ALBA ASSIS CAMPOS

O PROBLEMA DE ROTEAMENTO DE VEÍCULOS PARA COLETA DE LIXO COM JANELAS DE TEMPO: ABORDAGEM HEURÍSTICA

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS – BRASIL
2018

Ficha catalográfica preparada pela Biblioteca Central da Universidade Federal de Viçosa - Câmpus Viçosa

T
C198p
2018

Campos, Alba Assis, 1991-
O problema de roteamento de veículos para coleta de lixo com janelas de tempo : abordagem heurística / Alba Assis Campos. – Viçosa, MG, 2018.
xi, 69 f. : il. (algumas color.) ; 29 cm.

Inclui apêndice.
Orientador: José Elias Claudio Arroyo.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f. 67-69.

1. Pesquisa operacional. 2. Otimização combinatória. 3. Heurística. 4. Lixo - Eliminação - Transporte. I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. II. Título.

CDD 22. ed. 003

ALBA ASSIS CAMPOS

O PROBLEMA DE ROTEAMENTO DE VEÍCULOS PARA COLETA DE LIXO COM JANELAS DE TEMPO: ABORDAGEM HEURÍSTICA

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 30 de janeiro de 2018.

André Gustavo dos Santos

Marcos Henrique Fonseca Ribeiro

José Elias Cláudio Arroyo
(Orientador)

Dedico este trabalho a Deus e a minha família.

"A persistência é o caminho do êxito."
(Charles Chaplin)

Agradecimentos

Agradeço primeiramente a Deus pela minha vida, pela minha saúde e pela minha família.

A minha mãe Alba Valéria e a minha irmã Natália, pelo apoio e incentivo incondicional em todos os momentos da minha vida.

A minha avó Ivone, pelo amor e carinho que me deu em vida e sei que onde estiver estará feliz por mim. E a todos os familiares pelas palavras de incentivo.

Ao meu namorado Cassiano, pela sua ajuda e dedicação.

Agradeço a todos os amigos do Departamento de Informática.

Agradeço a todos os professores do Departamento de Informática da UFV que contribuíram em minha formação. Especialmente, ao meu orientador José Elias Arroyo e ao professor André Gustavo dos Santos pela dedicação, atenção, paciência e pelos ensinamentos.

Agradeço ao financiamento fornecido pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), ao qual foi essencial para realização e conclusão desse trabalho.

E finalmente, a todos aqueles que, direta ou indiretamente, contribuíram para a execução desse trabalho, os meus sinceros agradecimentos.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

CAMPOS, Alba Assis, M.Sc., Universidade Federal de Viçosa, janeiro de 2018. **O Problema de Roteamento de Veículos para Coleta de Lixo com Janelas de Tempo: Abordagem heurística.** Orientador: José Elias Claudio Arroyo.

Este trabalho aborda o Problema de Roteamento de Veículos para Coleta de Lixo com Janelas de Tempo (WCVRPTW), cujo objetivo é encontrar as rotas para os veículos coletores de lixo de modo que todos os clientes sejam plenamente atendidos e a distância total percorrida pelos veículos seja a menor possível, minimizando assim os custos de transporte. Para alcançar este objetivo é necessário que algumas restrições sejam atendidas: os clientes devem ser atendidos dentro de um período de tempo; existe horário de saída e retorno dos veículos ao depósito; os veículos possuem restrições de capacidade; as rotas possuem restrições de volume de carregamento e número de clientes atendidos; os veículos devem sair e retornar vazios ao depósito, e, quando cheios, devem ir para o aterro sanitário mais próximo para descarga do lixo. Além disso, os motoristas dos veículos devem realizar uma parada de almoço. O WCVRPTW é um problema real que pertence à classe NP-difícil. Para resolvê-lo, são desenvolvidos quatro algoritmos heurísticos: *Simulated Annealing* (SA), *Tabu Search* (TS), e dois algoritmos híbridos baseados nas metaheurísticas *Iterated Local Search* (ILS) e *Variable Neighborhood Descent* (VND), denominados ILS-VND e ILS-RVND. Os desempenhos dos algoritmos propostos são analisados em instâncias de pequeno e médio porte geradas para este trabalho, e também em instâncias de grande porte disponíveis na literatura. Os experimentos computacionais mostram que os algoritmos propostos são eficientes, competitivos e rápidos.

Abstract

CAMPOS, Alba Assis, M.Sc., Universidade Federal de Viçosa, January, 2018. **The Waste Collection Vehicle Routing Problem with Time Windows: Heuristic Approach.** Adviser: José Elias Claudio Arroyo.

In this work we address The Waste Collection Vehicle Routing Problem with Time Windows (WCVRPTW), whose objective is to find the routes for garbage collection vehicles so that all customers are fully served and the total distance traveled by the vehicles is the smallest possible, thus minimizing transportation costs. In order to achieve this objective, some constraints must be satisfied: customers must be served within a period of time; there are departing and return times of the vehicles to the warehouse; vehicles have capacity constraints; the routes have loading volume constraints and number of customers served; the vehicles should leave and return empty to the depot, and when full should go to the nearest landfill for garbage disposal. In addition, drivers of the vehicles must have a lunch break. WCVRPTW is a real problem that belongs to the NP-hard class. In this work, to solve it, four heuristic algorithms are developed: Simulated Annealing (SA), Tabu Search (TS), and two hybrid methods based on the Iterated Local Search (ILS) and Variable Neighborhood Descent (VND) metaheuristics, called ILS-VND and ILS-RVND. The performances of the proposed methods are analyzed in small and medium-sized instances generated in this work, and also in large instances available in the literature. Computational experiments show that the proposed methods are efficient, competitive and fast.

Capítulo 1

Introdução

O crescimento mundial da população e o processo de urbanização têm feito aumentar a produção nas grandes indústrias e o consumo de bens no mercado, apresentando um aumento na geração de lixo das cidades nos últimos anos. Enquanto as empresas ganham um crescimento em benefício deste comportamento consumista, o modelo de consumo deixa problemas ambientais, sociais e de saúde pública no mundo inteiro. O lixo coletado no Brasil em 2010 alcançou a taxa média de 306 kg/hab/ano, segundo o IBGE [2010]. Estas marcas surpreendentes colocam o Brasil entre os maiores produtores de lixo do mundo, com dispêndio de R$ 4 bilhões por ano. O custo da coleta somente com equipamentos e pessoal indica ser aproximadamente 50% deste total [IBAM, 2001].

O processo de coleta de lixo representa, geralmente, a maior preocupação dos órgãos de gerenciamento dos serviços de limpeza pública brasileiros, devido ao seu alto custo e o grau de dificuldade de realização. Segundo D'almeida et al. [2010], os serviços de limpeza absorvem entre 7 e 15% dos recursos de um orçamento municipal, dos quais 50 a 70% são destinados exclusivamente à coleta e ao transporte de lixo.

Desta forma, o propósito deste estudo é tratar do desenvolvimento de ferramentas computacionais que sejam capazes de otimizar os custos de coleta e transporte de lixo respeitando restrições operacionais, como restrições de tempo, capacidade dos veículos, demanda dos clientes, etc.

O Problema de Roteamento de Veículos, conhecido na literatura inglesa como *Vehicle Routing Problem* - VRP, foi inicialmente proposto por Dantzig & Ramser [1959], e tem como objetivo encontrar rotas para os veículos disponíveis de modo que todos os clientes sejam plenamente atendidos e a distância total percorrida seja a menor possível, minimizando assim os custos de transporte.

Visando atender a um mercado cada vez mais exigente e competitivo, diversas generalizações do VRP foram criadas. Cada generalização contempla uma determinada característica da situação encontrada no mundo real.

A generalização estudada neste trabalho é o Problema de Roteamento de Veículos para Coleta de Lixo com Janelas de Tempo, em inglês, *Waste Collection Vehicle Routing Problem with Time Windows* (WCVRPTW), proposto por Kim et al. [2006]. No WCVRPTW, além das restrições comuns existentes no caso clássico do VRP, os clientes devem ser atendidos dentro de um período de tempo, existem horários de saída e retorno dos veículos ao depósito, os veículos possuem restrições de capacidade e as rotas possuem restrições de volume de carregamento e número de clientes atendidos, ou seja, os motoristas dos veículos possuem restrições diárias em relação ao volume de lixo coletado e na quantidade de clientes atendidos. Os veículos devem sair e retornar vazios ao depósito, e quando cheios, devem ir para o aterro sanitário para descarga do lixo. Além disso, os motoristas dos veículos devem realizar uma parada de almoço.

A tarefa de resolver o WCVRPTW não é trivial, uma vez que problemas desta natureza podem ser classificados como NP-difícil. Estes problemas apresentam grande complexidade, no sentido de não existirem algoritmos eficientes e completos que forneçam a solução ótima em tempo adequado. Dessa forma, muitas vezes, sua resolução recorre a algoritmos heurísticos e metaheurísticos para encontrar uma resposta próxima a ótima em tempo aceitável. [Lenstra & Kan, 1981].

Este trabalho propõem três algoritmos heurísticos (*Iterated Local Search - Random Variable Neighborhood Descent* (ILS-RVND), *Iterated Local Search - Variable Neighborhood Descent* (ILS-VND) e *Simulated Anneling* (SA)) para a resolução do WCVRPTW. E apresenta a reimplementação do algoritmo heurístico *Tabu Search* (TS), proposto por Benjamin & Beasley [2010]. Todas as abordagens foram implementadas e testadas em instâncias da literatura e em novas instâncias geradas.

1.1 Objetivos

O objetivo geral deste trabalho é desenvolver heurísticas baseadas em metaheurísticas, que visem encontrar soluções de alta qualidade em tempo computacional aceitável, a fim de resolver o problema de roteamento de veículos para coleta de lixo com janelas de tempo.

1.1.1 Objetivos específicos

Para alcançar o objetivo geral, é necessário que sejam atingidos os objetivos específicos, listados a seguir:

(a) Realizar revisões bibliográficas para obter atualizações dos trabalhos que envolvem problemas de roteamento de veículos para coleta de lixo com janelas de tempo.

(b) Propor novas abordagens heurísticas para resolver este problema.

(c) Implementar cada uma das abordagens propostas.

(d) Realizar experimentos computacionais para calibração dos parâmetros utilizados nas estratégias implementadas.

(e) Realizar uma análise comparativa entre os resultados obtidos a partir das heurísticas propostas e os resultados das abordagens heurísticas da literatura.

1.2 Organização da dissertação

O presente trabalho está organizado da seguinte maneira:

Capítulo 2: Apresenta a definição formal do problema e os trabalhos anteriores contidos na literatura.

Capítulo 3: Apresenta as principais características dos algoritmos heurísticos implementados para a resolução deste trabalho.

Capítulo 4: Descreve os algoritmos propostos para resolução do WCVRPTW.

Capítulo 5: Apresenta os experimentos realizados neste trabalho.

Capítulo 6: Apresenta as considerações finais e as sugestões de futuros trabalhos.

Capítulo 2

Problema de Roteamento de Veículos para Coleta de Lixo com Janelas de Tempo

Neste capítulo são mostrados os conceitos e definições acerca do Problema de Roteamento de Veículos para Coleta de Lixo com Janelas de Tempo. Uma descrição do problema de roteamento de veículos e algumas de suas generalizações para contextualização do problema a ser tratado nessa dissertação são apresentados na Seção 2.1. Em seguida, na Seção 2.2, são apresentadas as características do problema abordado. Na Seção 2.2.1, um exemplo prático é descrito com o propósito de facilitar a compreensão dos diversos aspectos apresentados. Posteriormente, em 2.2.2, são apresentados os principais trabalhos da literatura que abordam o problema. Na Seção 2.2.3 é apresentada a definição formal do WCVRPTW. Por fim, sua formulação matemática é apresentada na Seção 2.2.4.

2.1 O problema de roteamento de veículos

O Problema de Roteamento de Veículos, conhecido na literatura como *Vehicle Routing Problem* (VRP), é um problema clássico da área de Pesquisa Operacional, sendo profundamente estudados pela comunidade científica da área. Este problema foi apresentado pela primeira vez por Dantzig & Ramser [1959], que estudaram a aplicação real na distribuição de gasolina para postos de venda de combustível. Trata-se de um problema de natureza combinatória, da classe NP-difícil [Lenstra & Kan, 1981].

O VRP é definido em um grafo não orientado, $G = (V, E)$, onde $V = \{0,1,...,N\}$ é o conjunto de vértices e $E = \{(i, j): i, j \in V\}$ o conjunto de arestas. O vértice 0 representa o depósito de onde partem m veículos e os demais vértices são locais ou clientes. Um custo não negativo, distância ou tempo de viagem é dado por c_{ij} definido para cada aresta de E. Cada vértice i tem uma demanda q_i e um tempo de serviço s_i. O VRP tem como objetivo definir as rotas dos veículos entre o depósito e os clientes que minimize a distância percorrida ou o tempo. As restrições básicas do problema consistem em: cada local é visitado uma única vez por um único veículo; cada rota é iniciada no depósito e finalizada no mesmo depósito; todas as demandas de todos os clientes devem ser satisfeitas, respeitando a capacidade Q do veículo [Caric & Gold, 2008].

Visando atender a um mercado cada vez mais exigente e competitivo, diversas generalizações do VRP foram criadas. Cada generalização contempla um conjunto de restrições que caracterizam diferentes situações encontradas no mundo real. Dentre as diversas generalizações existentes, podemos citar algumas:

- Problema de roteamento de veículos com coleta e entrega (*Vehicle Routing Problem with Pickup and Delivery* (VRPPD)): é uma generalização do VRP, no qual cada cliente possui uma oferta a ser coletada e uma demanda a ser atendida. Durante toda a rota, a capacidade máxima dos veículos não pode ser ultrapassada [Dumas et al., 1991; Savelsbergh & Sol, 1995; Berbeglia et al., 2007, 2010].

- Problema de roteamento de veículos com janelas de tempo (*Vehicle Routing Problem with Time Windows* (VRPTW)): é uma generalização do VRP adicionando a restrição de janela de tempo para os clientes. Neste problema, os veículos devem atender todos os clientes dentro do intervalo de tempo estipulado, sendo que os mesmos podem chegar antes do limite inferior da janela de tempo e esperar para começar o atendimento. Por outro lado, veículos atrasados não são permitidos, embora outras generalizações do problema permitam atrasos dentro de uma abordagem envolvendo penalidades [Solomon, 1987; Desrochers et al., 1990].

- Problema de roteamento de veículos com depósitos intermediários (*Vehicle Routing Problem with Intermediate Facilities* (VRPIF)): é uma generalização do VRP, onde os veículos podem realizar paradas em depósitos intermediários para descarregar as demandas coletadas [Kim et al., 2006].

2.2 Problema de roteamento de veículos para coleta de lixo com janelas de tempo

O Problema de Roteamento de Veículos para Coleta de Lixo com Janelas de Tempo, proposto por Kim et al. [2006], consiste em determinar rotas para os veículos coletores de lixo comercial, de forma a minimizar o custo de transporte, garantindo que a demanda dos clientes sejam totalmente atendidas sem que haja violação das restrições.

Problemas de coleta de lixo são frequentemente considerados como problemas de roteamento em arco, sem janelas de tempo. Entretanto, esse ponto de vista pode ser aplicado apenas para problemas de coleta de lixo residencial, uma vez que, na coleta de lixo comercial, as paradas podem conter janelas de tempo. Segundo Kim et al. [2006], o problema de roteamento de veículos para coleta de lixo comercial pode ser caracterizado como uma generalização dos problemas de roteamento de veículos com janela de tempo e depósitos intermediários (VRPTW-IF).

Neste trabalho, considera-se a existência de apenas um depósito, um conjunto de veículos homogêneos, um conjunto de pontos de coleta e um conjunto de aterros sanitários. Cada ponto de coleta tem uma janela de tempo, ou seja, um intervalo de tempo dentro do qual o serviço deve ser inicializado, e uma quantidade de demanda a ser coletada. Os veículos saem do depósito, coletam o lixo nos pontos de coleta, até que seu limite de capacidade seja atingido, descarregam o lixo em um aterro sanitário, repetem a coleta e descarga tantas vezes forem necessárias e possíveis, e finalmente retornam ao depósito, dentro de um limite de tempo pré-estabelecido. Os motoristas precisam realizar a parada de almoço, dentro do intervalo de 11 às 13 horas, por exemplo.

Dois problemas de restrição de capacidade são considerados ao se criar a rota: a capacidade do veículo e a capacidade da rota. A capacidade do veículo considera o volume máximo que cada um pode conter. E a capacidade da rota é a capacidade diária para cada motorista: número máximo de clientes atendidos e quantidade máxima de lixo coletado. A capacidade do veículo determina quando o descarregamento da carga deve ser realizado. Se a capacidade do veículo for maior que o volume da capacidade da rota, haverá apenas uma viagem ao aterro sanitário, e ela será a última antes do retorno ao depósito. Se, porém, o volume coletado na rota for maior que a capacidade do veículo, múltiplas viagens ao aterro sanitário serão necessárias em uma mesma rota. Cada veículo deve iniciar e terminar sua rota no depósito com volume zero.

2.2.1 Exemplo prático

Com o propósito de descrever de forma detalhada o problema estudado, a Figura 2.1 mostra o exemplo de uma rota de um veículo coletor de lixo, onde são atendidos 25 clientes e são utilizados 3 aterros sanitários. Note que o veículo parte do depósito e visita todos os clientes, com o intuito de recolher suas demandas. Quando o veículo está cheio, ele descarrega o lixo coletado no aterro sanitário mais próximo e continua visitando os clientes. A última visita do veículo deve ser no aterro sanitário, devido o problema impor a restrição que o veículo deve retornar vazio ao depósito.

A Figura 2.2 ilustra um exemplo que considera janelas de tempo e parada de almoço do motorista do veículo.

Note que na Figura 2.2 (a), todos os nós i, exceto o depósito, possuem janelas de tempo. Esses nós são visitados dentro do intervalo de tempo $[a_i,b_i]$, a_i representa início da janela de tempo e b_i representa final da janela de tempo. Por exemplo, no nó 1, a_i = 10:00 e b_i = 12:00, e o veículo realizou a visita às 10:30.

A Figura 2.2 (b) apresenta o exemplo da parada de almoço. Se a parada de almoço L, ponto virtual, acontecer entre os nós 2 e 3, a rota é inviável, pois o tempo de serviço (s_i) do nó 5 é estendido até 15:10.

A Figura 2.2 (c) apresenta uma rota viável do problema em questão, pois considera o atendimento dos clientes dentro do período de tempo e a parada de almoço do motorista do veículo. Note que o nó L possui janela de tempo [11:00-12:00] e foi atendido dentro dela, pois a parada aconteceu exatamente às 12:00. E a parada de almoço teve duração de 1 hora, pois o próximo cliente (nó 3) foi atendido às 13:00.

2.2.2 Revisão da literatura

Inicialmente o WCVRPTW foi abordado por Kim et al. [2006], que realizaram um estudo sobre o problema de roteamento de veículos para coleta de lixo comercial. Neste trabalho, os autores consideram como restrições, janelas de tempo, várias viagens aos aterros sanitários e pausa de almoço para os motoristas dos veículos. O problema tem como objetivos minimização do tempo de viagem, redução do número de rotas e balanceamento da carga de trabalho. Para atingir os objetivos, os autores desenvolveram um algoritmo construtivo baseado no algoritmo *Inserção de Solomon* [Solomon, 1987] e para refinar a solução utilizaram um algoritmo capacitado baseado em clusterização (CBA) [Kim et al., 2006]. Neste trabalho, os autores propuseram as instâncias baseadas em problemas do mundo real para testar os algoritmos.

