.; DESROSIERS, J.; DUMAS, Y.; SOLOMON, M. M.; VILLENEUVE, D. A request clustering algorithm for door-to-door handicapped transportation. **Transportation science**, INFORMS, v. 29, n. 1, p. 63–78, 1995.

KORTE, B.; VYGEN, J.; KORTE, B.; VYGEN, J. **Combinatorial optimization**. [S.l.]: Springer, 2012. v. 2.

LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; Pérez Cáceres, L.; STÜTZLE, T.; BIRATTARI, M. The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, v. 3, p. 43–58, 2016.

MIN, H. The multiple vehicle routing problem with simultaneous delivery and pick-up points. **Transportation Research Part A: General**, Elsevier, v. 23, n. 5, p. 377–386, 1989.

MINE, M. T.; SILVA, M. d. S. A.; OCHI, L. S.; SOUZA, M. J. F.; SILVA, T. C. B. da. O problema de roteamento de veículos com coleta e entrega simultânea: uma abordagem via iterated local search e genius. **TRANSPORTES**, v. 18, n. 3, 2010.

MONTANÉ, F. A. T.; GALVAO, R. D. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. **Computers & Operations Research**, Elsevier, v. 33, n. 3, p. 595–619, 2006.

MOSHEIOV, G. The travelling salesman problem with pick-up and delivery. **European Journal of Operational Research**, Elsevier, v. 79, n. 2, p. 299–310, 1994.

NAGY, G.; SALHI, S. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. **European journal of operational research**, Elsevier, v. 162, n. 1, p. 126–141, 2005.

NEŠETRIL, J.; NEŠETRILOVÁ, H. The origins of minimal spanning tree algorithms–boruvka and jarnık. **Documenta Mathematica, vol. Extra Volume ISMP**, p. 127–141, 2012.

NIE, C.; LEUNG, H. A survey of combinatorial testing. **ACM Computing Surveys (CSUR)**, ACM, v. 43, n. 2, p. 11, 2011.

PARRAGH, S. N.; DOERNER, K. F.; HARTL, R. F. A survey on pickup and delivery problems. **Journal für Betriebswirtschaft**, Springer, v. 58, n. 1, p. 21–51, 2008.

PSARAFTIS, H. N. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. **Transportation Science**, INFORMS, v. 14, n. 2, p. 130–154, 1980.

RENAUD, J.; BOCTOR, F. F.; LAPORTE, G. Perturbation heuristics for the pickup and delivery traveling salesman problem. **Computers & Operations Research**, Elsevier, v. 29, n. 9, p. 1129–1141, 2002.

RIBEIRO, C. C. Metaheuristics and applications. **Advanced School on Artificial Intelligence, Estoril, Portugal**, 1996.

ROUSSEAU, L.-M.; GENDREAU, M.; PESANT, G. Using constraint-based operators to solve the vehicle routing problem with time windows. **Journal of heuristics**, Springer, v. 8, n. 1, p. 43–58, 2002.

RUOHONEN, K. Graph theory. tampereen teknillinen yliopisto. originally titled graafiteoria, lecture notes translated by tamminen, j., lee, k. **C. and Piché, R**, 2013.

SALHI, S.; NAGY, G. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. **Journal of the operational Research Society**, Springer, v. 50, n. 10, p. 1034–1042, 1999.

SAVELSBERGH, M. W.; SOL, M. The general pickup and delivery problem. **Transportation science**, INFORMS, v. 29, n. 1, p. 17–29, 1995.

SILVA, T. C. B. d. Genils-ts-cl-pr: um algoritmo heurístico para resolução do problema de roteamento de veículos com coleta e entrega simultânea. Programa de Pós-Graduação em Ciência da Computação. Departamento de Computação, Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto., 2012.

SOUZA, M. Inteligência computacional para otimização (notas de aula). **Ouro Preto: UFOP. Disponível em:< http://www. iceb. ufop. br/prof/marcone>. Acesso em**, v. 22, 2008.

SUBRAMANIAN, A. **Metaheurística Iterated Local Search aplicada ao problema de roteamento de veículos com coleta e entrega simultânea**. Tese (Doutorado) — Dissertação de mestrado, Universidade Federal da Paraíba, João Pessoa, 2008.

SUBRAMANIAN, A.; DRUMMOND, L. M. d. A.; BENTES, C.; OCHI, L. S.; FARIAS, R. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. **Computers & Operations Research**, Elsevier, v. 37, n. 11, p. 1899–1911, 2010.

SUBRAMANIAN, A.; UCHOA, E.; OCHI, L. S. A hybrid algorithm for a class of vehicle routing problems. **Computers & Operations Research**, Elsevier, v. 40, n. 10, p. 2519–2531, 2013.

SUBRAMANIAN, A.; UCHOA, E.; PESSOA, A. A.; OCHI, L. S. Branch-and-cut with lazy separation for the vehicle routing problem with simultaneous pickup and delivery. **Operations Research Letters**, Elsevier, v. 39, n. 5, p. 338–341, 2011.

TOTH, P.; VIGO, D. The vehicle routing problem, ser. siam monographs on discrete mathematics and applications. **Society for Industrial and Applied Mathematics**, 2002.

VANSTEENWEGEN, P.; SOUFFRIAU, W.; BERGHE, G. V.; OUDHEUSDEN, D. V. A guided local search metaheuristic for the team orienteering problem. **European journal of operational research**, Elsevier, v. 196, n. 1, p. 118–127, 2009.

VOSS, S.; MARTELLO, S.; OSMAN, I. H.; ROUCAIROL, C. **Meta-heuristics: Advances and trends in local search paradigms for optimization**. [S.l.]: Springer Science & Business Media, 2012.

VOUDOURIS, C.; TSANG, E. Partial constraint satisfaction problems and guided local search. **Proc., Practical Application of Constraint Technology (PACT'96), London**, p. 337–356, 1996.

VURAL, A. V. **A GA based meta-heuristic for capacitated vehicle routing problem with simultaneous pick-up and deliveries**. Tese (Doutorado), 2003.

WASSAN, N. A.; WASSAN, A. H.; NAGY, G. A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. **Journal of combinatorial optimization**, Springer, v. 15, n. 4, p. 368–386, 2008.

WEST, D. B. *et al.* **Introduction to graph theory**. [S.l.]: Prentice hall Upper Saddle River, 2001. v. 2.

XU, H.; CHEN, Z.-L.; RAJAGOPAL, S.; ARUNAPURAM, S. Solving a practical pickup and delivery problem. **Transportation science**, INFORMS, v. 37, n. 3, p. 347–364, 2003.

ZACHARIADIS, E. E.; TARANTILIS, C. D.; KIRANOUDIS, C. T. A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. **Expert Systems with applications**, Elsevier, v. 36, n. 2, p. 1070–1081, 2009.