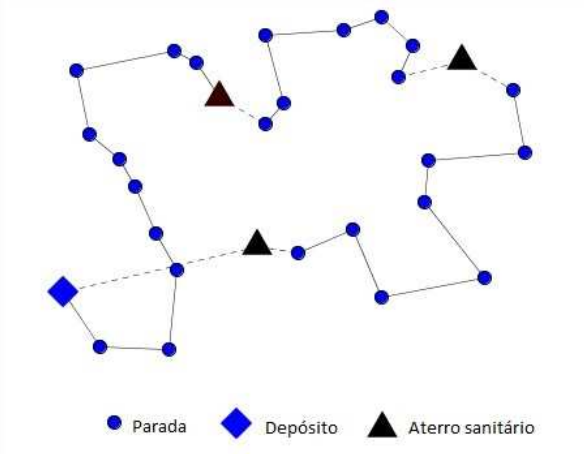

Figura 2.1: Exemplo de rota de um veículo com múltiplos aterros sanitários - FONTE (Kim et al. [2006]).

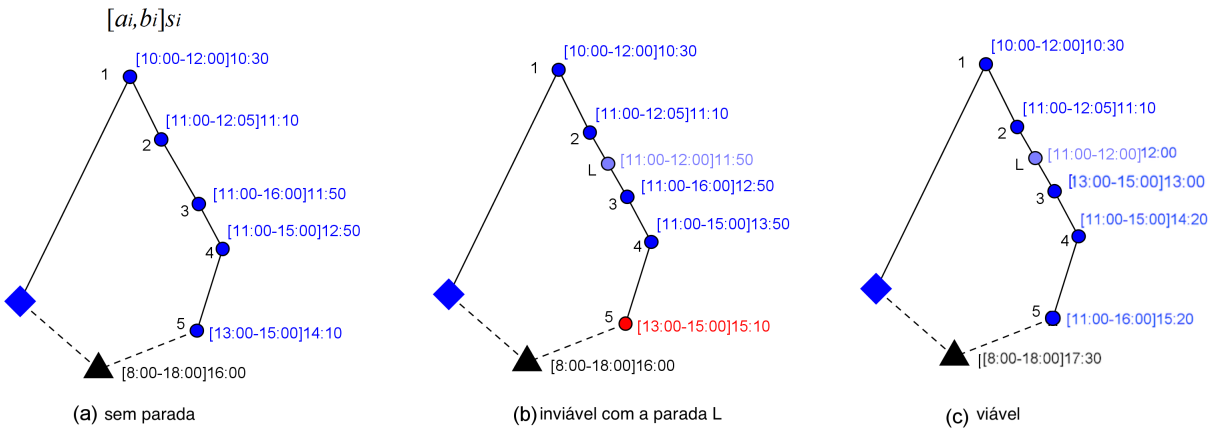

Figura 2.2: Rota de um veículo com pausa de almoço do motorista - FONTE (Kim et al. [2006]).

Logo depois, em 2007, Ombuki-Berman et al. [2007] estudaram o mesmo problema e propuseram um algoritmo genético multiobjetivo. Neste trabalho, os autores utilizaram o conjunto de instâncias proposto por Kim et al. [2006].

Benjamin & Beasley [2010] propuseram um algoritmo construtivo guloso e apresentaram três metaheuríticas: *Tabu Search* (TS), *Variable Neighborhood Search* (VNS), e *Variable Neighbourhood Tabu Search* (VNTS), para a resolução do problema. Os autores utilizaram o mesmo conjunto de instâncias proposto por Kim et al. [2006]. Benjamin & Beasley [2013] melhoraram a efetividade dos algoritmos TS, VNS e VNTS [Benjamin & Beasley, 2010].

Islam & Rahman [2012] propuseram um algoritmo *Ant colony optimization*

(ACO) para resolver o problema WCVRPTW. O conjunto de instâncias utilizado foi proposto por Kim et al. [2006] e os resultados obtidos pelo algoritmo ACO foram melhores do que os algoritmos propostos por Benjamin & Beasley [2010].

Buhrkal et al. [2012] apresentaram um modelo de programação linear inteira (MILP) do problema WCVRPTW, e propuseram um algoritmo *Adaptive Large Neighborhood Search* (ALNS) para resolver o problema. O algoritmo foi testado no conjunto instâncias proposto por Kim et al. [2006] e em um conjunto de instâncias fornecidas por uma empresa de coleta de lixo dinamarquesa. Os resultados obtidos pelo algoritmo ALNS foram melhores do que os propostos por Benjamin & Beasley [2010] e Islam & Rahman [2012].

Han & Ponce Cueto [2015] apresentaram as principais contribuições sobre o Problema de Roteamento de Veículos para Coleta de Lixo da literatura e analisaram os diferentes métodos e técnicas para resolução do problema.

Recentemente, Idrus et al. [2017] apresentaram uma revisão bibliográfica das principais contribuições do Problema de Roteamento de Veículos para Coleta de Lixo nos últimos 11 anos. Os objetivos foram identificar a função objetivo do problema, as restrições, as heurísticas e o conjunto de instâncias utilizado em cada um dos problemas apresentados nos estudos de caso.

2.2.3 Definição formal do problema

O problema estudado neste trabalho pode ser definido da seguinte maneira: seja $V = V^d \cup V^f \cup V^c$ o conjunto de vértice, onde $V^d = \{0\}$ corresponde ao depósito, $V^f = \{1,...,m\}$ corresponde os aterros sanitários e $V^c = \{m + 1,...,m + n\}$ corresponde aos clientes, $G = (V,A)$ um grafo completo e $A = \{(i,j)\}|\ i,j \in V,\ i \neq j\}$ o conjunto de arestas. Cada cliente i possui uma janela de tempo $[a_i, b_i]$ e uma demanda q_i. Considere $K = \{1,...,k\}$ como o conjunto de veículos homogêneos disponíveis no depósito e t_{ij} e c_{ij} são o tempo de viagem e o custo associado a cada arco $(i,\ j)$, respectivamente. O problema consiste em determinar as rotas dos veículos de tal forma que as demandas dos clientes sejam atendidas e os custos operacionais de transportes sejam minimizados. As seguintes restrições devem ser satisfeitas:

- Toda rota começa e termina no depósito;
- Todo veículo deve sair e retornar vazio ao depósito;
- A capacidade do veículo deve ser respeitada;
- Cada cliente é atendido somente uma vez por somente um veículo;

- Todos os clientes devem ser atendidos dentro da sua respectiva janela de tempo;
- Cada motorista do veículo deve realizar uma única pausa para almoço dentro da janela de tempo;
- Cada rota possui um limite diário de paradas (atendimento aos clientes);
- Cada rota possui um limite diário de carregamento (volume de lixo coletado).

2.2.4 Formulação matemática

A formulação matemática apresentada a seguir é proposta por Buhrkal et al. [2012] e utiliza os seguintes índices, parâmetros e variáveis de decisão.

Índices

- $V = V^d \cup V^f \cup V^c$
- V^d conjunto de depósito representado por $V^d = \{0\}$;
- V^f conjunto de aterro sanitário;
- V^c conjunto de cliente;
- K conjunto de veículo representado por $K = \{1, ..., k\}$;

Parâmetros

- $c_{i,j}$ distância entre os clientes i e j;
- $t_{i,j}$ tempo de deslocamento entre os clientes i e j;
- C capacidade do veículo;
- s_i tempo de serviço do cliente i;
- q_i demanda do cliente i;
- $[a_i, b_i]$ janela de tempo para atender o cliente i;
- S número máximo de clientes que podem ser atendidos;
- R quantidade máxima de lixo que pode ser coletado;
- $[a^u, b^u]$ janela de tempo da parada de almoço;

- M constante de valor suficientemente grande.

Variáveis de decisão

- $w_{i,k}$ tempo que o veículo k iniciou o serviço no cliente i;
- $d_{i,k}$ demanda acumulada do cliente i no veículo k;
- s^u duração da parada de almoço;
- x_{ijk} variável binária que assume 1 se a aresta (i,j) é usada pelo veículo k e 0 caso contrário;
- y_{ijk} variável binária que assume 1 se ocorreu parada de almoço do motorista do veículo k nas arestas (i,j) e 0 caso contrário.

Modelo de PLI

$$Minimize \sum_{(i,j)\in A} \sum_{k\in K} c_{ij} x_{ijk} \tag{2.1}$$

Sujeito a

$$\sum_{j\in V} x_{0jk} = 1, \forall k \in K \tag{2.2}$$

$$\sum_{i\in V} x_{i0k} = 1, \forall k \in K \tag{2.3}$$

$$\sum_{i\in V} \sum_{k\in K} x_{ijk} = 1, \forall j \in V_c \tag{2.4}$$

$$\sum_{i\in V} x_{ijk} = \sum_{i\in V} x_{jik}, \forall j \in V_c \cup V_f, k \in K \tag{2.5}$$

$$a_i \leq w_{ik} \leq b_i, \forall i \in V, k \in K \tag{2.6}$$

$$w_{ik} + s_i + y_{ijk} s^u + t_{ij} \leq w_{jk} + (1 - x_{ijk})M, \forall (i,j) \in A, k \in K \tag{2.7}$$

$$\sum_{i\in(0)} d_{ik} = 0, \forall k \in K \tag{2.8}$$

$$d_{ik} + q_i \leq d_{jk} + (1 - x_{ijk})M, \forall i \in V \setminus V_f, j \in V, k \in K \tag{2.9}$$

$$d_{ik} \leq C, \forall i \in V, k \in K \tag{2.10}$$

$$\sum_{(i,j)\in A} y_{ijk} = 1, \forall k \in K \tag{2.11}$$

$$y_{ijk} \leq x_{ijk}, \forall (i,j) \in A, k \in K \tag{2.12}$$

$$a^u + s^u + t_{ij} \leq w_{jk} + (1 - y_{ijk})M, \forall (i,j) \in A, k \in K \tag{2.13}$$

$$w_{ik} + s_i + t_{i,j} \leq b^u + (1 - y_{ijk})M, \forall (i,j) \in A, k \in K \tag{2.14}$$

$$\sum_{(i) \in V} \sum_{j \in V^c} x_{ijk} \leq S, \forall k \in K \tag{2.15}$$

$$\sum_{i \in V^c} d_{ik} \leq R, \forall k \in K \tag{2.16}$$

$$d_{ik} \geq 0, \forall i \in V, k \in K \tag{2.17}$$

$$x_{ijk} \in \{0,1\}, \forall (i,j) \in A, k \in K \tag{2.18}$$

$$y_{ijk} \in \{0,1\}, \forall (i,j) \in A, k \in K \tag{2.19}$$

A função objetivo (2.1) busca minimizar o custo total da viagem, sujeito as seguintes restrições:

- (2.2): O veículo inicia a rota a partir do depósito.
- (2.3): O veículo retorna para o depósito ao final da rota.
- (2.4): Cada cliente deve ser visitado por um único veículo.
- (2.5): Conservação de fluxo.
- (2.6) e (2.7): Garantem a viabilidade das rotas com relação às restrições de janela de tempo.
- (2.8): O veículo deve partir e retornar vazio ao depósito.
- (2.9): Acumula a demanda coletada de todos os clientes.
- (2.10): A capacidade do veículo não deve ser ultrapassada.
- (2.11) O motorista do veículo deve fazer uma única parada de almoço.
- (2.12): A parada de almoço só pode acontecer entre os vértices i e j se eles estiverem conectados.
- (2.13) e (2.14): A parada de almoço deve acontecer dentro do intervalo de tempo.
- (2.15): Número máximo de clientes que podem ser atendidos.
- (2.16): Quantidade máxima de lixo que pode ser coletada.

- (2.17): Restrição de não-negatividade para demanda acumulada.
- (2.18) e (2.19): Restrições de integralidade.

Capítulo 3

Uma Revisão de Métodos Heurísticos e Metaheurísticos

Tendo em vista que o presente trabalho consiste em um problema de natureza combinatória, cujo esforço computacional é não polinomial, ou seja, cresce exponencialmente com o tamanho do problema, torna-se necessária uma abordagem heurística para a resolução deste problema, uma vez que os algoritmos exatos, apesar de todos os avanços observados, são incapazes de obter soluções ótimas em um tempo de processamento viável [Cunha, 2006].

Heurística é uma técnica baseada em processos intuitivos que conduz à resolução de problemas de elevado nível de complexidade. Essa técnica procura uma solução de boa qualidade a um custo computacional razoável, sem garantir sua otimalidade [Talbi, 2009].

As heurísticas são divididas em duas classes:

- A heurística construtiva constrói uma solução inicial, elemento a elemento. Para selecionar o elemento a ser inserido, é necessário ter um conhecimento do problema para a sua elaboração.

- A heurística de refinamento ou busca local caminha iterativamente sobre a vizinhança de uma solução inicial, na busca por melhores resultados.

As metaheurísticas podem ser definidas como heurísticas estruturadas, ou seja, possuem alguma estratégia de manipulação das heurísticas de busca local, visando melhorar a exploração das vizinhanças, escapando de ótimos locais [Talbi, 2009].

A seguir, são apresentadas as metaheurísticas que influenciaram o desenvolvimento deste trabalho.

Algoritmo 1: ILS

```
1  begin
2      s_0 ← GerarSoluçãoInicial();
3      s* ←BuscaLocal(s_0);
4      while critério de parada não satisfeito do
5          s' ← Perturbar(s*);
6          s'' ← BuscaLocal (s');
7          AvaliaCriterioAceitacao (s*, s'')
8      end
9      return s*;
10 end
```

3.1 *Iterated Local Search* - ILS

A metaheurística *Iterated Local Search* (ILS) foi proposta por Lourenço et al. [2003] e consiste de uma técnica que explora o espaço de soluções através de um algoritmo de busca local e por um algoritmo que aplica perturbações aos ótimos locais encontrados. Basicamente a técnica consiste em aplicar uma busca local a partir de uma solução inicial e, após encontrar uma solução de ótimo local, é aplicada uma perturbação nessa solução para poder explorar outras regiões do espaço de solução. Esse processo se repete até que algum critério de parada seja satisfeito (número de iterações, tempo e outros).

Para desenvolver um algoritmo baseado em ILS, quatro procedimentos devem ser definidos: (i) Procedimento *GerarSoluçãoInicial*(), gera uma solução inicial para o problema; (ii) Procedimento *BuscaLocal*(), responsável por explorar o espaço de soluções afim de encontrar ótimos locais; (iii) Procedimento *Perturbar*(), que modifica a solução corrente de maneira que um novo ponto de partida do espaço de soluções seja explorado e (iv) Procedimento *CritérioAceitação*(), responsável por determinar qual solução será utilizada durante a próxima etapa da perturbação.

O Algoritmo 1 apresenta o pseudocódigo do ILS. Percebe-se que na linha 2 gera-se uma solução inicial s_0. Essa solução é refinada por um método de busca local, gerando um ótimo local s^* (linha 3). A solução s^* sofre uma perturbação, dando origem a uma outra solução s' (linha 5). Esta solução s' passa por um processo de refinamento, resultando em um novo ótimo local s'' (linha 6). Em seguida é verificado de qual solução a busca prosseguirá, se do ótimo local corrente s^* ou do novo s''. O procedimento continua até que o critério de parada seja satisfeito

Algoritmo 2: VND

```
1  begin
2      Seja s_0 uma solução inicial e r o número de estruturas de vizinhança
3      s ← s_0
4      k ← 1
5      while k ≤ r do
6          Encontre o melhor vizinho s' ∈ N^(k)(s);
7          if f(s') < f(s) then
8              s ← s'
9              k ← 1;
10         end
11         else
12             k ← k + 1;
13         end
14     end
15     return s;
16 end
```

3.2 *Variable Neighborhood Descent* - VND

Proposta por Mladenović & Hansen [1997], a metaheurística *Variable Neighborhood Descent* (VND) consiste em um algoritmo de busca local que explora o espaço de soluções através de trocas sistemáticas da função de vizinhança. Baseia-se nas seguintes observações: um ótimo local de uma vizinhança não é necessariamente ótimo local de outra vizinhança e um ótimo global é um ótimo local de todas as estruturas de vizinhanças [Talbi, 2009].

O funcionamento do VND é detalhado no Algoritmo 2. Seja s_0 a solução inicial do problema e r o número de estruturas de vizinhança (linha 2). A variável s recebe a solução inicial do problema (linha 3) e a variável k é inicializada com a primeira estrutura de vizinhança (linha 4). A cada iteração é realizada uma busca local em uma estrutura de vizinhança k, gerando uma nova solução s' (linha 6). Se a função objetivo de s' for menor que a de s, então a solução s é atualizada e a busca local é reiniciada com a primeira estrutura de vizinhança (linhas 7 - 10). Caso contrário, a busca local continua com a próxima estrutura de vizinhança, $k + 1$ (linhas 11 - 13). O laço de repetição termina quando k for maior que r, ou seja, não existirem mais estruturas de vizinhanças a serem exploradas.

3.3 *Random Variable Neighborhood Descent* - RVND

Alguns trabalhos da literatura, como Subramanian et al. [2010], Souza et al. [2010], propuseram a escolha aleatória das estruturas de vizinhança durante o processamento da busca, criando uma variação do VND denominada *Random Variable Neighborhood Descent*. Ao contrário do VND, que utiliza uma ordem pré-definida de vizinhanças para explorar o espaço de soluções, o RVND utiliza uma ordem aleatória a cada chamada.

O pseudocódigo do procedimento RVND é descrito no Algoritmo 3. Seja $\Gamma = \{N^1,...,N^n\}$ uma lista composta por n estruturas de vizinhanças (linha 2). A cada execução do laço de repetição (linhas 3-13), uma vizinhança $nc \in \Gamma$ é aleatoriamente escolhida (linha 4) e o melhor vizinho é escolhido (linha 5). Caso a solução atual seja melhorada, Γ é reinicializada com todas as vizinhanças (linha 8), caso contrário, nc é removida de Γ (linha 11). O laço de repetição termina quando a lista Γ torna-se vazia.

Algoritmo 3: RVND

```
1  begin
2      Γ ← Inicializar (Γ)
3      while Γ ≠ ∅ do
4          nc ← Aleatorio(N^1, N^n)
5          s' ← MelhorVizinho (s, nc)
6          if f(s') < f(s) then
7              s ← s';
8              Γ ← Inicializar(Γ)
9          end
10         else
11             Γ ← Γ − {nc};
12         end
13     end
14     return s;
15 end
```

3.4 *Simulated Annealing* - SA

A metaheurística *Simulated Annealing* (SA) foi proposta por Kirkpatrick et al. [1987]. Trata-se de um procedimento de busca local probabilístico. Seu funciona-

mento é associado a uma operação chamada recozimento, relacionada à termodinâmica. Esta operação consiste em aquecer o sistema até atingir uma alta temperatura para então diminuí-la em etapas lentas até que o estado de equilíbrio seja alcançado.

O pseudocódigo do procedimento SA é descrito no Algoritmo 4. Este algoritmo começa sua busca a partir de uma solução inicial (linha 2). O procedimento consiste em um laço de repetição que gera aleatoriamente, em cada iteração, um único vizinho s' da solução corrente s (linha 7). O critério para aceitar uma solução como a solução atual é feito em função de uma probabilidade determinada de acordo com a qualidade da solução e com a temperatura atual (linhas 8 - 9). Caso o critério de aceitação seja válido, o valor da melhor solução atual (s) recebe o valor da nova solução (s') (linha 10). Em seguida, é verificado se o valor da nova solução (s') é superior ao valor da melhor solução encontrada até o momento (s^*) (linha 11), caso verdadeiro, então esta possui seu valor atualizado (linha 12). Posteriormente, o contador de iteração é incrementado (linha 15) e a temperatura T é modificada pela taxa de resfriamento (linha 17). O algoritmo é encerrado quando a temperatura atual atingir ou estiver abaixo da temperatura mínima.

Algoritmo 4: Simulated Annealing

```
1  begin
2      s* ← s
3      T ← T_i
4      while T > T_f do
5          IterT ← 0
6          while IterT < SAmax do
7              s' ← gerar aleatoriamente uma solução vizinha s' ∈ N(s)
8              Δ = f(s') − f(s)
9              if (Δ < 0) or random [0,1] < e^{−Δ/T} then
10                 s ← s'
11                 if (f(s') < f(s*)) then
12                     s* ← s'
13                 end
14             end
15             IterT ← IterT + 1
16         end
17         T ← T x (1 − α)
18     end
19     return s*
20 end
```

3.5 *Tabu Search* - TS

Proposta por Glover [1986], a metaheurística Tabu Search (Busca Tabu) consiste em incluir uma estrutura de memória que permita o algoritmo de busca local escapar de soluções ótimas locais.

Segundo Glover [1986], a Busca Tabu inclui elementos de memória de curto prazo, que evitam a reversão de movimentos recentes, e uma memória de mais longo prazo, para reforçar a atração por soluções mais distantes. O algoritmo realiza buscas locais sempre visando encontrar um ótimo local, permitindo movimentos que não sejam de melhoria da solução, para evitar retornar em soluções previamente avaliadas.

Os movimentos realizados são armazenados em uma lista tabu. Estes ficam armazenados na lista segundo um critério de tempo (por exemplo número de iterações), e enquanto o movimento estiver registrado na lista tabu, este movimento é considerado tabu, não podendo ser realizado, e impedindo a ciclagem da busca local.

A função aspiração é utilizada para considerar uma solução s que melhore a melhor solução encontrada até o momento, mesmo se o movimento realizado seja um movimento proibido (movimento tabu). O pseudocódigo do procedimento Busca Tabu é apresentado no Algoritmo 5. Percebe-se que na linha 2 gera-se uma solução inicial s_0. Em seguida, a cada iteração o algoritmo explora um subconjunto V da vizinhança $N(s)$ da solução corrente s. A solução s' de V com função objetivo de menor valor torna-se a nova solução corrente mesmo que s' seja pior que s (s' não pode estar armazenado na lista tabu). Esses movimentos são armazenados na lista tabu (linha 10) e permanecem proibidos por número de iterações. Em seguida, é verificado se o valor da nova solução (s) é superior ao valor da melhor solução encontrada até o momento (s^*) (linha 12), caso verdadeiro, então esta possui seu valor atualizado (linha 13). Posteriormente, o contador de iteração é incrementado (linha 14). O procedimento continua até que o critério de parada seja satisfeito.

Algoritmo 5: Busca Tabu

```
1  begin
2      Seja s_0 solução inicial
3      s* ← s_0
4      Iter ← 0
5      MelhorIter ← 0
6      T ← ∅
7      while (critério de parada não satisfeito) do
8          Seja s' ← s ⊕ m o melhor elemento de V ⊆ N (s) tal que o
           movimento m não seja tabu (m ∉ T) ou s' atenda a condição de
           aspiração (f(s') < A(f(s)))
9          Atualize a Lista Tabu T
10         s ← s'
11         if f(s) < f(s*) then
12             s* ← s;
13         end
14         Iter ← Iter + 1
15     end
16     return s*;
17 end
```

Capítulo 4

Metodologia para Resolução do WCVRPTW

Neste Capítulo, estão descritos os algoritmos heurísticos usados para a resolução do WCVRPTW. O primeiro algoritmo é baseado nas metaheurísticas *Iterative Local Search* (ILS) [Lourenço et al., 2003] e *Variable Neighborhood Descent* (VND) [Mladenović & Hansen, 1997], denotado como ILS-VND. O segundo algoritmo é similar ao ILS-VND. A diferença entre eles está no procedimento de busca local, que utiliza uma ordenação aleatória das vizinhanças (RVND), nomeado ILS-RVND. O terceiro algoritmo é baseado na metaheurística *Simulated Annealing* [Kirkpatrick et al., 1987] e o último algoritmo é a reimplementação da metaheurística *Tabu Search* proposta por Benjamin & Beasley [2010].

A Seção 4.1 descreve a representação computacional da solução do problema. A Seção 4.2 apresenta o algoritmo construtivo da solução inicial para o WCVRPTW. As estruturas de vizinhanças são vistas na Seção 4.3. Por fim, as Seções 4.4, 4.5, 4.6, 4.7 fazem o detalhamento dos algoritmos desenvolvidos.

4.1 Representação da solução

A representação da solução do WCVRPTW é dada por uma estrutura de dados que armazena a soma das distâncias totais (DT) percorridas pelos veículos, o número total de veículos (NV) utilizados e o conjunto de rotas. Cada rota também é definida como uma lista contendo o depósito, os aterros sanitários e os clientes, dispostos de acordo com a ordem de visita, e a distância total percorrida na rota.

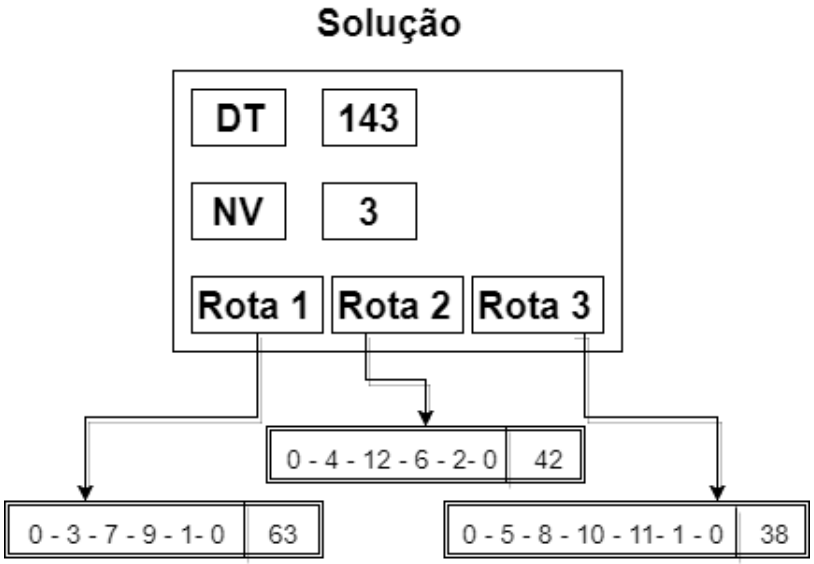

Figura 4.1: Representação de uma solução do WCVRPTW

A Figura 4.1 ilustra a representação de uma solução com três rotas para uma instância fictícia do WCVRPTW com dez clientes, dois aterros sanitários e um depósito. Note que o depósito é representado pelo número 0, os aterros sanitários são representados pelos números 1 e 2 e os demais números representam os clientes.

4.2 Heurística construtiva

A heurística construtiva apresentada neste trabalho é baseada em um algoritmo guloso, proposto por Benjamin & Beasley [2010]. Esta heurística é responsável por criar a solução inicial dos algoritmos propostos e é composta por cinco etapas, que são descritas nos pseudocódigos a seguir.

Na primeira etapa da heurística construtiva uma nova rota é criada. O pseudocódigo da etapa 1 é descrito no Algoritmo 6. A princípio, a variável B é inicializada com a lista de clientes não roteados (linha 2). Em seguida, é verificado se existe cliente a ser roteado (linha 3), se verdadeiro, as variáveis para controle do veículo são inicializadas (linhas 4 - 9) e a etapa 2 é executada. Caso contrário, o algoritmo é finalizado, pois todos os clientes foram roteados. As variáveis para controle do veículo são descritas abaixo:

- T: representa o tempo atual do veículo;
- $Stotal$: representa o número de clientes roteados até o momento;

- *Qtotal*: representa o volume total de lixo coletado pelo veículo;
- *Qatual*: representa a carga atual do veículo;
- r: representa o nó do último cliente visitado;
- *rest*: representa se houve parada de almoço.

Algoritmo 6: Etapa 1

```
1  begin
2      B ← V^c // B representa conjunto de clientes a serem roteados
3      if |B| ≠ 0 then
4          T ← a_0 // T é o tempo atual e a_0 janela de tempo inicial do
           depósito
5          Stotal ← 0 //número de clientes roteados
6          Qtotal ← 0 // volume total de demanda coletada
7          Qatual ← 0 // carga atual do veículo
8          r ← 0 // último nó visitado
9          rest ← 0 // parada de almoço do motorista
10         go to etapa 2
11     end
12     else
13         return Rotas
14     end
15 end
```

Na segunda etapa da heurística construtiva é realizada uma tentativa de parada de almoço do motorista do veículo. O pseudocódigo da etapa 2 é descrito no Algoritmo 7. Se o motorista não tiver realizado a parada de almoço ($rest = 0$) e o tempo atual (T) estiver dentro do período de almoço $[a^u, b^u]$ (linha 2), o motorista do veículo realiza a parada de almoço, *rest* é atualizado com valor 1 e T é acrescido com a duração de tempo da parada (s^u) (linha 3 e 4). Em seguida a etapa 3 é inicializada.

Na terceira etapa da heurística construtiva é realizada a inserção de um novo cliente à rota. A inserção é realizada a cada iteração, se o cliente i apresentar o menor tempo e satisfazer as seguintes condições (linha 3):

1. $j \in B$: assegura que o cliente j pertença ao conjunto de clientes não roteados B;

Algoritmo 7: Etapa 2

```
1 begin
2     if rest = 0 and T ∈ [a^u, b^u] then
3         rest ← 1
4         T ← T + s^u
5     end
6     go to etapa 3
7 end
```

2. $T + t_{rj} \in [a_j, b_j]$: assegura que o tempo resultante T do veículo pertença ao intervalo de tempo do cliente j, após o acréscimo do tempo de deslocamento do cliente r, último visitado, ao cliente j;

3. $Qatual + q_j \leq C$: assegura que a capacidade do veículo C seja maior ou igual a carga atual ($Qatual$) do veículo após o acréscimo do lixo coletado no cliente j.

4. $Qtotal + q_j \leq R$: assegura que a quantidade máxima de lixo a ser coletada R seja maior ou igual ao volume total de lixo coletado após o acréscimo do lixo coletado no cliente j;

5. $Stotal + 1 \leq S$: assegura que o número máximo de clientes visitados S seja maior ou igual ao número de clientes vistados após a inserção do cliente j a rota;

6. θ + $t_{j,n(j,\theta)}$+ $s_{n(j,\theta)}$+ $t_{n(j,\theta),0} \leq b_0$: esta expressão assegura que o veículo coletor de lixo irá chegar antes da janela de tempo final do depósito após visitar o cliente j, realizar o serviço de coleta em j, deslocar até o aterro sanitário disponível mais próximo, realizar o serviço de descarga do lixo e deslocar até o depósito. A notação $\theta = T + t_{rj} + s_j$ representa o tempo resultante após o veículo visitar e realizar o serviço no cliente j. Já as notações $t_{j,n(j,\theta)}$, $s_{n(j,\theta)}$, $t_{n(j,\theta),0}$, determinam o tempo de descolamento do cliente j ao aterro sanitário disponível mais próximo, o tempo de serviço para descarga do lixo e o tempo de deslocamento do aterro sanitário visitado até o depósito, respectivamente;

7. $\theta \leq b_u$, se $rest = 0$: assegura a parada de almoço do motorista do veículo, pois o cliente j torna-se inviável, caso o valor θ seja superior à janela de tempo final da parada de almoço.

Então, este cliente i é adicionado à rota e as variáveis do veículo são atualizadas (linhas 4 - 8), em seguida o cliente j é removido do conjunto B e o Algoritmo 7 é

executado novamente (linha 10). Caso contrário, a etapa 3 é finalizada e a etapa 4 é executada. O pseudocódigo da etapa 3 é descrito no Algoritmo 8.

Algoritmo 8: Etapa 3

```
1  begin
2      i = arg min {t_rj | j ∈ B, T+ t_rj ∈ [a_j,b_j], Qatual + q_j ≤ C, Qtotal +
       q_j ≤ R, Stotal + 1 ≤ S, θ + t_j,n(j,θ)+ s_n(j,θ)+ t_n(j,θ),0 ≤ b_0, θ ≤ b_u
       se rest = 0, onde θ = T+ t_rj+ s_j}
3      if i satisfaz as condições then
4          T ← t_ri+s_i // t_ri representa tempo de viagem do último cliente
           visitado ao cliente i
5          r ← i
6          Qtotal ← Qtotal + q_i
7          Qcurrent ← Qcurrent + q_i
8          Stotal ← Stotal + 1
9          B ← B - {i}
10         go to etapa 2
11     end
12     else
13         go to etapa 4
14     end
15 end
```

Na quarta etapa da heurística construtiva é realizada a descarga do lixo no aterro sanitário mais próximo. O pseudocódigo da etapa 4 é descrito no Algoritmo 9. Primeiramente, verifica-se o valor da carga atual do veículo, caso este valor seja maior que 0 (linha 2), então o tempo atual do veículo (T) é acrescido ao tempo de deslocamento do último cliente visitado r até o aterro sanitário disponível mais próximo e ao tempo de serviço gasto para descarregar o lixo (linha 3). Em seguida, a variável r recebe o nó do aterro sanitário visitado (linha 4). A carga atual *Qatual* é atualizada para o valor 0 (linha 5). Finalmente, o Algoritmo 7 é executado novamente (linha 6). Caso contrário, o Algoritmo 9 é finalizado e a etapa 5 é executada. As notações $n(r,T)$, $t_{r,n(r,T)}$, $s_{n(r,T)}$, determinam o aterro sanitário disponível mais próximo do cliente r, o tempo de deslocamento do cliente r até o aterro sanitário determinado e o tempo de serviço para descarga do lixo, respectivamente.

A quinta etapa da heurística construtiva é executada quando não existirem mais clientes do conjunto B que satisfaçam as condições das etapas 3 e 4 dos Algoritmos 8 e 9. O pseudocódigo da etapa 5 é descrito no Algoritmo 10.

Nesta etapa, o cliente inserido à rota será aquele que possuir o menor tempo de espera para alcançar seu intervalo de tempo e satisfazer as condições 1, 3, 4, 5, 6,

Algoritmo 9: Etapa 4

```
1  begin
2      if Qatual > 0 then
3          T ← T + t_{r,n(r,T)} + s_{n(r,T)}
4          r ← n(r, T)
5          Qatual ← 0
6          go to etapa 2
7      end
8      else
9          go to etapa 5
10     end
11 end
```

7 descritas na etapa 3 e a expressão $T + t_{rj} < E_j$, que assegura que o tempo atual T do veículo acrescido do tempo de deslocamento do cliente r até o cliente j seja menor que a janela de espera inicial do cliente j.

Neste algoritmo, a notação θ é representada por $\theta = E_j + s_j$, onde θ recebe a soma do tempo de espera da janela inicial do cliente j e o tempo de serviço gasto para coletar o lixo no cliente j.

Se o cliente i satisfaz as condições (linha 3), as variáveis de controle do veículo são atualizadas (linha 4 - 8), em seguida o cliente i escolhido é removido do conjunto (B) (linha 9) e o Algoritmo 2 é executado novamente (linha 10). Caso contrário, o Algoritmo 1 é executado e uma nova rota é inicializada. O algoritmo construtivo é encerrado quando todos os clientes são roteados.

4.3 Estruturas de vizinhança

Nesta seção, são apresentadas as diferentes estruturas de vizinhança utilizadas e a definição do conceito de vizinho para este problema.

As estruturas de vizinhança são estratégias utilizadas para realização de movimentos de melhora da solução obtida pela heurística construtiva. Essas estruturas são utilizadas na fase de busca local das heurísticas propostas e podem ser classificadas de duas formas distintas. A primeira forma de classificação é definida como estruturas de vizinhanças **intra-rotas**, onde os clientes são movidos dentro de uma mesma rota. Já nas estruturas **inter-rotas**, os clientes são movidos para rotas diferentes.

No procedimento de busca local é utilizado o conceito de vizinho. Seja, $N(nc, i)$ o conjunto de vizinhos, onde nc representa cardinalidade do conjunto e i o cliente.

Algoritmo 10: Etapa 5

```
1  begin
2      i = arg min{E_j | j ∈ B, T+ t_rj < E_j, Qatual + q_j ≤ C, Qtotal + q_j ≤
       R, Stotal + 1 ≤ S, θ + t_j,n(j,θ)+ s_n(j,θ)+ t_n(j,θ),0 ≤ b_0, θ ≤ b_u se
       rest = 0, onde θ = E_j + s_j}
3      if i satisfaz as condições then
4          T ← E_i+ s_i // E_i representa tempo de espera para janela inicial
           abrir
5          r ← i
6          Qtotal ← Qtotal + q_i
7          Qcurrent ← Qcurrent + q_i
8          Stotal ← Stotal + 1
9          B← B - {i}
10         go to etapa 2
11     end
12     else
13         go to etapa 1
14     end
15 end
```

Temos que, vizinhos podem ser definidos como: clientes que possuem janelas de tempo compatíveis. Ou seja, o cliente j é definido vizinho do cliente i se for possível visitar o cliente i, realizar o serviço em i, deslocar até o cliente j e realizar o serviço em j, sem esperar por sua janela de tempo abrir.

A seguir, as estruturas de vizinhança Swap, Shift, Or-Opt2, Or-Opt3, Exchange, InsertCustomer, InsertDisposalSite e ExchangeDisposalSite, serão detalhadas conforme seu funcionamento e classificadas de acordo com sua utilização.

4.3.1 Vizinhanças intra-rotas

As estruturas de vizinhanças intra-rotas são definidas como movimentos que acontecem em uma mesma rota. Neste trabalho foram implementadas seis estruturas de vizinhanças intra-rotas. São elas:

- N^1 - ***Or-opt2*:** esta vizinhança é formada por soluções obtidas retirando dois clientes adjacentes da rota e inserindo-os sequencialmente em outra posição da rota. Na Figura 4.2, os clientes 6 e 7 são movidos para outra posição da Rota 1.

- N^2 - ***Or-opt3*:** esta vizinhança é formada por soluções obtidas retirando três clientes adjacentes da rota e e inserido-os sequencialmente em outra posição

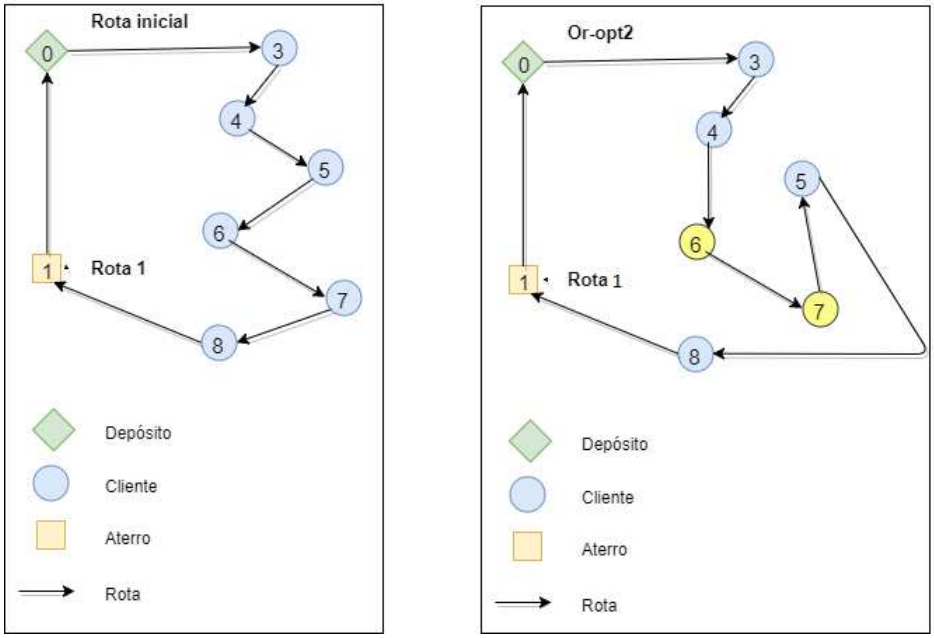

Figura 4.2: Or-opt2

da rota. Na Figura 4.3, os clientes 3, 4 e 5 são movidos para outra posição da Rota 1.

- N^3 - ***Exchange*:** esta vizinhança é formada por soluções obtidas permutando-se o posicionamento de dois clientes da rota. A Figura 4.4 ilustra a troca de posição do cliente 4 com o 7.
- N^4 - ***InsertCustomer*:** esta vizinhança é formada por soluções obtidas retirando clientes de uma rota e inserindo-os em outra posição da rota.
- N^5 - ***InsertDisposalSite*:** esta vizinhança é formada por soluções obtidas retirando-se os aterros sanitários de uma rota e inserindo-os em outra posição da rota.
- N^6 - ***ExchangeDisposalSite*:** esta vizinhança é formada por soluções obtidas permutando o posicionamento dos aterros sanitários na rota.

4.3.2 Vizinhanças inter-rotas

Nas vizinhanças inter-rotas ocorrem movimentos de clientes entre duas rotas distintas. Foram implementadas quatro estruturas de vizinhanças inter-rotas. São elas:

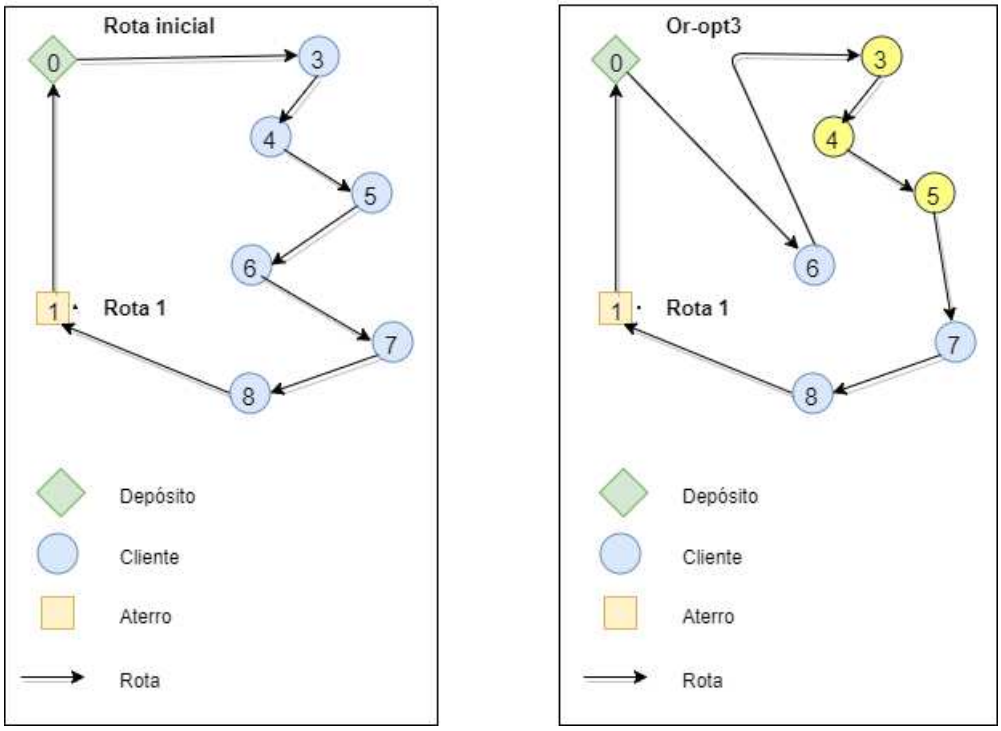

Figura 4.3: Or-opt3

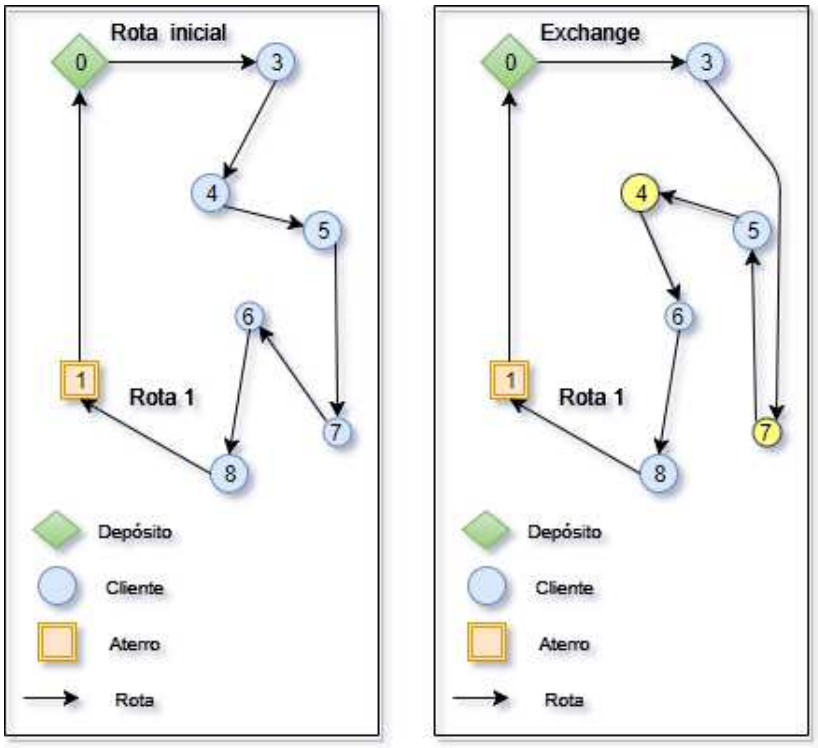

Figura 4.4: Exchange

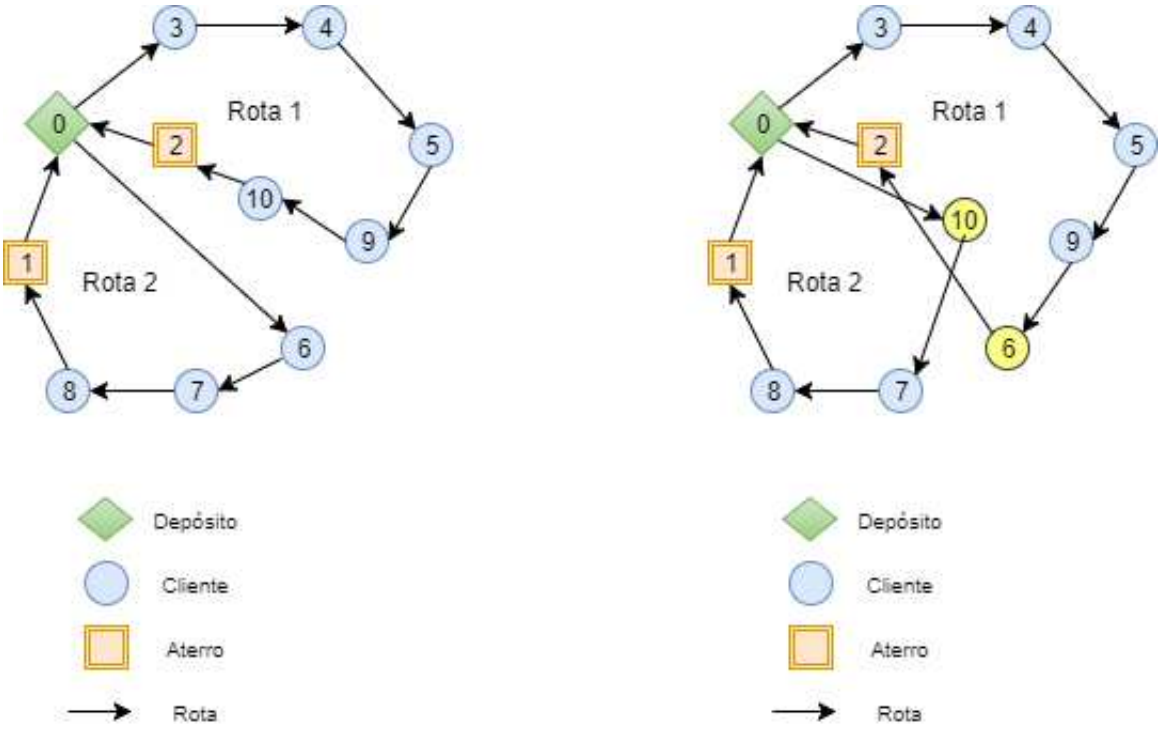

Figura 4.5: Swap (1,1)

- N^7 - ***Swap(1,1)***: Permuta dois clientes de rotas distintas. Na Figura 4.5, o cliente 10 da Rota 1 é trocado com o cliente 6 da Rota 2.
- N^8 - ***Swap(2,2)*****:** É similar ao Swap(1,1), porém dois clientes consecutivos são permutados. A ilustração dessa estrutura de vizinhança pode ser verificada na Figura 4.6, onde os clientes 9 e 10 pertencentes à Rota 1 são trocados pelos clientes 6 e 7 do conjunto de clientes da Rota 2.
- N^9 - ***Swap(2,1)*****:** É a permutação de dois clientes adjacentes com um outro cliente que pertence a uma rota distinta, conforme ilustrado na Figura 4.7, onde os clientes adjacentes 6 e 7 da Rota 2 são trocados pelo cliente 10 da Rota 1.
- N^{10} - ***Shift(1,0)*****:** Um cliente é removido de uma rota e inserido em outra. Na Figura 4.8, o cliente 9 é realocado da Rota 1 para Rota 2.

4.4 Algoritmo ILS-VND

O algoritmo proposto para o refinamento da solução é feito através de um algoritmo híbrido, denominado de ILS-VND, que combina as metaheurísticas ILS [Lourenço

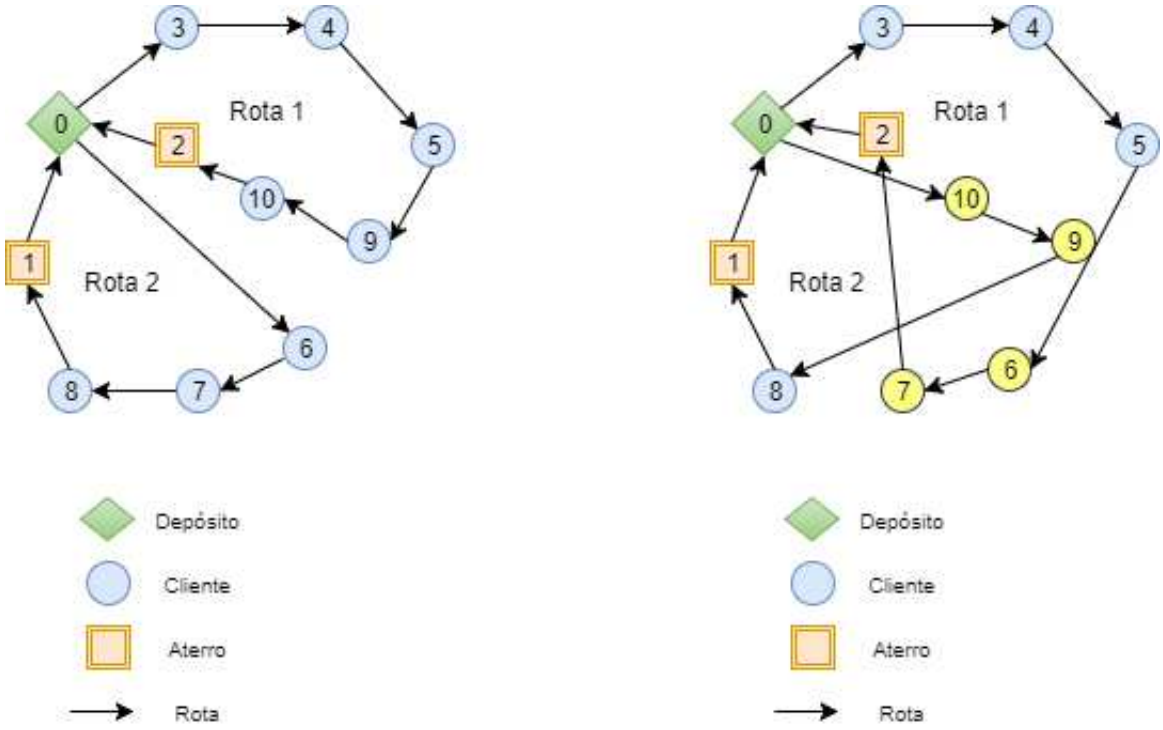

Figura 4.6: Swap (2,2)

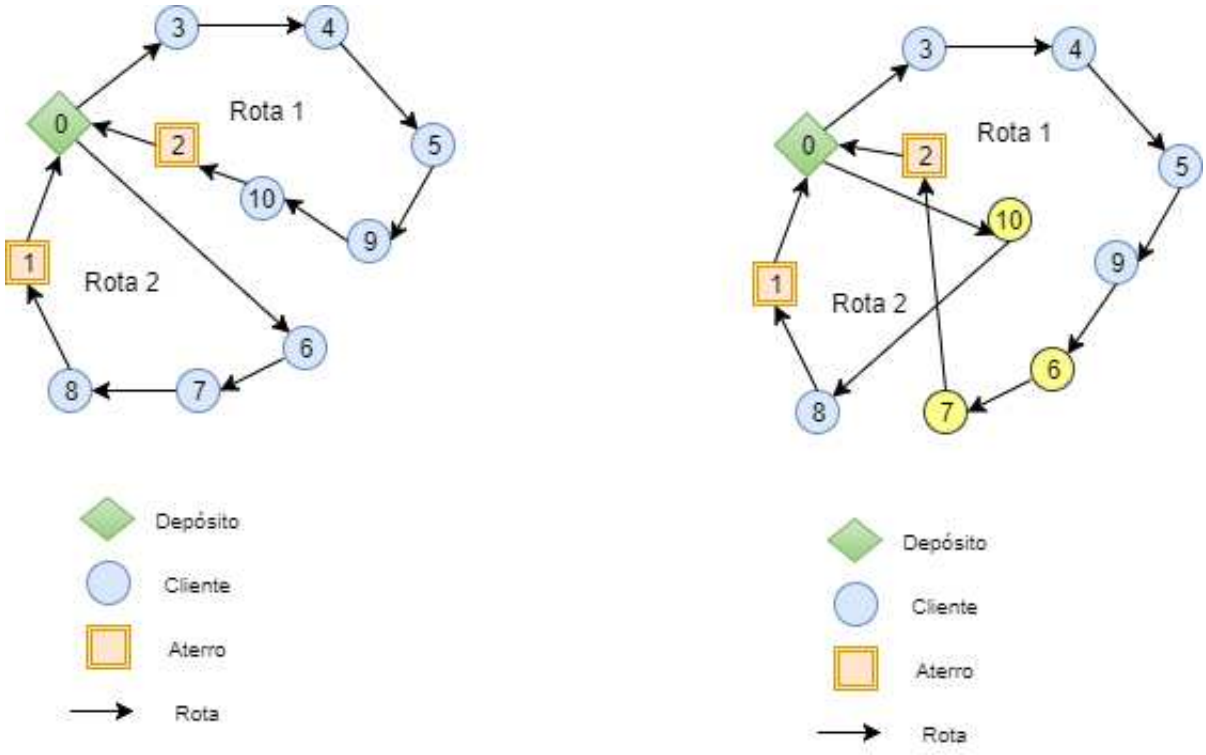

Figura 4.7: Swap (2,1)

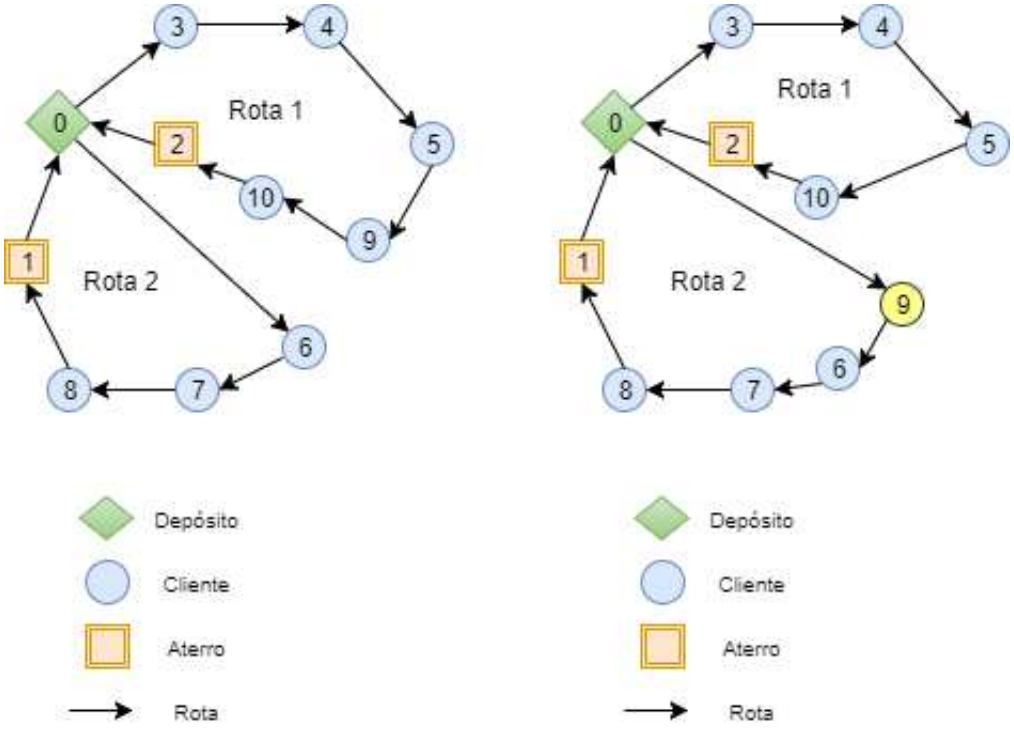

Figura 4.8: Shift (1,0)

et al., 2003] e VND [Mladenović & Hansen, 1997]. O Algoritmo 11 apresenta o pseudocódigo do ILS-VND. O algoritmo é executado iterativamente até completar o critério de parada, número de iterações ($MaxIterILS$) (linha 10). Inicialmente, uma solução inicial é construída pelo método heurística construtiva, descrito na Seção 4.2. Em seguida, aplica-se a busca local VND1 sobre s_0 (linha 3) de modo que a solução resultante é atribuída a s^*. Posteriormente, inicia-se um processo iterativo que tem, como critério de parada, o número de iterações. A cada iteração, a solução corrente s^* é modificada com a aplicação da perturbação, gerando uma nova solução s' (linha 5). Essa solução s' é refinada através do algoritmo de busca local VND2 (linha 6), descritas a seguir. Se a função objetivo de (s'') for menor que a de (s^*), então s^* é atualizada (linhas 7 - 9). Em seguida, o contador ($IterILS$) é incrementado. Ao final, a melhor solução encontrada s^* é retornada (linha 12).

4.4.1 Busca local VND

No ILS-VND foram utilizadas dois diferentes procedimentos de busca local VND, denominados de VND1 e VND2. As VND1 e VND2 consistem em realizar uma busca local na solução inicial e na solução perturbada, respectivamente. A busca local VND1 utiliza a estrutura de vizinhança N^3 (Exchange). Já a busca local VND2

Algoritmo 11: ILS-VND(Max_Iter, K^*)

```
1  begin
2      s_0 ← GerarSoluçãoInicial();
3      s* ← BuscaLocalVND1(s_0, K*);
4      while iterILS ≤ Max_IterILS do
5          s' ← Perturbar(s*);
6          s'' ← BuscalLocalVND2 (s');
7          if f(s'') < f(s*) then
8              s* ← s'';
9          end
10         iterILS ← iterILS + 1;
11     end
12     return s*;
13 end
```

utiliza as estruturas de vizinhança N^4, N^5 e N^6 (InsertCustomer, InsertDisposalSite, ExchangeDisposalSite).

O Algoritmo 12 apresenta o pseudocódigo da busca local VND1. Inicialmente, o algoritmo inicializa Γ com o tamanho das vizinhanças (linha 2). Em seguida, se Γ não estiver vazio, então nc recebe o menor valor do conjunto K^* (linhas 3 - 4) e este valor é retirado do conjunto Γ (linha 5). A cada execução do laço principal (linhas 6 - 17), o cliente i é trocado com seu vizinho j. Caso a solução obtida seja viável e melhorada, o valor da melhor solução é atualizado e Γ reinicializado com todas as vizinhanças, caso contrário, a troca entre os clientes é desfeita. O laço de repetições termina quando não houver mais vizinhanças a serem exploradas. Finalmente, na linha 19 é retornada a melhor solução encontrada s^*.

O Algoritmo 13 apresenta o pseudocódigo da busca local VND2. Este algoritmo utiliza as estruturas de vizinhança (N^4, N^5, N^6) na busca local. Inicialmente, k recebe a primeira estrutura de vizinhança (linha 3). No laço principal (linhas 4 - 12) as estruturas de vizinhanças são exploradas até que a melhor solução seja encontrada. A primeira estrutura de vizinhança é explorada até um ótimo local ser encontrado (linha 5), caso a solução obtida seja melhorada, o valor da melhor solução é atualizado e k é reinicializado com a primeira estrutura de vizinhança (linha 7). Caso contrário, k é incrementado e uma nova estrutura de vizinhança é explorada (linha 10). O laço de repetições termina quando não houver mais vizinhanças a serem exploradas. Finalmente, a melhor solução encontrada s^* é retornada (linha 13).

Algoritmo 12: VND1(s, K^*)

```
1  begin
2  |  Γ ← K*; (conjunto com o tamanho das vizinhanças)
3  |  while Γ ≠ ∅ do
4  |  |  nc ← min{k|k ∈ Γ};
5  |  |  Γ ← Γ − {nc};
6  |  |  for all clientes i do
7  |  |  |  for all clientes j ∈ N(nc, i) do
8  |  |  |  |  s' ← valor obtido pela troca de i e j;
9  |  |  |  |  Avaliar a solução s';
10 |  |  |  |  if f(s') < f(s) and s' == viavel then
11 |  |  |  |  |  s* ← s'; Γ ← K*;
12 |  |  |  |  end
13 |  |  |  |  else
14 |  |  |  |  |  Desfaz a troca entre os clientes
15 |  |  |  |  end
16 |  |  |  end
17 |  |  end
18 |  end
19 |  return s*;
20 end
```

4.4.2 Perturbação

Mecanismos de perturbação permitem a movimentação de clientes, sem que haja verificação de redução de custo, com a finalidade de diversificar uma solução.

O critério de aceitação do movimento de perturbação é que ele não pode gerar uma solução inviável. Como consequência, podem gerar soluções com custos piores que os das soluções já encontradas, porém permitem que o algoritmo escape de mínimos locais. A perturbação utilizada neste algoritmo consiste em três trocas de clientes aleatórios entre as rotas.

O Algoritmo ILS-VND apresenta os seguintes parâmetros: contador de iteração ($IterILS$), lista com tamanho das vizinhanças utilizadas (K^*). Estes foram definidos na Seção 5.4.

4.5 Algoritmo ILS-RVND

O algoritmo ILS-RVND é similar ao algoritmo ILS-VND. A diferença entre estes algoritmos está no procedimento de busca local. A busca local é realizada pelo procedimento RVND, o qual utiliza uma ordenação de vizinhança randômica. Neste

Algoritmo 13: VND2(s)

```
1  begin
2  |  Seja {N^4, N^5, N^6} o conjunto de estruturas de vizinhança;
3  |  k ← 1; {vizinhança atual}
4  |  while k ≤ 3 do
5  |  |  Encontrar o melhor vizinho s' ∈ N^k de s;
6  |  |  if (f(s') < f(s)) then
7  |  |  |  s* ← s'; k ← 1;
8  |  |  end
9  |  |  else
10 |  |  |  k ← k + 1;
11 |  |  end
12 |  end
13 |  return s*;
14 end
```

algoritmo, novas estruturas de vizinhanças foram consideradas nos procedimentos de busca local e perturbação.

O Algoritmo 14 apresenta o pseudocódigo do algoritmo ILS-RVND. A abordagem é executada iterativamente até completar o critério de parada, número de iterações ($MaxIterILS$) do algoritmo (linha 11). Este parâmetro $MaxIterILS$ é definido durante os experimentos computacionais, na Seção 5.4. Inicialmente, é gerada uma solução inicial s_0 usando a heurística construtiva (linha 2). Em seguida, aplica-se a busca local RNVD sobre s_0 (linha 3) de modo que a solução resultante é atribuída a s^*. Posteriormente, inicia-se um processo iterativo que tem, como critério de parada, o número de iterações. A cada iteração, a solução corrente s^* é modificada com a aplicação da perturbação, gerando a solução s' (linha 5). Essa solução s' é refinada através do algoritmo de busca local RVND (linha 6). Caso a solução s'' seja melhor que a solução s^*, ela é utilizada como ponto de partida para a próxima iteração, senão é mantida a solução s^* (linhas 7 - 9). Ao final do Algoritmo ILS-RVND é retornada a melhor solução encontrada s^* (linha 12).

O processo de busca local é repetido em todas as iterações do algoritmo, após a aplicação de uma perturbação. No ILS-RVND é utilizado o algoritmo de busca local RVND, que emprega todas as estruturas de vizinhanças, descritas na Seção 4.3.

Algoritmo 14: ILS-RNVD

```
1  begin
2      s_0 ← GerarSoluçãoInicial();
3      s* ←BuscaLocalRVND(s_0);
4      while iterILS < MaxIterILS do
5          s' ← Perturbar(s*);
6          s'' ← BuscaLocalRVND (s');
7          if f(s'') < f(s*) then
8              s* ← s''
9          end
10         iterILS ← iterILS + 1
11     end
12     return s*;
13 end
```

4.5.1 Busca local RVND

Na busca local RVND, são utilizadas as sete estruturas de vizinhança propostas neste trabalho e descritas na Seção 4.3. Entretanto, não é estabelecida a ordem de aplicação de cada uma dessas estruturas, ou seja, a cada chamada do procedimento, uma ordem de processamento é definida aleatoriamente. O critério de aceite da nova solução são os valores da função de avaliação. Se a solução s' for melhor do que a solução corrente, a solução corrente s recebe a melhor solução e a lista Γ é reinicializada com todas as vizinhanças. Caso contrário, remove-se a vizinhança nc da lista Γ. O algoritmo finaliza quando a lista de vizinhanças estiver vazia, isto é, quando analisam-se todas as vizinhanças e não é encontrada uma solução melhor.

O Algoritmo 15 apresenta o pseudocódigo da implementação realizada na busca local RVND.

4.5.2 Perturbação

Neste algoritmo, foram utilizadas duas estruturas de perturbação escolhidas aleatoriamente em cada iteração do laço principal do ILS. São as seguintes:

- **Random-Swap(1,1)** - consiste na aplicação do movimento Swap(1,1) de forma aleatória.
- **Random-Swap(2,2)** - consiste na aplicação do movimento Swap(2,2) de forma aleatória.

Algoritmo 15: Busca Local RVND

```
1  begin
2      Γ ← Inicializar (Γ)
3      while Γ ≠ ∅ do
4          nc ← Aleatorio (N^1 a N^7)
5          s' ← MelhorVizinho (s, nc)
6          if f(s') < f(s) then
7              s ← s';
8              Γ ← Inicializar(Γ)
9          end
10         else
11             Γ ← Γ − {nc};
12         end
13     end
14     return s;
15 end
```

4.6 Algoritmo SA

O Algoritmo 16 descreve os passos realizados pela metaheurística SA proposta para o WCVRPTW.

O algoritmo inicia de uma solução inicial gerada conforme Seção 4.2 (linha 2). Enquanto os critérios de parada não são satisfeitos (linhas 4 - 6), o algoritmo seleciona aleatoriamente uma solução vizinha a partir da solução atual usando a estrutura de vizinhança N^7 (linha 7). O critério para aceitar uma solução como a solução atual é feito em função de uma probabilidade determinada de acordo com a qualidade da solução e com a temperatura atual (linhas 8 - 9). Caso o critério de aceitação seja válido, o valor da melhor solução atual (s) recebe o valor da nova solução (s') (linha 10). Em seguida, é verificado se o valor da nova solução (s') é superior ao valor da melhor solução encontrada até o momento (s^*) (linha 11), caso verdadeiro, então esta possui seu valor atualizado (linha 12). Posteriormente, o contador de iteração é incrementado (linha 15) e a temperatura T é modificada pela taxa de resfriamento (linha 17). O algoritmo é encerrado quando a temperatura atual atingir ou estiver abaixo da temperatura mínima. Para finalizar, o algoritmo retorna a melhor solução encontrada durante o processo de otimização (linha 19).

O Algoritmo SA apresenta quatro parâmetros: temperatura inicial (T_i), temperatura final (T_f), número de iterações em cada temperatura ($SAmax$) e taxa de resfriamento (α). Estes quatro parâmetros foram calibrados na Seção 5.4.

Algoritmo 16: Simulated Annealing

```
1  begin
2      s* ← s
3      T ← T_i
4      while T > T_f do
5          IterT ← 0
6          while IterT < SAmax do
7              s' ← gerar aleatoriamente uma solução vizinha s' ∈ N(s)
8              Δ = f(s') − f(s)
9              if (Δ < 0) or random [0,1] < e^{−Δ/T} then
10                 s ← s'
11                 if (f(s') < f(s*)) then
12                     s* ← s'
13                 end
14             end
15             IterT ← IterT + 1
16         end
17         T ← T x (1 − α)
18     end
19     return s*
20 end
```

4.7 Algoritmo TS

Os Algoritmos 17 e 18 descrevem os passos realizados pela metaheurística TS proposta por Benjamin & Beasley [2010] para o WCVRPTW.

O algoritmo TS considera como movimento a troca de dois clientes que estão ou não na mesma rota. E possui as seguintes variáveis:

- Δ: representa quanto tempo o cliente ficará na lista tabu.
- M: contador de iteração da lista tabu
- $\delta(i)$: armazena a última iteração do cliente i. É utilizado para julgar se o movimento do cliente i é tabu ou não.
- *zmove* valor da solução após a troca dos clientes.
- *zcurrent*: valor da solução atual.
- *zbest*: melhor valor encontrado pelo algoritmo.
- m: controla o número de troca dos pares de clientes em que não houve melhora da solução *zbest* ou *zcurrent*.

- *znon*: permite movimentos que não sejam de melhoria da solução. Esta variável armazena o melhor valor de não melhoria da solução.

- ϕ: fator de diversificação da solução atual (*zcurrent*). O valor considerado é: $\geq zcurrent + \phi$.

O algoritmo inicia atribuindo aos contadores M e m o valor 0 (linha 1) e garantido que a lista tabu esteja vazia (linha 2). Em seguida o primeiro passo do Algoritmo 17 é iniciado, as variáveis $Znon$, m e $flag$ são atribuídas aos respectivos valores, ∞, $m+1$ e 0 (linhas 4 - 6). A variável $flag$ representa se houve mudanças nas variáveis *zcurrent* e *zbest* durante o primeiro passo. Posteriormente, é realizada a tentativa de troca do cliente i com o cliente j (o cliente j deve ser vizinho do cliente i)(linhas 7 - 10). Caso a solução *zmove* seja viável são realizadas três verificações, sendo que a primeira ocorre independentemente de estar na lista tabu (critério aspiração) e as demais ocorrem quando o movimento não está na lista tabu.

- o valor da nova solução(*zmove*) é o melhor valor encontrado até o momento, ou seja, $zmove < zbest$ (linha 12). Se verdadeiro, então as variáveis do problema são atualizadas (linhas 13-19).

- o valor da nova solução (*zmove*) é melhor do que a solução atual(*zcurrent*), ou seja, $zmove < zcurrent$ (linha 25). Se verdadeiro, então as variáveis do problema são atualizadas (linhas 26-30).

- o valor da nova solução (*zmove*) é melhor que o movimento de não melhora da solução(*znon*) e $zmove \geq zcurrent + \phi$ (linha 32), esta última condição tem como objetivo escapar do ótimo local. Se verdadeiro, então *znon* é atualizado com o valor de *zmove* (linha 33) e é armazenado o valor do movimento dos clientes i e j na lista tabu (linha 34).

Caso nenhuma das três verificações sejam verdadeiras, então a troca dos cliente é desfeita e um novo par de clientes é considerado (linhas 38 - 40).

O passo 2 inicia verificando se já foram executados m movimentos sem que tenha ocorrido melhoria (linha 1). Caso a flag seja igual 1 (linha 2), então houve melhoria de *zbest* ou *zcurrent*, com isso o algoritmo retorna ao passo 1 (linha 3). Caso contrário, é armazenado um novo melhor valor de não melhoria da solução com a *zcurrent* recebendo este valor (linha 6). O movimento dos clientes i e j são armazenados na lista tabu e o seu contador é incrementado (linhas 7- 8). Após

isso o algoritmo retorna ao passo 1 (linha 9). Este algoritmo é encerrado quando o critério de parada é satisfeito, ou seja, m atinge o número máximo de iterações.

O Algoritmo TS apresenta três parâmetros: contador de iteração (m), tamanho da lista tabu (Δ) e fator de diversificação (ϕ). Estes apresentam os respectivos valores: 5, 7, $zbest/(20 * quantidadeClientes)$. Foram considerados os valores dos parâmetros definidos por Benjamin & Beasley [2010], com o intuito de comparar o desempenho dos algoritmos.

Algoritmo 17: Tabu Search - Passo 1

```
1  M ← 0 m ← 0
2  δ(i) ← −(Δ + 1)
3  begin
4      Znon ← ∞
5      m ← m + 1
6      flag ← 0
7      for all clientes i do
8          for all clientes j ∈ N(i, nc) do
9              zmove ← valor obtido pela troca de i e j
10             Avaliar a solução zmove;
11             if (zmove == viavel) then
12                 if (zmove < zbest) then
13                     zbest ← zmove
14                     zcurrent ← zbest
15                     δ(i) = δ(j) = M
16                     M ← M + 1
17                     Znon ← ∞
18                     m ← 0
19                     flag ← 1
20                 end
21                 if (|M − δ(i) ≤ Δ| or |M − δ(j) ≤ Δ|) then
22                     Desfaz a troca entre os clientes
23                 end
24                 else
25                     if (zmove < zcurrent) then
26                         zcurrent ← zmove
27                         δ(i) = δ(j) = M
28                         M ← M + 1
29                         Znon ← ∞
30                         flag ← 1
31                     end
32                     if (zmove < Znon and zmove ≥ zcurrent + φ) then
33                         Znon ← Zmove
34                         α ← i ; β ← j
35                     end
36                 end
37             end
38             else
39                 Desfaz a troca entre os clientes
40             end
41         end
42     end
43 end
```

Algoritmo 18: Tabu Search - Passo 2

```
if m<5 then
    if (flag == 1) then
        go to Passo 1
    end
    else
        Zcurrent ← Znon
        δ(α) = δ(β) = M
        M ← M + 1
        go to Passo 1
    end
end
return Zbest
```

Capítulo 5

Experimentos Computacionais

Neste capítulo, são apresentadas as características das instâncias, a calibração dos parâmetros dos algoritmos heurísticos e os resultados alcançados pelos diferentes algoritmos propostos para o WCVRPTW.

5.1 Ambiente computacional utilizado

Para realização dos testes computacionais, todos os algoritmos foram codificados na linguagem C++ e executados em um computador Intel Core i7 CPU @ 3.40GHz x 8, com 32GB de memória RAM e sistema operacional Ubuntu 15.10 LTS, 64 bits. Já a abordagem exata (formulação MILP) foi implementada via ILOG Concert Technology C++ e foi executada utilizando o software ILOG CPLEX versão acadêmica 12.5, com todas as configurações padrões, exceto para o tempo de execução, que foi limitado a 1 hora. Para análise estatística foi utilizado o software Minitab (versão trial).

5.2 Métrica para avaliação dos algoritmos heurísticos

Para verificar a qualidade nos experimentos de comparação dos algoritmos foi utilizada a medida do Desvio Percentual Relativo (RPD em inglês *Relative Percentage Deviation*) com relação à melhor solução conhecida. O RPD é determinado da seguinte maneira:

$$RPD = \frac{f_{atual} - f_{melhor}}{f_{melhor}} \times 100\% \quad (5.1)$$

Em que f_{atual} é o valor da solução obtida pelo algoritmo, para uma determinada instância, e f_{melhor} é o melhor resultado conhecido obtido dentre todos os algoritmos, para esta instância. Cada instância foi executada uma vez pelo *solver* e dez vezes pelas heurísticas. A análise considera o melhor resultado das dez execuções.

5.3 Instâncias do problema

Para testar os algoritmos apresentados no Capítulo 4, foram utilizadas, como referência, as 10 instâncias propostas por Kim et al. [2006], para o WCVRPTW. Estas instâncias variam de acordo com o números de clientes, aterros sanitários, demandas, tamanho dos intervalos das janelas de tempo e tempo de serviço.

A Tabela 5.1 detalha a forma de apresentação das informações das instâncias e apresenta, como exemplo, parte dos dados da instância 102_stop, sendo que:

- *Capacidade do veículo* representa a capacidade máxima de carga do veículo. Como a frota utilizada é homogênea, esse valor é o mesmo para todos os veículos da frota.
- *Carga máxima* representa o volume máximo de carga que cada veículo pode coletar por dia.
- *Cliente máximo* representa o número máximo de clientes que cada veículo pode atender por dia.
- *Duração almoço* representa o tempo que o motorista fica parado durante o período de almoço, em segundos.
- *Velocidade* representa a velocidade do veículo, parâmetro utilizado para calcular a distância entre os pontos de parada.
- *Id* representa o número de identificação do ponto de parada (depósito, aterro sanitário ou cliente).
- X e Y representa as coordenadas geográficas que definem a localização geográfica do ponto de parada.
- *Early Time* e *Late Time* representam o intervalo de tempo em que o ponto de parada está disponível para receber o veículo, em horas.

- *Service Time* representa o tempo total gasto para realizar o serviço, em segundos.
- *Load* representa a demanda que o veículo deve coletar no ponto de parada.
- *Type* representa o tipo do ponto de parada (0- depósito, 1- cliente, 2- aterro sanitário).

Tabela 5.1: Informações da instância exemplo 102_stop

Capacidade do veículo: 280.0
Carga máxima: 400.0
Cliente máximo: 500
Duração almoço: 3600
Velocidade: 40

Id	X	Y	Early Time	Late Time	Service Time	Load	Type
0	1215029.00	3461398.00	0400	1500	0	9999.00	0
1	1229125.00	3460107.00	0000	2400	1600	9999.00	2
2	1422490.00	3563104.00	0000	2400	1600	9999.00	2
3	1208344.00	3469904.00	0700	2400	180	12.00	1
4	1214253.00	3472668.00	0000	2400	30	4.00	1
5	1201867.00	3467794.00	0000	0700	120	6.00	1
6	1200929.00	3463471.00	0700	2400	120	8.00	1
7	1209068.00	3464516.00	0700	2400	60	16.00	1
8	1209433.00	3461581.00	0000	0700	30	6.00	1
9	1209468.00	3454950.00	0000	2400	90	18.00	1
10	1381044.00	3453572.00	0000	2400	120	8.00	1

Neste trabalho, também propomos a criação de um novo *benchmark* com 60 instâncias de pequeno porte e 100 de médio porte. Para as instâncias de pequeno porte o número de clientes varia de 10 a 20 crescendo de 2 em 2, o que totaliza seis grupos de instâncias separadas pelo número de clientes. Cada grupo possui 10 instâncias. Para instâncias de médio porte, o número de clientes varia de 50 a 200 crescendo de 50 em 50, totalizando quatro grupos distintos. Este conjunto de instâncias foi gerado aleatoriamente.

5.4 Calibração dos parâmetros dos algoritmos heurísticos

Com o intuito de determinar a melhor abordagem dentre todas as propostas neste trabalho, foi realizado um conjunto de experimentos computacionais que tem como objetivo definir para cada uma, as configurações dos parâmetros que permitem as mesmas alcançarem seus melhores resultados.

Os experimentos foram realizados nos algoritmos SA, ILS-VND e ILS-RVND, utilizando uma amostra composta por 20% do conjunto total de instâncias. Para cada instância, cada algoritmo foi executado 10 vezes usando cada uma das diferentes configurações de parâmetros descritas nas Tabelas 5.2, 5.6. O desvio percentual relativo (*Relative Percentage Deviation* - RPD) foi utilizado como variável resposta na análise estatística.

O algoritmo heurístico SA (Algoritmo 16) apresenta quatro parâmetros controlados: temperatura inicial (T_i), temperatura final (T_f), número de iterações em cada temperatura ($SAmax$) e fator de resfriamento (α). Os diferentes valores testados para cada um destes parâmetros e suas combinações são mostrados nas Tabelas 5.2, 5.3.

A Figura 5.1 mostra o gráfico de médias resultante do teste HSD de Tukey com nível de confiança de 95% para as 18 configurações testadas. Percebe-se através desta figura que não existe diferença significativa entre as configurações, pois os intervalos se sobrepõem. No entanto, é possível ver nesta figura que a menor média de RPD é obtida com a combinação 3. Sendo assim, o algoritmo SA utiliza como parâmetros a configuração $Ti = 15$, $Tf = 10$, $SAmax = 100$ e $\alpha = 0.001$.

Tabela 5.2: Conjunto de valores testados na calibração do SA.

Parâmetros	Valores testados	Quantidade
T_i	15, 50	2
T_f	1, 5, 10	3
$SAmax$	100	1
α	0.001, 0.005, 0.010	3
Total		18

O algoritmo heurístico ILS-VND apresenta três parâmetros controlados: ordem de vizinhanças (V), número máximo de iterações ($MaxIterILS$) e tamanho das vizinhanças (Tam). Os diferentes valores para casa um destes parâmetros e suas combinações são mostrados nas Tabelas A.8, 5.5. E os resultados são apresentados no Apêndice A.

A Figura 5.2 mostra o gráfico de médias resultante do teste HSD de Tukey comnível de confiança de 95% para as 18 configurações testadas. Percebe-se através desta figura que não existe diferença significativa entre as configurações, pois os intervalos se sobrepõem. No entanto, é possível ver nesta figura que a menor média de RPD é obtida com a combinação 18. Sendo assim, o algoritmo ILS-VND utiliza

Tabela 5.3: Combinação de valores testados na calibração do SA.

Número	Combinação testada			
	T_i	T_f	$SAmax$	α
1	15	1	100	0.001
2	15	5	100	0.001
3	15	10	100	0.001
4	15	1	100	0.005
5	15	5	100	0.005
6	15	10	100	0.005
7	15	1	100	0.010
8	15	5	100	0.010
9	15	10	100	0.010
10	50	1	100	0.001
11	50	5	100	0.001
12	50	10	100	0.001
13	50	1	100	0.005
14	50	5	100	0.005
15	50	10	100	0.005
16	50	1	100	0.010
17	50	5	100	0.010
18	50	10	100	0.010

Tabela 5.4: Conjunto de valores testados na calibração do ILS-VND.

Parâmetros	Valores testados	Quantidade
V	N^4, N^5, N^6	6
$MaxIterILS$	10, 20, 50	3
Tam	5, 10, 25, 50	1
Total		18

como parâmetros a configuração $V = N^5$, N^4, N^6, $MaxIterILS = 50$ e $Tam = 5$, 10, 25, 50.

O algoritmo heurístico ILS-RVND (Algoritmo 14) apresenta apenas um parâmetro controlado: o número máximo de iterações ($MaxIterILS$). A Tabela 5.6 apresenta os valores testados para este parâmetro. E a Figura 5.3 mostra o gráfico de médias resultante do teste HSD de Tukey com nível de confiança de 95%.

É possível ver pela Figura 5.3 que a configuração $MaxIterILS$ com 1000 iterações apresenta melhores resultados que as demais configurações testadas. Sendo assim, o procedimento ILS do algoritmo ILS-RVND utiliza esta configuração.

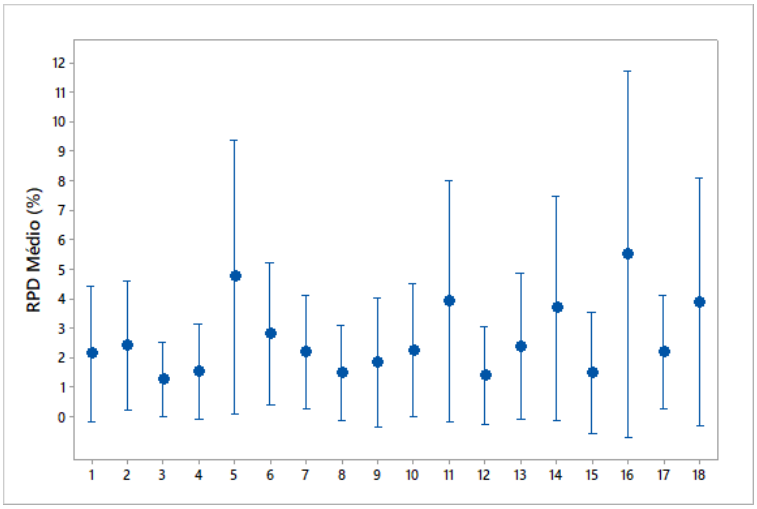

Figura 5.1: Gráfico de Médias e intervalos HSD de Tukey com nível de confiança de 95% para a calibração dos parâmetros do SA.

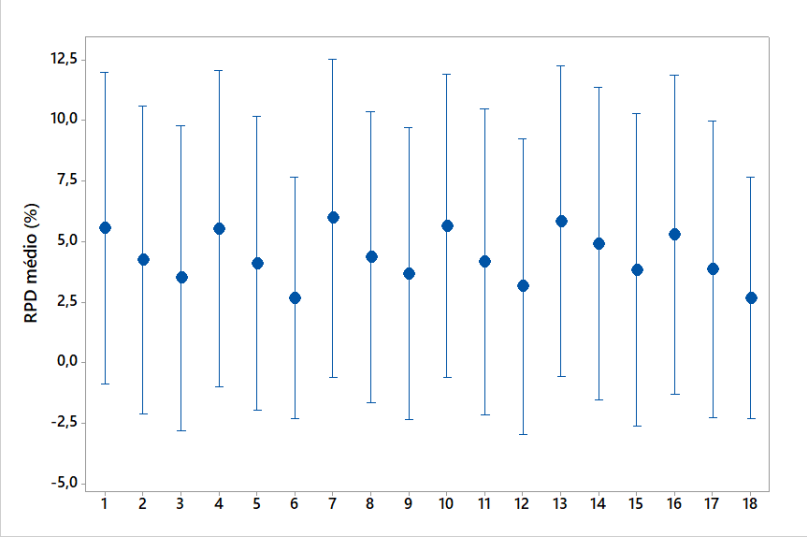

Figura 5.2: Gráfico de Médias e intervalos HSD de Tukey com nível de confiança de 95% para a calibração dos parâmetros do ILS-VND.

Tabela 5.5: Combinação de valores testados na calibração do ILS-VND.

Número	Combinação testada		
	V	$MaxIterILS$	Tam
1	N^4 N^5 N^6	10	5,10,25,50
2	N^4 N^5 N^6	20	5,10,25,50
3	N^4 N^5 N^6	50	5,10,25,50
4	N^4 N^6 N^5	10	5,10,25,50
5	N^4 N^6 N^5	20	5,10,25,50
6	N^4 N^6 N^5	50	5,10,25,50
7	N^5 N^6 N^4	10	5,10,25,50
8	N^5 N^6 N^4	20	5,10,25,50
9	N^5 N^6 N^4	50	5,10,25,50
10	N^6 N^4 N^5	10	5,10,25,50
11	N^6 N^4 N^5	20	5,10,25,50
12	N^6 N^4 N^5	50	5,10,25,50
13	N^6 N^5 N^4	10	5,10,25,50
14	N^6 N^5 N^4	20	5,10,25,50
15	N^6 N^5 N^4	50	5,10,25,50
16	N^5 N^4 N^6	10	5,10,25,50
17	N^5 N^4 N^6	20	5,10,25,50
18	N^5 N^4 N^6	50	5,10,25,50

Tabela 5.6: Conjunto de valores testados na calibração do ILS-RVND.

Parâmetro	Valores testados
$MaxIterILS$	50, 100, 500, 1000

5.5 Avaliação das heurísticas nas instâncias geradas

O primeiro conjunto de testes analisado será em relação às novas instâncias que foram geradas. Este novo *benchmark* de instâncias foi dividido em dois grupos, as de pequeno porte, com número de clientes variando de 10 a 20 e o número de aterros sanitários variando entre 1 e 2, e as de médio porte, com número de clientes variando de 50 a 200 e os aterros sanitários variando de 2 a 5. Para o primeiro grupo, pequeno porte, executamos o modelo matemático utilizando o *solver* CPLEX e os algoritmos ILS-VND, ILS-RVND, TS e SA. Para o segundo grupo, médio porte, foram executados apenas os algoritmos propostos. A seguir, são apresentados separadamente os

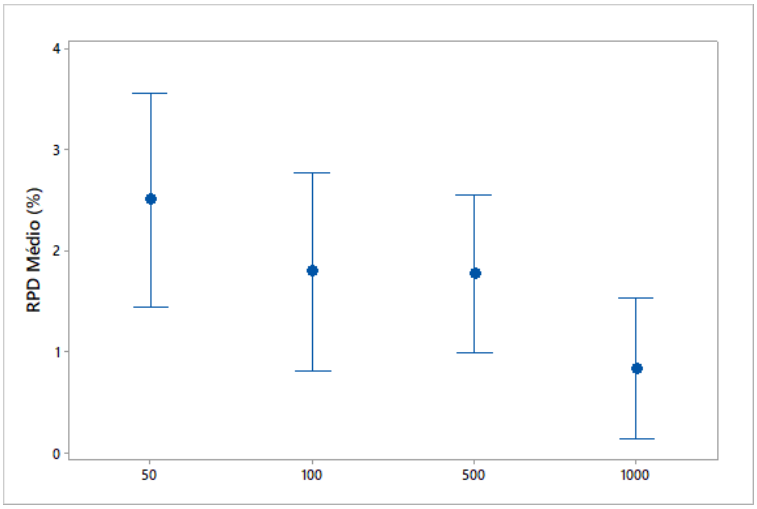

Figura 5.3: Gráfico de Médias e intervalos HSD de Tukey com nível de confiança de 95% para a calibração dos parâmetros do ILS-RVND.

resultados obtidos.

5.5.1 Resultados para instâncias de pequeno porte

Os resultados obtidos pelo modelo matemático e pelos algoritmos heurísticos propostos são apresentados nas Tabelas 5.7 e 5.8, enquanto a Tabela 5.9 apresenta resultados estatísticos que comparam os diferentes algoritmos de resolução do WC-VRPTW.

As Tabelas 5.7 e 5.8 reportam os resultados das instâncias de pequeno porte (p1 até p60). As duas primeiras colunas de cada uma dessas tabelas descrevem as características das instâncias, seguidas das colunas *nv*, Best, Time(s) que informam, respectivamente, o número de veículos utilizados para solução do problema, o valor da melhor solução obtida pelo CPLEX e o tempo de processamento. As colunas seguintes apresentam os melhores resultados obtidos pelos algoritmos heurísticos propostos. Estes são apresentadas em quatro colunas, sendo que *nv*, Best, Avg, Time(s), informam, respectivamente, o número de veículos utilizados, o valor da melhor solução, o valor da solução média obtida e o tempo total de processamento gasto em 10 execuções. É importante apontar que os melhores resultados foram destacados em negrito com o intuito de evidenciar a qualidade dos resultados alcançados por cada algoritmo.

Para uma melhor disposição visual dos resultados apresentados nas Tabelas 5.7 e 5.8, agrupamos os resultados por número de clientes. Cada grupo de clientes contém a média do valor obtido por cada um dos algoritmos para todas as instâncias

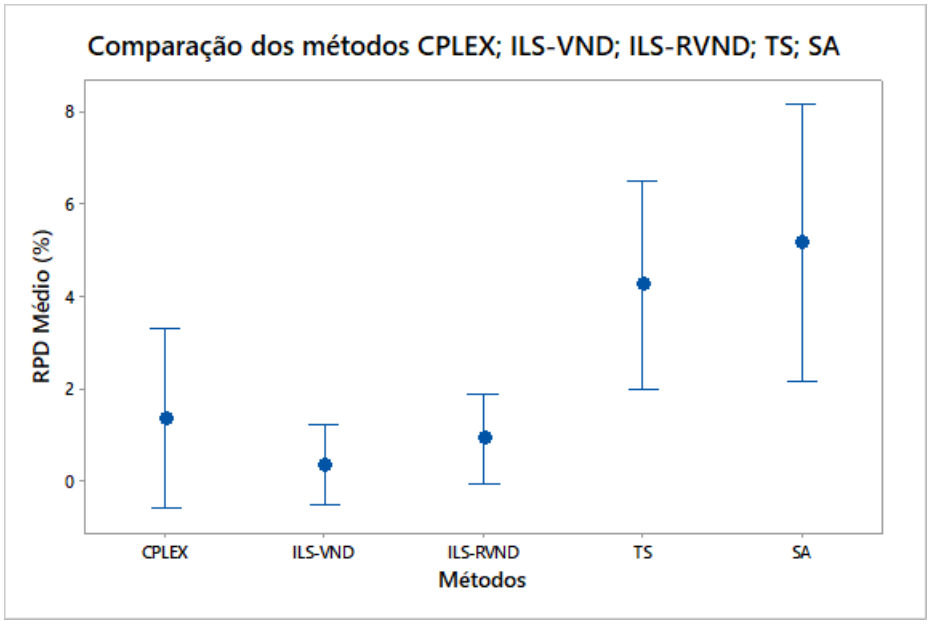

Figura 5.4: Comparação de algoritmos para instâncias de pequeno porte

com o respectivo número de clientes. A Tabela 5.9 e a Figura 5.4 avaliam o desempenho do modelo matemático (CPLEX), e dos algoritmos propostos ILS-VND, ILS-RVND, TS e SA.

Como pode-se observar, o algoritmo que obteve melhor desempenho para esse grupo de instâncias foi o ILS-VND. Das 60 instâncias testadas ele conseguiu o melhor resultado em 42 delas. O ILS-RVND obteve o melhor resultado em 37. Já, o TS, SA e CPLEX obtiveram os resultados 12, 9 e 23, respectivamente. Na Figura 5.4, está o gráfico de médias resultantes do teste HSD de Tukey com nível de confiança de 95%. Ao analisá-la pode-se perceber que os algoritmos ILS-VND e ILS-RVND têm desempenho significativamente melhor que os outros dois algoritmos TS e SA, uma vez que não há sobreposição dos intervalos entre estes eles. Além disso, percebe-se que não há diferenças significativas entre os algoritmos CPLEX, ILS-VND e ILS-RVND, já que estes intervalos se interceptam.

Tabela 5.7: Comparação entre as soluções obtidas pelo CPLEX e pelos algoritmos propostos para o WCVRPTW, considerando as instâncias de pequeno porte.

Instância	CPLEX			ILS-VND				ILS-RVND				TS				SA			
($\|V^c\|$,$\|V^f\|$)	nv	Best	Time (s)	nv	Best	Avg	Time (s)	nv	Best	Avg	Time	nv	Best	Avg	Time (s)	nv	Best	Avg	Time (s)
p1: (10, 1)	1	**80.67**	0.23	1	**80.67**	81.15	0.00	1	**80.67**	84.23	0.00	1	86.66	90.45	0.00	1	85.32	92.11	0.05
p2: (10, 1)	1	**88.66**	0.35	1	**88.66**	88.66	0.00	1	**88.66**	88.66	0.00	1	94.95	94.95	0.00	1	94.00	95.30	0.08
p3: (10, 1)	1	**46.05**	3600.00	1	**46.05**	46.05	0.00	1	**46.05**	46.17	0.00	1	47.18	47.18	0.00	1	**45.90**	47.39	0.08
p4: (10, 1)	1	**46.79**	212.27	1	**46.79**	46.79	0.00	1	**46.79**	46.79	0.00	1	**46.79**	46.79	0.00	1	**46.79**	58.39	0.09
p5: (10, 1)	1	**20.91**	0.30	1	**20.91**	20.91	0.00	1	**20.91**	21.16	0.00	1	21.77	21.77	0.00	1	22.10	25.15	0.13
p6: (10, 1)	1	**4.54**	1887.37	1	**4.54**	4.54	0.00	1	**4.54**	4.54	0.00	1	**4.54**	4.56	0.00	1	**4.54**	5.60	0.10
p7: (10, 1)	1	**4.52**	1304.55	1	**4.52**	4.52	0.00	1	**4.52**	4.52	0.00	1	**4.52**	4.52	0.00	1	**4.52**	6.55	0.09
p8: (10, 1)	1	**83.84**	0.12	1	**83.84**	83.84	0.00	1	**83.84**	83.84	0.00	1	**83.84**	83.84	0.00	1	**83.84**	98.12	0.11
p9: (10, 1)	1	**89.34**	0.29	1	**89.34**	89.34	0.00	1	**89.34**	89.90	0.00	1	92.53	92.53	0.00	1	92.53	106.48	0.08
p10: (10, 1)	1	**95.84**	0.32	1	**95.84**	95.84	0.00	1	**95.84**	97.31	0.00	1	**95.84**	98.79	0.00	1	98.79	99.97	0.08
p11: (12, 1)	1	**83.96**	10.11	1	97.01	98.23	0.00	1	99.61	99.61	0.00	1	101.51	102.23	0.00	1	102.05	102.05	0.06
p12: (12, 1)	1	**86.84**	13.86	1	**86.84**	89.43	0.00	1	**86.84**	86.84	0.03	1	87.96	87.96	0.00	1	88.53	90.21	0.09
p13: (12, 1)	1	84.13	3600.00	1	**52.87**	52.87	0.00	1	**52.87**	52.87	0.00	1	54.43	56.16	0.00	1	54.43	67.06	0.11
p14: (12, 1)	1	**77.97**	123.69	1	**77.97**	77.97	0.00	1	**77.97**	78.48	0.00	1	78.50	78.50	0.00	1	78.50	93.50	0.09
p15: (12, 1)	1	**77.98**	123.97	1	**77.98**	77.97	0.00	1	**77.98**	78.29	0.00	1	**77.98**	78.00	0.00	1	78.50	141.80	0.11
p16: (12, 1)	1	**80.01**	2.7	1	**80.01**	80.01	0.00	1	**80.01**	80.01	0.00	1	**80.01**	80.01	0.00	1	**80.01**	135.80	0.09
p17: (12, 1)	1	**50.82**	1.74	1	**50.82**	50.82	0.00	1	**50.82**	51.32	0.00	1	51.37	51.37	0.00	1	51.37	70.20	0.09
p18: (12, 1)	1	**44.11**	3600.00	1	**44.11**	44.11	0.00	1	**44.11**	44.70	0.00	1	44.93	44.93	0.00	1	45.17	53.00	0.09
p19: (12, 1)	1	**42.93**	3600.00	1	**42.93**	42.93	0.00	1	**42.93**	42.93	0.00	1	**42.93**	42.93	0.00	1	43.27	47.16	0.11
p20: (12, 1)	1	**40.60**	3600.00	1	**40.60**	40.60	0.00	1	**40.60**	40.94	0.00	1	40.72	40.72	0.00	1	40.72	50.72	0.08
p21: (14, 1)	1	**78.67**	0.71	1	**78.67**	78.67	0.00	1	78.74	79.15	0.00	1	79.19	79.19	0.00	1	79.19	136.46	0.13
p22: (14, 1)	1	**78.68**	0.69	1	**78.68**	78.68	0.00	1	**78.68**	79.19	0.00	1	78.90	78.90	0.00	1	79.19	137.23	0.13
p23: (14, 1)	1	**156.92**	1.31	1	**156.92**	165.86	0.00	1	165.86	165.86	0.00	1	162.86	162.86	0.00	1	165.86	167.98	0.14
p24: (14, 1)	1	**161.15**	2.3	1	**161.15**	161.25	0.00	1	**161.15**	169.86	0.00	1	172.15	172.15	0.00	1	172.15	287.25	0.11
p25: (14, 1)	1	**117.25**	0.62	1	**117.25**	117.67	0.00	1	121.65	124.44	0.00	1	124.64	124.64	0.00	1	124.64	163.49	0.11
p26: (14, 1)	1	**118.80**	0.21	1	**118.80**	118.80	0.00	1	**118.80**	120.28	0.00	1	**118.80**	118.80	0.00	1	121.76	158.10	0.12
p27: (14, 1)	1	**109.19**	0.58	1	**109.19**	114.19	0.00	1	**109.19**	109.19	0.00	1	114.19	114.19	0.00	1	114.19	180.15	0.12
p28: (14, 1)	1	119.29	3600.00	1	**114.07**	114.07	0.00	1	115.68	117.34	0.00	1	119.53	119.53	0.00	1	119.53	177.36	0.12
p29: (14, 1)	1	**97.98**	0.66	1	**97.98**	97.98	0.00	1	**97.98**	97.98	0.00	1	**97.98**	97.98	0.00	1	**97.98**	112.80	0.10
p30: (14, 1)	1	**153.49**	0.43	1	**153.49**	153.49	0.00	1	**153.49**	153.49	0.00	1	**153.49**	153.49	0.00	1	**153.49**	229.29	0.12

Tabela 5.8: (continuação) Comparação entre as soluções obtidas pelo CPLEX e pelos algoritmos propostos para o WCVRPTW, considerando as instâncias de pequeno porte.

Instância	CPLEX			ILS-VND				ILS-RVND				TS				SA			
($\|V^c\|$,$\|V^f\|$)	nv	Best	Time (s)	nv	Best	Avg	Time (s)	nv	Best	Avg	Time	nv	Best	Avg	Time (s)	nv	Best	Avg	Time (s)
p31: (16, 1)	1	100.73	3600.00	1	**99.86**	99.95	0.08	1	104.54	106.14	0.00	1	104.45	127.26	0.00	1	105.14	109.11	0.15
p32: (16, 1)	1	53.60	3600.00	1	55.54	55.54	0.08	1	**53.60**	64.31	0.00	1	55.54	55.54	0.00	1	65.28	95.70	0.17
p33: (16, 1)	1	82.69	3600.00	1	86.98	87.0	0.07	1	**78.74**	83.72	0.00	1	80.07	109.83	0.00	1	83.04	96.92	0.10
p34: (16, 1)	1	**50.03**	37.87	1	51.71	51.71	0.00	1	51.71	51.71	0.00	1	51.71	51.71	0.00	1	51.71	54.64	0.09
p35: (16, 1)	1	60.69	3600.00	1	**58.74**	58.94	0.07	1	60.78	63.47	0.00	1	63.66	63.66	0.00	1	64.63	71.55	0.16
p36: (16, 1)	1	**66.79**	5.14	1	67.39	68.54	0.07	1	68.37	71.43	0.00	1	69.15	70.24	0.00	1	69.65	72.85	0.19
p37: (16, 1)	1	**64.20**	63.31	1	**64.20**	64.20	0.09	1	66.69	67.57	0.00	1	68.21	68.21	0.00	1	68.21	70.76	0.18
p38: (16, 1)	1	**50.63**	23.94	1	**50.63**	50.63	0.08	1	**50.63**	53.69	0.00	1	52.74	52.74	0.00	1	53.80	54.28	0.15
p39: (16, 1)	1	**51.86**	6.96	1	59.07	59.07	0.07	1	59.07	62.55	0.00	1	61.21	61.21	0.00	1	61.21	64.05	0.10
p40: (16, 1)	1	**60.26**	2.40	1	**60.26**	60.26	0.08	1	**60.26**	65.72	0.00	1	65.72	79.59	0.00	1	65.72	108.12	0.19
p41: (18, 2)	1	**81.00**	3600.00	1	82.65	82.65	0.07	1	82.65	83.67	0.00	1	83.48	0.14	0.00	1	83.48	91.28	0.17
p42: (18, 2)	1	**75.55**	3600.00	1	75.55	75.55	0.08	1	75.55	82.35	0.00	1	80.44	90.92	0.00	1	87.63	92.33	0.14
p43: (18, 2)	1	**81.20**	3600.00	1	86.78	86.78	0.06	1	86.78	86.78	0.00	1	86.78	90.32	0.00	1	86.78	94.24	0.19
p44: (18, 2)	1	**77.33**	3600.00	1	79.21	79.21	0.03	1	79.21	79.21	0.00	1	79.21	82.11	0.00	1	79.21	88.24	0.19
p45: (18, 2)	1	**60.69**	275.65	1	**60.69**	60.69	0.03	1	**60.69**	65.22	0.00	1	**60.69**	70.12	0.00	1	**60.69**	75.14	0.14
p46: (18, 2)	1	74.67	3600.00	1	**62.32**	62.32	0.08	1	**62.32**	65.06	0.01	1	66.14	74.14	0.00	1	66.33	76.24	0.16
p47: (18, 2)	1	62.09	3600.00	1	**32.79**	32.79	0.07	1	**32.79**	34.20	0.00	1	34.61	34.61	0.00	1	37.20	56.30	0.16
p48: (18, 2)	1	**47.23**	3600.00	1	**47.23**	43.23	0.07	1	53.69	59.47	0.00	1	63.23	63.23	0.00	1	64.46	69.78	0.14
p49: (18, 2)	1	70.95	3600.00	1	70.95	70.95	0.08	1	**70.84**	77.43	0.00	1	82.07	82.07	0.00	1	82.94	96.61	0.14
p50: (18, 2)	1	**65.27**	3600.00	1	67.40	67.40	0.08	1	73.81	74.98	0.00	1	76.86	88.58	0.00	1	77.43	90.68	0.17
p51: (20, 2)	1	**64.12**	3600.00	1	69.11	69.11	0.10	1	69.11	70.67	0.00	1	69.11	69.11	0.00	1	70.84	82.53	0.16
p52: (20, 2)	1	**67.28**	3600.00	1	69.99	70.27	0.10	1	**67.28**	69.99	0.00	1	69.99	74.23	0.00	1	69.99	76.71	0.17
p53: (20, 2)	1	**53.53**	3600.00	1	57.31	57.31	0.11	1	57.31	57.70	0.00	1	61.17	61.17	0.00	1	61.17	75.47	0.20
p54: (20, 2)	1	**38.28**	3600.00	1	41.95	41.95	0.09	1	41.95	44.10	0.00	1	44.61	44.61	0.00	1	51.17	55.47	0.19
p55: (20, 2)	1	54.30	3600.00	1	**26.44**	26.44	0.10	1	27.33	28.35	0.00	1	28.43	35.45	0.00	1	27.64	28.23	0.10
p56: (20, 2)	1	**42.59**	3600.00	1	53.48	53.48	0.10	1	53.48	54.65	0.00	1	63.48	64.21	0.00	1	64.58	67.79	0.18
p57: (20, 2)	1	**55.67**	3600.00	1	58.12	58.12	0.09	1	56.22	59.88	0.00	1	64.48	64.48	0.00	1	59.36	59.64	0.20
p58: (20, 2)	1	**63.39**	3600.00	1	**63.39**	64.23	0.09	1	63.51	64.70	0.00	1	65.82	65.82	0.00	1	63.43	64.72	0.19
p59: (20, 2)	1	**57.70**	3600.00	1	**57.70**	57.70	0.10	1	58.32	61.56	0.00	1	58.43	62.10	0.00	1	58.43	64.66	0.16
p60: (20, 2)	1	**70.81**	3600.00	1	**70.81**	70.81	0.11	1	**70.81**	70.86	0.00	1	74.02	80.64	0.00	1	75.21	79.56	0.21

Tabela 5.9: Médias dos RPDs (por grupo de instâncias de 10 a 20 clientes) dos algoritmos comparados para o melhor resultado encontrado.

Clientes	CPLEX	ILS-VND	ILS-RVND	TS	SA
10	**0.0**	**0.0**	**0.0**	3.1	3.0
12	2.7	**0.0**	0.3	1.4	1.7
14	0.4	**0.0**	1.2	2.9	3.5
16	**0.0**	2.0	2.0	4.8	7.3
18	4.5	**0.0**	1.9	7.2	9.1
20	0.4	**0.0**	**0.0**	6.0	6.4
Média	1.3	**0.3**	0.9	4.2	5.1

5.5.2 Resultados para instâncias de médio porte

Nesta seção, é feita a comparação do desempenho dos algoritmos apresentados neste trabalho: ILS-VND, ILS-RVND, TS e SA, para as instâncias do grupo de médio porte (instâncias geradas). Todos os algoritmos foram executados 10 vezes. Os resultados obtidos são analisados utilizando o valor do RPD (Equação 5.1), o qual é obtido através do melhor valor da função objetivo nas 10 execuções.

A Tabela 5.10 apresenta os valores dos RPDs calculados a partir dos melhores resultados (nas 10 execuções) encontrados pelos algoritmos, para as instâncias de médio porte. Os valores dos RPDs são agrupados pelo número de clientes atendidos. Os valores em negrito representam os melhores resultados obtidos. Claramente, é possível perceber que o algoritmo ILS-RVND encontrou os melhores resultados em todos os grupos de instâncias.

Tabela 5.10: Valores médios de RPDs (em 10 execuções) por grupo de instâncias de 50 a 200 clientes

Clientes	ILS-VND	ILS-RVND	TS	SA
50	0.02	**0.00**	8.82	10.84
100	0.03	**0.00**	9.38	12.47
150	0.06	**0.00**	8.86	10.92
200	0.01	**0.00**	6.67	7.69
Média	0.03	**0.00**	8.43	10.48

Na Figura 5.5, está o gráfico de médias resultantes do teste HSD de Tukey com nível de confiança de 95%. Nota-se a partir desta figura que os algoritmos ILS-

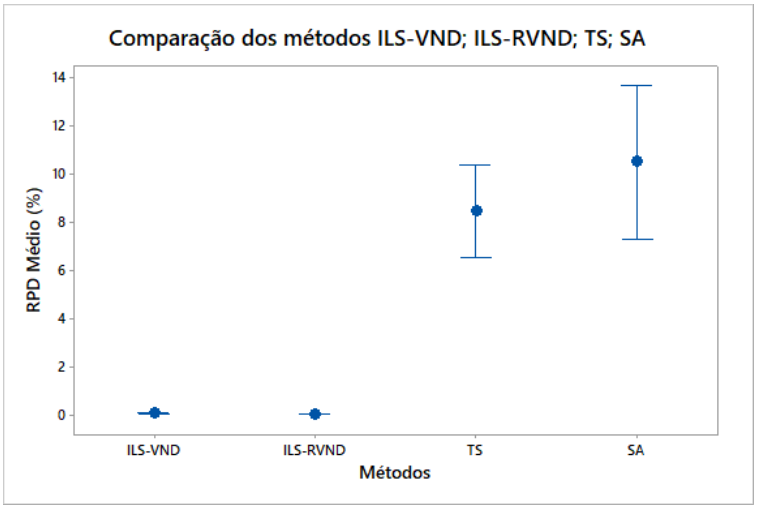

Figura 5.5: Comparação de algoritmos para instâncias de médio porte

VND e ILS-RVND tem desempenho significativamente melhor que os dois outros algoritmos TS e SA, uma vez que não há sobreposição dos intervalos entre estes algoritmos. Além disso, percebe-se que não há diferenças significativas entre os algoritmos ILS-VND e ILS-RVND, já que estes gráficos se interceptam.

5.6 Avaliação das heurísticas em instâncias da literatura

Utilizamos também o grupo de instâncias da literatura para comparação dos algoritmos heurísticos propostos. Estas instâncias foram criadas por Kim et al. [2006] para o problema WCVRPTW. Neste conjunto existem 10 instâncias, onde o número de clientes varia de 99 a 2092 e o número de aterros sanitários varia de 1 a 7.

A Tabela 5.11 apresenta o valor da função objetivo final da execução dos algoritmos propostos, para as instâncias da literatura (PL01 até PL10), em comparação aos algoritmos CBA [Kim et al., 2006], VNTS [Benjamin & Beasley, 2013] e TS [Benjamin & Beasley, 2010], propostos na literatura. As duas primeiras colunas da tabela apresentam as características das instâncias (nome da instância e número de clientes, aterros sanitários), seguidas das colunas *nv*, *Best* que informam, respectivamente, o número de veículos utilizados para solução do problema e o valor da melhor solução obtida pelo algoritmo. Por fim, a última linha apresenta a média dos resultados obtidos. É importante apontar que os melhores resultados foram destacados em negrito com o intuito de evidenciar a qualidade dos resultados alcançados por cada

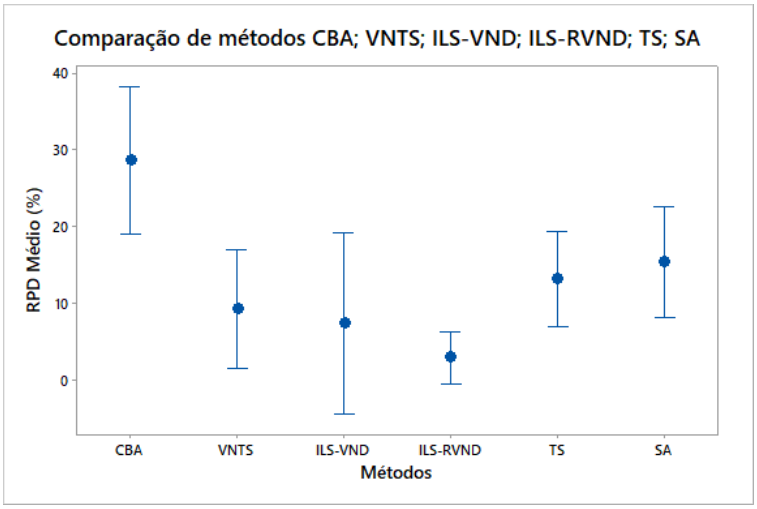

Figura 5.6: Comparação dos algoritmos para instâncias da literatura

algoritmo. Observando a solução média apresentada na última linha da Tabela 5.11, é possível verificar que, entre as heurísticas apresentadas, o ILS-RVND apresentou o menor valor, 777.6. Já a melhor solução da literatura tomada como referência apresentou o valor 841.0, portanto houve um melhoramento de cerca de 7.5% na qualidade das soluções, graças à estratégia proposta na heurística ILS-RVND.

Comparando a heurística TS proposta por Benjamin & Beasley [2010] e sua reimplementação. É possível verificar na última linha da Tabela 5.11, que a solução média apresentada pela heurística re-implementada apresentou um menor valor, 887.9. Já, a heurística TS* da literatura apresentou o valor 928.5, portanto houve uma melhora de 4.37% na qualidade das soluções.

Para validar os resultados obtidos e verificar se as diferenças são estatisticamente significativas, foi realizada a ANOVA paramétrica. A Figura 5.6 apresenta o gráfico de médias resultante do teste HSD de Tukey com nível de confiança de 95%, o qual compara os algoritmos da literatura CBA, VNTS, TS e os algoritmos propostos: ILS-VND, ILS-RVND, TS e SA. Nota-se, a partir desta figura, que o algoritmo ILS-RVND tem desempenho significativamente melhor que os três outros algoritmos (CBA, TS e SA), uma vez que não há sobreposição dos intervalos entre estes algoritmos. Além disso, percebe-se que não há diferenças significativas entre os algoritmos VNTS, ILS-VND e ILS-RVND, já que estes gráficos se interceptam.

Cada instância foi executada 10 vezes para cada algoritmo proposto. Os tempos computacionais encontram-se na Tabela 5.12. Observando o tempo médio apresentado na última linha da tabela, é possível verificar que, entre as abordagens apresentadas o ILS-VND apresentou o menor tempo, 4.7 segundos. Já o algoritmo

VNTS da literatura, demandou um tempo maior, cerca de 2143.3 segundos. As heurísticas ILS-RVND, TS e SA, registraram, em média 5.4, 8.9 e 15.4 segundos, respectivamente. Comparadas com as demais heurísticas da literatura, estas demandaram baixo custo computacional. Diante dessas análises, a proposta ILS-RVND apresentou-se com melhor desempenho, pois consegue encontrar soluções boas com baixo tempo de execução se comparadas aos outros algoritmos.

Tabela 5.11: Comparação entre as soluções obtidas pelas heurísticas, considerando as instâncias da literatura.

Instância	CBA		VNTS		TS*		TS		ILS-VND		ILS-RVND		SA	
($\|V^c\|$,$\|V^f\|$)	nv	Best	nv	Best	nv	Best	nv	Best	nv	Best	nv	Best	nv	Best
PL01: (99, 2)	3	205.1	3	**156.9**	3	183.5	3	176.0	3	240.5	3	174.5	3	176.6
PL02: (275, 1)	3	527.3	3	454.7	3	464.5	2	389.1	2	394.0	2	**380.4**	2	391.9
PL03: (330, 4)	6	205.0	6	**186.7**	6	204.6	6	202.4	6	194.8	6	201.5	6	204.9
PL04: (442, 1)	11	87.0	11	79.7	11	89.1	10	74.2	10	72.8	10	**72.0**	10	74.9
PL05: (784, 19)	5	769.5	5	641.8	5	756.3	4	728.6	4	**634.5**	4	**634.5**	4	745.6
PL06: (1048, 2)	18	2370.4	17	2123.8	17	2250.5	17	2137.7	17	1876.8	17	**1872.4**	17	2150.7
PL07: (1347, 3)	7	1039.7	8	874.7	8	915.1	8	850.9	8	656.8	8	**654.1**	8	871.9
PL08: (1596, 2)	13	1459.2	13	**1206.1**	14	1410.4	13	1339.5	12	1338.3	12	1301.3	13	1385.3
PL09: (1927, 4)	17	1395.3	16	1127.7	16	1262.8	16	1162.5	16	1037.5	16	**1032.3**	16	1180.9
PL10: (2092, 7)	16	1833.8	16	1558.1	16	1749.0	16	1818.9	16	1461.9	16	**1453.7**	16	1816.8
Média		989.2		841.0		928.5		887.9		790.7		**777.6**		899.9

Tabela 5.12: Comparação entre os tempos das soluções obtidas pelas heurísticas, considerando as instâncias da literatura.

Instância	CBA	VNTS	TS*	TS	ILS-VND	ILS-RVND	SA
($\|V^c\|$,$\|V^f\|$)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time(s)	Time(s)
PL01: (99, 2)	3.0	16.0	4.0	0.3	**0.0**	0.5	0.8
PL02: (275, 1)	10.0	272.0	13.0	0.3	**0.04**	0.6	0.6
PL03: (330, 4)	11.0	219.0	16.0	0.6	**0.04**	0.6	0.3
PL04: (442, 1)	16.0	273.0	28.0	1.6	**0.12**	1.6	1.8
PL05: (784, 19)	92.0	2152.0	73.0	3.2	**0.96**	2.7	9.5
PL06: (1048, 2)	329.0	637.0	116.0	10.2	**2.0**	5.9	12.3
PL07: (1347, 3)	95.0	2698.0	162.0	12.1	**3.8**	4.6	18.9
PL08: (1596, 2)	212.0	2556.0	223.0	13.7	5.2	**4.8**	26.1
PL09: (1927, 4)	424.0	6541.0	346.0	19.8	18.4	**17.3**	38.4
PL10: (2092, 7)	408.0	6069.0	332.0	27.3	17.2	**16.3**	46.0
Média	160.0	2143.3	131.3	8.9	**4.7**	5.4	15.4

Capítulo 6

Conclusões

Neste trabalho foram implementados quatro algoritmos heurísticos para a resolução do Problema de Roteamento de Veículos para Coleta de lixo Comercial com Janelas de Tempo. O primeiro algoritmo é baseado nas metaheurísticas *Iterated Local Search* (ILS) [Lourenço et al., 2003] e *Variable Neighborhood Descent* (VND) [Mladenović & Hansen, 1997], denotado como ILS-VND. O segundo algoritmo é similar ao ILS-VND, a diferença entre eles está no procedimento de busca local, que utiliza uma ordenação aleatória das vizinhanças (RVND), nomeado ILS-RVND. O terceiro algoritmo é a reimplementação da metaheurística *Tabu Search*, proposta por Benjamin & Beasley [2010] e o último algoritmo é baseado na metaheurística *Simulated Annealing* [Kirkpatrick et al., 1987].

Os algoritmos propostos foram testados em 10 instâncias, encontradas na literatura [Kim et al., 2006], e em conjunto de 160 instâncias de pequeno e médio porte, propostas neste trabalho. Os resultados obtidos foram validados através de testes estatísticos.

Estes resultados indicam claramente que os algoritmos propostos são altamente eficientes quando comparados com o algoritmo CBA, proposto por Kim et al. [2006]. Em relação aos resultados obtidos pelo VNTS, melhor algoritmo do *estado da arte*, o algoritmo ILS-RNVD obteve melhores resultados em 7 das 10 instâncias da literatura, seguido do algoritmo ILS-VND. Porém, os algoritmos SA e TS não obtiveram o mesmo desempenho.

Em relação a reimplementação do algoritmo TS*, proposto por Benjamin & Beasley [2010], o algoritmo reimplementado obteve melhores resultados que o algoritmo TS* da literatura.

Sendo assim, é possível concluir que as propostas apresentadas neste trabalho são consistentes e promissoras, visto que além dos bons resultados alcançados, os

algoritmos desenvolvidos possuem uma estrutura simples e que basicamente necessita de poucos parâmetros de calibração. Outro aspecto positivo dos algoritmos são o tempo de execução, que são extremamente curtos, o que é ideal para aplicações reais que requerem resposta rápida às mudanças, facilitando a tomada de decisão pelos gestores.

Como trabalhos futuros, propõem-se a criação de um novo algoritmo construtivo e a implementação de um algoritmo evolucionário, visto que para este problema poucos autores propuseram mataheurísticas populacionais. Este algoritmo buscaria aprimorar os resultados encontrados. Além disso, pretende-se estender os algoritmos propostos para tratar outras variantes do WCVRPTW.

6.1 Publicação

Campos, Alba A., and José Elias C. Arroyo. "An ILS Heuristic for the Waste Collection Vehicle Routing Problem with Time Windows. *International Conference on Intelligent Systems Design and Applications*. Springer, Cham, 2016.

Apêndice A

Experimentos ILS-VND

Neste apêndice são apresentados os resultados obtidos por meio dos experimentos de calibração da abordagem ILS-VND. Tais resultados indicam quais as melhores configurações de parâmetros que permitem o melhor desempenho da abordagem. Os resultados obtidos pela calibração foram validados estatisticamente na Seção 5.4.

Os experimentos foram realizados utilizando uma amostra composta por 20% do conjunto total de instâncias. O algoritmo foi testado com 6 combinações diferentes de parâmetros, resultando em 18 abordagens diferentes. Os valores testados foram: $V \in \{N^4, N^5, N^6\}$, $MaxIterILS \in \{10, 20, 50\}$ e $Tam \in \{5, 10, 25, 50\}$. A Tabela A.1 apresenta a descrição dos rótulos utilizados nas tabelas de resultados.

Tabela A.1: Descrição dos identificadores utilizados nas tabelas de resultados relacionadas ao ILS-VND.

Identificador	**Descrição**
Id	Identificador da instância
$Best$	Melhor solução em 10 execuções.
Avg	Solução média em 10 execuções.
$Time(s)$	Tempo médio em segundos(s) de 10 execuções

Devido à quantidade de dados, os resultados da abordagem não puderam ser apresentados na mesma tabela. Estes podem ser vistos nas Tabelas A.2, A.3, A.4, A.5, A.6, A.7.

Analisando as médias dos RPD's apresentados na Tabela A.9, a abordagem pode ser classificada da melhor para a pior nesta ordem: ILS-VND 18(2,64%), ILS-VND 6(2,66%), ILS-VND 12(3,13%), ILS-VND 3(3,48%),ILS-VND 9(3,65%), ILS-VND 15(3,80%), ILS-VND 17(3,84%), ILS-VND 5(4,07%), ILS-VND 11(4,15%), ILS-VND 2(4,22%), ILS-VND 8(4,33%),ILS-VND 14(4,89%), ILS-VND 16(4,26%),

ILS-VND 4(5,51%), ILS-VND 1(5,52%), ILS-VND 3(3,48%), ILS-VND 13(5,82%), ILS-VND 7(5,96%). Como pode ser observado nesta classificação, a configuração de parâmetros que obteve o melhor valor médio RPD foi o ILS-VND 18. Sendo assim a abordagem ILS-VND utilizou como parâmetros as vizinhanças N^5, N^4, N^6, com 50 iterações e tamanhos de vizinhanças {5, 10, 25, 50}.

Tabela A.2: Resultados referentes às soluções do Algoritmo ILS-VND - combinação testada N^4 N^5 N^6.

Id	10 iterações			20 iterações			50 iterações		
	Best	*Avg*	*Time(s)*	*Best*	*Avg*	*Time(s)*	*Best*	*Avg*	*Time(s)*
1	88.69	88.83	0.00	**88.66**	88.87	0.00	**88.66**	88.87	0.00
2	**46.05**	46.17	0.00	**46.05**	46.39	0.00	**46.05**	46.06	0.00
3	**46.79**	46.79	0.00	**46.79**	46.79	0.00	**46.79**	46.79	0.00
4	20.96	20.99	0.00	21.22	21.22	0.00	**20.91**	20.94	0.00
5	**4.54**	4.54	0.00	**4.54**	4.54	0.00	**4.54**	4.54	0.00
6	**4.52**	4.52	0.00	**4.52**	4.52	0.00	**4.52**	4.52	0.00
7	**83.84**	83.84	0.00	**83.84**	83.84	0.00	**83.84**	83.84	0.00
8	91.30	91.30	0.00	89.60	89.90	0.00	**89.30**	89.70	0.00
9	98.70	98.70	0.00	96.70	97.70	0.00	**95.80**	96.70	0.00
10	104.70	104.80	0.00	102.30	102.70	0.00	**101.00**	101.10	0.00
11	56.10	59.00	0.00	54.40	60.8	0.00	**53.90**	54.10	0.00
12	394.03	394.03	0.40	393.70	393.80	0.64	**391.10**	391.27	1.30
13	202.60	204.60	0.40	201.60	203.80	0.70	**193.05**	194.40	1.40
14	75.11	75.78	0.60	68.99	72.32	2.10	**68.96**	69.23	3.66

Tabela A.3: Resultados referentes às soluções do Algoritmo ILS-VND - combinação testada N^4 N^6 N^5.

Id	10 iterações			20 iterações			50 iterações		
	Best	*Avg*	*Time(s)*	*Best*	*Avg*	*Time(s)*	*Best*	*Avg*	*Time(s)*
1	88.67	88.92	0.00	88.67	88.77	0.00	**88.66**	88.71	0.00
2	46.05	46.17	0.00	46.09	46.18	0.00	**46.05**	46.06	0.00
3	**46.79**	46.79	0.00	**46.79**	46.79	0.00	**46.79**	46.79	0.00
4	21.22	21.64	0.00	21.19	20.98	0.00	**20.91**	20.94	0.00
5	**4.54**	4.54	0.00	**4.54**	4.54	0.00	**4.54**	4.54	0.00
6	**4.52**	4.52	0.00	**4.52**	4.52	0.00	**4.52**	4.52	0.00
7	**83.84**	83.84	0.00	**83.84**	83.84	0.00	**83.84**	83.84	0.00
8	90.16	92.04	0.00	89.69	89.84	0.00	**89.34**	89.87	0.00
9	98.71	98.77	0.00	98.50	98.97	0.00	**95.84**	97.49	0.00
10	104.80	105.01	0.00	102.96	103.24	0.00	**100.95**	102.87	0.00
11	61.98	62.17	0.00	54.49	59.70	0.00	**52.87**	52.92	0.00
12	392.12	392.31	0.32	388.68	390.78	0.64	**366.96**	389.18	1.31
13	200.27	203.83	0.48	196.47	202.77	0.75	**194.98**	200.65	1.49
14	68.81	72.14	0.60	68.69	72.08	1.94	**67.70**	71.48	3.51

Tabela A.4: Resultados referentes às soluções do Algoritmo ILS-VND - combinação testada N^6 N^5 N^4.

Id	10 iterações			20 iterações			50 iterações		
	Best	*Avg*	*Time(s)*	*Best*	*Avg*	*Time(s)*	*Best*	*Avg*	*Time(s)*
1	**88.92**	88.92	0.00	**88.92**	88.92	0.00	**88.92**	88.92	0.00
2	46.89	46.27	0.00	46.10	46.39	0.00	**46.05**	46.06	0.00
3	46.73	47.18	0.00	46.79	46.85	0.00	**46.05**	46.36	0.00
4	21.16	21.22	0.00	21.09	21.89	0.00	**20.91**	21.45	0.00
5	**4.54**	4.54	0.00	**4.54**	4.54	0.00	**4.54**	4.54	0.00
6	**4.52**	4.52	0.00	**4.52**	4.52	0.00	**4.52**	4.52	0.00
7	**83.84**	83.84	0.00	**83.84**	83.84	0.00	**83.84**	83.84	0.00
8	90.16	91.71	0.00	89.85	89.89	0.00	**89.34**	89.51	0.00
9	96.76	98.79	0.00	96.76	97.89	0.00	**95.84**	96.44	0.00
10	104.80	105.03	0.00	101.01	105.73	0.00	**100.15**	100.88	0.00
11	56.10	59.00	0.00	54.40	60.8	0.00	**53.90**	54.10	0.00
12	388.36	388.52	0.37	392.23	392.40	0.61	**393.71**	393.73	1.31
13	201.59	202.81	0.55	201.59	202.81	0.55	**199.3**	202.11	1.80
14	73.05	74.70	1.07	68.98	72.21	1.90	**67.79**	70.42	3.75

Tabela A.5: Resultados referentes às soluções do Algoritmo ILS-VND - combinação testada N^6 N^4 N^5.

Id	10 iterações			20 iterações			50 iterações		
	Best	*Avg*	*Time(s)*	*Best*	*Avg*	*Time(s)*	*Best*	*Avg*	*Time(s)*
1	**88.92**	88.92	0.00	**88.92**	88.92	0.00	**88.92**	88.92	0.00
2	**46.05**	46.17	0.00	**46.05**	46.05	0.00	**46.05**	46.0	0.00
3	47.18	47.39	0.00	46.73	47.18	0.00	**46.05**	46.36	0.00
4	21.19	21.22	0.00	20.96	21.22	0.00	**20.91**	21.00	0.00
5	**4.54**	4.54	0.00	**4.54**	4.54	0.00	**4.54**	4.54	0.00
6	**4.52**	4.52	0.00	**4.52**	4.52	0.00	**4.52**	4.52	0.00
7	**83.84**	83.84	0.00	**83.84**	83.84	0.00	**83.84**	83.84	0.00
8	89.69	91.37	0.00	89.69	89.98	0.00	**89.34**	89.79	0.00
9	98.79	99.97	0.00	98.71	98.79	0.00	**96.76**	98.56	0.00
10	101.08	106.17	0.00	100.95	101.03	0.00	**100.15**	100.34	0.00
11	62.40	79.46	0.00	53.95	73.76	0.00	**52.87**	56.11	0.00
12	385.22	388.88	0.35	392.42	394.01	0.54	**387.02**	391.20	1.14
13	201.56	202.93	0.51	199.62	200.4	0.83	**198.5**	200.91	1.87
14	72.71	72.97	1.26	69.97	70.20	2.02	**66.83**	70.19	3.45

Tabela A.6: Resultados referentes às soluções do Algoritmo ILS-VND - combinação testada N^5 N^6 N^4.

Id	10 iterações			20 iterações			50 iterações		
	Best	*Avg*	*Time(s)*	*Best*	*Avg*	*Time(s)*	*Best*	*Avg*	*Time(s)*
1	88.69	88.83	0.00	**88.67**	88.87	0.00	**88.67**	88.92	0.00
2	**46.05**	46.27	0.00	**46.05**	46.68	0.00	**46.05**	46.06	0.00
3	46.73	47.18	0.00	46.79	46.85	0.00	**46.05**	46.36	0.00
4	**20.91**	21.00	0.00	**20.91**	21.10	0.00	**20.91**	20.91	0.00
5	**4.54**	4.54	0.00	**4.54**	4.54	0.00	**4.54**	4.54	0.00
6	**4.52**	4.52	0.00	**4.52**	4.52	0.00	**4.52**	4.52	0.00
7	**83.84**	83.84	0.00	**83.84**	83.84	0.00	**83.84**	83.84	0.00
8	91.3	91.87	0.00	89.60	89.90	0.00	**89.30**	89.90	0.00
9	96.76	98.79	0.00	98.11	98.99	0.00	**95.84**	96.44	0.00
10	104.80	105.12	0.00	101.01	103.12	0.00	**100.15**	101.47	0.00
11	62.40	67.80	0.00	60.32	62.12	0.00	**56.11**	57.33	0.00
12	388.36	388.52	0.34	392.27	392.97	0.66	**386.79**	388.66	1.20
13	198.43	202.09	0.49	196.95	200.33	0.84	**195.20**	198.58	1.43
14	72.57	72.77	1.02	68.98	72.21	1.90	**67.79**	70.42	3.75

Tabela A.7: Resultados referentes às soluções do Algoritmo ILS-VND - combinação testada N^5 N^4 N^6.

Id	10 iterações			20 iterações			50 iterações		
	Best	*Avg*	*Time(s)*	*Best*	*Avg*	*Time(s)*	*Best*	*Avg*	*Time(s)*
1	**88.67**	88.83	0.00	**88.67**	88.92	0.00	**88.67**	88.83	0.00
2	**46.05**	46.27	0.00	**46.05**	46.39	0.00	**46.05**	46.06	0.00
3	46.73	46.79	0.00	46.79	46.85	0.00	**46.05**	46.36	0.00
4	**20.91**	21.00	0.00	**20.91**	21.10	0.00	**20.91**	20.91	0.00
5	**4.54**	4.54	0.00	**4.54**	4.54	0.00	**4.54**	4.54	0.00
6	**4.52**	4.52	0.00	**4.52**	4.52	0.00	**4.52**	4.52	0.00
7	**83.84**	83.84	0.00	**83.84**	83.84	0.00	**83.84**	83.84	0.00
8	91.30	91.30	0.00	89.60	89.90	0.00	**89.30**	89.70	0.00
9	96.78	98.79	0.00	96.76	97.59	0.00	**95.84**	96.31	0.00
10	101.08	106.12	0.00	100.95	101.30	0.00	**100.15**	100.34	0.00
11	61.98	62.17	0.00	59.49	59.86	0.00	**52.87**	54.33	0.00
12	392.12	392.31	0.34	388.68	390.78	0.66	**366.96**	371.66	1.20
13	201.56	202.09	0.49	199.62	200.33	0.84	**193.05**	195.58	1.43
14	70.05	72.77	1.02	69.01	72.21	1.90	**67.98**	70.22	3.75

Tabela A.8: Conjunto de valores testados na calibração do ILS-VND.

Parâmetros	Valores testados	Quantidade
V	N^4, N^5, N^6	6
$MaxIterILS$	10, 20, 50	3
Tam	5, 10, 25, 50	1
Total		18

Tabela A.9: Valores médios de RPD's por grupo de parâmetros testados.

Número	Combinação testada			
	V	$MaxIterILS$	Tam	*Médias RPD*
1	N^4, N^5, N^6	10	5,10,25,50	5,52
2	N^4, N^5, N^6	20	5,10,25,50	4,22
3	N^4, N^5, N^6	50	5,10,25,50	3,48
4	N^4, N^6, N^5	10	5,10,25,50	5,51
5	N^4, N^6, N^5	20	5,10,25,50	4,07
6	N^4, N^6, N^5	50	5,10,25,50	2,66
7	N^5, N^6, N^4	10	5,10,25,50	5,96
8	N^5, N^6, N^4	20	5,10,25,50	4,33
9	N^5, N^6, N^4	50	5,10,25,50	3,65
10	N^6, N^4, N^5	10	5,10,25,50	5,63
11	N^6, N^4, N^5	20	5,10,25,50	4,15
12	N^6, N^4, N^5	50	5,10,25,50	3,13
13	N^6, N^5, N^4	10	5,10,25,50	5,82
14	N^6, N^5, N^4	20	5,10,25,50	4,89
15	N^6, N^5, N^4	50	5,10,25,50	3,80
16	N^5, N^4, N^6	10	5,10,25,50	5,26
17	N^5, N^4, N^6	20	5,10,25,50	3,84
18	N^5, N^4, N^6	50	5,10,25,50	2,64

Referências Bibliográficas

Benjamin, A. & Beasley, J. (2013). Metaheuristics with disposal facility positioning for the waste collection vrp with time windows. *Optimization Letters*, 7(7):1433--1449.

Benjamin, A. M. & Beasley, J. (2010). Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. *Computers & Operations Research*, 37(12):2270--2280.

Berbeglia, G.; Cordeau, J.-F.; Gribkovskaia, I. & Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1--31.

Berbeglia, G.; Cordeau, J.-F. & Laporte, G. (2010). Dynamic pickup and delivery problems. *European journal of operational research*, 202(1):8--15.

Buhrkal, K.; Larsen, A. & Ropke, S. (2012). The waste collection vehicle routing problem with time windows in a city logistics context. *Procedia-Social and Behavioral Sciences*, 39:241--254.

Caric, T. & Gold, H. (2008). Vehicle routing problem.

Cunha, C. B. (2006). Contribuição à modelagem de problemas em logistica e transportes.

Dantzig, G. B. & Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80--91.

Desrochers, M.; Lenstra, J. K. & Savelsbergh, M. W. (1990). A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research*, 46(3):322--332.

Dumas, Y.; Desrosiers, J. & Soumis, F. (1991). The pickup and delivery problem with time windows. *European journal of operational research*, 54(1):7--22.

D'almeida, M. L. O.; Vilhena, A. et al. (2010). Lixo municipal: manual de gerenciamento integrado. *São Paulo: IPT/Cempre*, 2.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533--549.

Han, H. & Ponce Cueto, E. (2015). Waste collection vehicle routing problem: literature review. *PROMET-Traffic&Transportation*, 27(4):345--358.

IBAM (2001). - Manual Gerenciamento Integrado de Resíduos Sólidos. `http://www.resol.com.br/cartilha4/manual.pdf`. [Acesso em 10 de setembro de 2017].

IBGE (2010). Domicílios particulares ocupados, por situação e localização da área, segundo os municípios. `http://www.censo2010.ibge.gov.br/sinopse/index.php?dados=212&uf=41`. [Acesso em 15 de setembro de 2017].

Idrus, Z.; Ku-Mahamud, K. R. & Benjamin, A. M. (2017). Waste collection vehicle routing problem benchmark datasets and case studies: A review. *Journal of Theoretical and Applied Information Technology*, 95(5):1048.

Islam, R. & Rahman, M. S. (2012). An ant colony optimization algorithm for waste collection vehicle routing with time windows, driver rest period and multiple disposal facilities. In *Informatics, Electronics & Vision (ICIEV), 2012 International Conference on*, pp. 774--779. IEEE.

Kim, B.-I.; Kim, S. & Sahoo, S. (2006). Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12):3624--3642.

Kirkpatrick, S.; Gelatt, C. D. & Vecchi, M. P. (1987). Optimization by simulated annealing. In *Readings in Computer Vision*, pp. 606--615. Elsevier.

Lenstra, J. K. & Kan, A. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221--227.

Lourenço, H. R.; Martin, O. C. & Stutzle, T. (2003). Iterated local search. *International series in operations research and management science*, pp. 321--354.

Mladenović, N. & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, 24(11):1097--1100.

Ombuki-Berman, B. M.; Runka, A. & Hanshar, F. (2007). Waste collection vehicle routing problem with time windows using multi-objective genetic algorithms. In *Proceedings of the Third IASTED International Conference on Computational Intelligence*, pp. 91--97. ACTA Press.

Savelsbergh, M. W. & Sol, M. (1995). The general pickup and delivery problem. *Transportation science*, 29(1):17--29.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254--265.

Souza, M. J.; Coelho, I. M.; Ribas, S.; Santos, H. G. & Merschmann, L. H. d. C. (2010). A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research*, 207(2):1041--1051.

Subramanian, A.; Drummond, L. M. d. A.; Bentes, C.; Ochi, L. S. & Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899--1911.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons.