UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS VII - GOVERNADOR ANTÔNIO MARIZ
CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS
CURSO DE BACHARELADO EM CIÊNCIAS COMPUTAÇÃO

MIQUÉIAS GARCIA DE LUCENA

IMPLEMENTAÇÃO E ANÁLISE DE META-HEURÍSTICAS HIBRIDAS PARA O VRPTW E APLICAÇÕES EM PROBLEMAS LOGÍSTICOS

PATOS
2025

MIQUÉIAS GARCIA DE LUCENA

IMPLEMENTAÇÃO E ANÁLISE DE META-HEURÍSTICAS HIBRIDAS PARA O VRPTW E APLICAÇÕES EM PROBLEMAS LOGÍSTICOS

Trabalho de Conclusão do Curso II apresentado à coordenação do curso de Bacharelado em Computação do Centro de Ciências Exatas e Sociais Aplicadas da Universidade Estadual da Paraíba, em cumprimento às exigências legais para a obtenção do título de Graduado no Curso de Bacharelado em Ciências da Computação.

Área de concentração: Inteligência Artificial

Orientadora: Profª. Drª. Jannayna Domingues Barros Filgueira

PATOS
2025

L935i Lucena, Miquéias Garcia de.
Implementação e análise de meta-heurísticas híbridas para o VRPTW e aplicações em problemas logísticos [manuscrito] / Miquéias Garcia de Lucena. - 2025.
74 f. : il. color.

Digitado.

Trabalho de Conclusão de Curso (Graduação em Ciência da computação) - Universidade Estadual da Paraíba, Centro de Ciências Exatas e Sociais Aplicadas, 2025.

"Orientação : Prof. Dra. Jannayna Domingues Barros Filgueira, Coordenação do Curso de Computação - CCEA".

1. VRPTW. 2. Meta-heurísticas. 3. Logística. 4. Comparação de algoritmos. I. Título

21. ed. CDD 005.12

Elaborada por Kaliane Eveny Martins de Oliveira - CRB - 15/986 BSC7

MIQUÉIAS GARCIA DE LUCENA

IMPLEMENTAÇÃO E ANÁLISE DE META-HEURÍSTICAS HIBRIDAS PARA O VRPTW E APLICAÇÕES EM PROBLEMAS LOGÍSTICOS

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Ciência da Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação

Aprovada em: 05/06/2025.

BANCA EXAMINADORA

Documento assinado eletronicamente por:

- **Harllem Alves do Nascimento** (***.796.924-**), em **13/06/2025 19:30:55** com chave **1256aed048a611f0a7d306adb0a3afce**.
- **Danilo Coura Moreira** (***.504.434-**), em **14/06/2025 10:58:35** com chave **aa24b7d0492711f094261a7cc27eb1f9**.
- **Jannayna Domingues Barros Filgueira** (***.837.144-**), em **13/06/2025 19:08:52** com chave **fd97467e48a211f09a6b06adb0a3afce**.

Documento emitido pelo SUAP. Para comprovar sua autenticidade, faça a leitura do QrCode ao lado ou acesse https://suap.uepb.edu.br/comum/autenticar_documento/ e informe os dados a seguir.
Tipo de Documento: Folha de Aprovação do Projeto Final
Data da Emissão: 14/06/2025
Código de Autenticação: ef65b7

Dedico este trabalho a todos que me ajudaram durante a minha jornada neste curso, amigos, colegas e professores. Dedico também aos meus familiares e namorada que sempre me apoiaram, e foram de grande importância durante o período do desenvolvimento deste trabalho.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me conceder a saúde e forças todos os dias.

À toda a minha família, em especial à minha mãe Vanuza, que sempre me deu o suporte e o conforto, me apoiando em todos os momentos.

À minha companheira Elainny, por sempre estar ao meu lado em todos os momentos, me animando, incentivando e apoiando sempre que precisei durante o processo deste trabalho.

À minha professora e orientadora Drª. Jannayna Domingues, por seu esforço contínuo em me auxiliar no desenvolvimento deste trabalho, sempre disponível e presente para me orientar ao longo desse processo.

Aos meus amigos e colegas de faculdade, pelos apoios e incentivos durante o desenvolvimento deste trabalho.

RESUMO

Este trabalho tem como objetivo analisar o desempenho de diferentes algoritmos aplicados ao Problema de Roteamento de Veículos com Janelas de Tempo (VRPTW), uma das variantes do VRP mais estudadas e relevantes na área de otimização combinatória e logística, além da análise e comparação dos algoritmos, é apresentada uma demonstração da integração desses algoritmos em soluções de mapas reais. Para isso, foram selecionados e implementados três algoritmos meta-heurísticos, algoritmo Genético, Otimização por Colônia de Formigas e Busca Tabu, que utilizam da heurística construtiva I1 de Solomon (1987) como estratégia de iniciação e busca local *Relocate*, *Exchange*, 2-opt e 2-opt* para aprimoramento de soluções. Os testes foram realizados utilizando dados dos *datasets* já consolidados de Solomon (1987), com 56 instâncias de 100 clientes. Os resultados demonstram um destaque da Busca Tabu em relação aos demais algoritmos, com uma boa qualidade de solução e tempo de execução inferiores aos demais. Este estudo contribui para a seleção mais adequada de algoritmos em sistemas de roteamento aplicados à logística urbana, demonstrando os principais algoritmos e comparando algumas das principais soluções utilizadas.

Palavras-Chave: VRPTW; Meta-heurísticas; Logística; Comparação de algoritmos.

ABSTRACT

This work aims to analyze the performance of different algorithms applied to the Vehicle Routing Problem with Time Windows (VRPTW), one of the most studied and relevant VRP variants in the fields of combinatorial optimization and logistics. In addition to analyzing and comparing the algorithms, it presents a demonstration of their integration into real-world map solutions. To this end, three metaheuristic algorithms were selected and implemented Genetic Algorithm, Ant Colony Optimization, and Tabu Search, which employ Solomon's (1987) I1 constructive heuristic for initialization and the local search operators Relocate, Exchange, 2-opt, and 2-opt for solution refinement. Tests were conducted using data from the well-established Solomon (1987) datasets, consisting of 56 instances with 100 customers each. The results highlight the superior performance of Tabu Search over the other algorithms, achieving high solution quality and shorter execution times. This study contributes to the more informed selection of algorithms for routing systems in urban logistics by showcasing key methods and comparing some of the leading solutions in use.*

Keywords: *VRPTW; Metaheuristics; Logistics; Algorithm Comparison.*

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ACM	*Association for Computing Machinery*
ACO	*Ant Colony Optimization*
ALNS	*Adaptive Large Neighborhood Search*
CVRP	*Capacitated Vehicle Routing Problem*
CVRPLIB	*Capacitated Vehicle Routing Problem Library*
DIMACS	*Center for Discrete Mathematics and Theoretical Computer Science*
EA	*Evolutionary Algorithm*
FNDE	Fundo Nacional de Desenvolvimento da Educação
GA	*Genetic Algorithm*
GeoJSON	*Geographic JSON*
GIS	*Geographic Information System*
GLS	*Guided Local Search*
GVRP	*Green Vehicle Routing Problem*
HGS	*Hybrid Genetic Search*
IEEE	*Institute of Electrical and Electronics Engineers*
ILS	*Iterated Local Search*
JSON	*JavaScript Object Notation*
LNS	*Large Neighborhood Search*
MA	*Memetic Algorithm*
MDVRPTW	*Multi-Depot Vehicle Routing Problem with Time Windows*
mTSP	*multiple Traveling Salesman Problem*
OSM	OpenStreetMaps
RFC	*Request for Comment*
SA	*Simulated Annealing*
SREX	*Selective Route Exchange*
TMS	*Transportation Management System*
TS	*Tabu Search*
TSP	*Traveling Salesman Problem*
VNS	*Variable Neighborhood Search*
VRP	*Vehicle Routing Problem*
VRP-REP	*Vehicle Routing Problem Repository*
VRPTW	*Vehicle Routing Problem with Time Windows*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A logística é um processo essencial para o bom funcionamento das cadeias de entregas, pois otimiza o funcionamento do transporte de bens e serviços, o que gera um grande desafio na gerência de milhares de entregas. Com isso, existe uma grande demanda para uma solução que gerencie e otimize estes processos de forma eficiente, para cumprir com as altas demandas.

Um dos principais desafios está na gerência das rotas individuais para cada veículo que realiza as entregas aos consumidores finais, que visa otimizar uma frota de n veículos. Nesse contexto, o problema de roteamento de veículos é criado (*Vehicle Routing Problem* - VRP) buscando encontrar as rotas ótimas para melhorar o fluxo de trabalho de uma frota de veículos.

O VRP possui múltiplas variantes focadas em problemas específicos, como a variação da capacidade dos veículos, restrições de horário para a entrega definidas pelos clientes e a existência de múltiplos depósitos, que são apenas alguns dos exemplos de variações do problema. A gerência desses roteamentos é um dos maiores problemas que as empresas de logística enfrentam nos dias atuais, considerando que atualmente a demanda para essas soluções é uma das maiores da história (Konstantakopoulos; Gayialis; Kechagias, 2022).

O VRP teve uma das suas primeiras aparições no trabalho de Dantzig e Ramser (1959), que é apresentado como uma generalização do Problema do Caixeiro Viajante (*Traveling Salesman Problem* - TSP), utilizado para uma frota de caminhões de combustível.

O TSP busca encontrar a rota de menor custo que passe entre n pontos dados, assumindo que cada um dos pontos possui uma conexão entre si, com a rota de solução gerada passando necessariamente apenas uma vez por cada ponto. O número de possibilidades de variações possíveis para n pontos é calculado por $\frac{1}{2}n!$, com isso a complexidade do problema cresce rapidamente, dado $n = 15$, o número de rotas diferentes é 653,837,184,000 (Dantzig; Ramser, 1959).

Já o VRP pode ser descrito como um problema que busca encontrar rotas ótimas para entregas ou coletas de um ou múltiplos depósitos para cidades ou clientes espalhados geograficamente, e que está sujeito a restrições específicas (Laporte,

1992). Este possui diversas variantes, cada uma com particularidades e restrições específicas para atender as necessidades de seu local de aplicação.

Para métrica de exemplo, o site *Vehicle Routing Problem Repository* (VRP-REP) contém fontes de dados para testes de algoritmo de roteamento. Em novembro de 2024 o site contabiliza 59 variantes do VRP cadastradas (VEHICLE ROUTING PROBLEM REPOSITORY, 2014). Essa grande variedade se justifica devido às diversas aplicações que esse problema possui, cada uma com suas peculiaridades.

O alto número de variantes gera uma grande necessidade de estudo de algoritmos de roteamento para atender a esses critérios, e serem usados no planejamento de rotas para frotas de veículos, aplicando a áreas como logística de entregas, mobilidade urbana, cadeias de suprimentos, ou até mesmo para emergências, como roteamento de ambulâncias.

A ideia central do presente trabalho se baseia em uma pesquisa primária que busca avaliar o desempenho e utilidade de meta-heurísticas híbridas, utilizando de estratégias construtivas e busca local aplicados na resolução do problema de roteamento de veículos com janela de tempo (VRPTW). Este problema combina restrições que são amplamente utilizadas no problema de roteamento de veículos, sendo elas a janela de tempo, onde cada consumidor possui um tempo inicial e final para seu serviço e a capacidade de carga, que deve respeitar o limite de capacidade de cada veículo da frota.

Os testes de desempenho são introduzidos neste contexto para avaliar, utilizando de *datasets* reconhecidos pela literatura, o desempenho, consistência e qualidade dos algoritmos aplicados, e com a análise dos resultados identificar forças e fraquezas dos algoritmos. Por fim, após a análise dos resultados, demonstrando a aplicabilidade desses algoritmos em contextos reais, com base em ferramentas atuais, que facilitam a criação de um sistema de roteamento.

1.1 Objetivos

O presente trabalho tem como objetivo geral testar, analisar e demonstrar aplicação para as principais soluções do VRPTW utilizando algoritmos meta-heurísticos híbridos. Para isto, apresenta-se os seguintes objetivos específicos:

- Explorar os principais algoritmos utilizados em pesquisas para resolução do VRPTW;

- Selecionar principais *datasets* da literatura para avaliação dos algoritmos selecionados;
- Criar ambiente de testes e algoritmos com base nas instâncias selecionadas;
- Realizar análise dos dados obtidos nos testes, buscando avaliar pontos fracos e fortes dos algoritmos, além da avaliação da usabilidade para o problema;
- Implementar um protótipo com base nos algoritmos selecionados e testados para demonstrar a usabilidade;
- Identificar tendências emergentes na área.

1.2 Resumo da Metodologia

O presente trabalho realiza uma pesquisa primária, buscando pesquisar e analisar fatores importantes de algoritmos genéticos, busca tabu e otimização por colônia de formigas, meta-heurísticas amplamente aplicadas para o problema de roteamento de veículos com janela de tempo (VRPTW), que são utilizadas em conjunto com uma fase construtiva inicial e busca local para melhora de solução.

Para isso, utilizaram-se de testes por meio dos *datasets* de Solomon (1989) e comparação com resultados do estado da arte disponíveis no site *Capacitated Vehicle Routing Problem Library* (CVRPLIB) (CVRPLIB [...], 2025). Além disso, explorar e demonstrar o uso de ferramentas para implementação de protótipo demonstrativo para utilização em cenários reais por meio do OpenStreetMaps.

1.3 Organização e Estrutura do Trabalho

No Capítulo 1 (Introdução) será discutido sobre a importância dos algoritmos de geração de rotas para diversas áreas da sociedade. Além disso, serão apresentados os objetivos, relevância, justificativa e resumo da metodologia do trabalho. Em seguida o Capítulo 2 (Fundamentação teórica) apresenta os fundamentos teóricos utilizados no trabalho, explorando conceitos logísticos e geográficos, definição do VRPTW, algoritmos importantes e testes de desempenho, estabelecendo a base de conhecimentos para o desenvolvimento e compreensão do trabalho.

O Capítulo 3 (Metodologia) apresenta a metodologia utilizada na pesquisa, métodos e ferramentas usados na construção dos testes, algoritmos e para o desenvolvimento do protótipo. Após a definição dos métodos, os resultados são

demonstrados no Capítulo 4 (Resultados e Discussões), onde é feita a amostragem e discussões dos resultados obtidos por meio dos testes e do protótipo. Por fim, é feita as conclusões do trabalho no Capítulo 5 (Conclusão), além dos trabalhos futuros que podem dar continuidade a esta pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os principais tópicos para o desenvolvimento deste trabalho. Uma breve contextualização da área de logística e conceitos importantes dos algoritmos em cenários reais, definição do Problema de Roteamento de Veículos (*Vehicle Routing Problem* - VRP) e da variante usada no trabalho, apresentação dos algoritmos importantes para o Problema de Roteamento de Veículos com Janela de Tempo (*Vehicle Routing Problem with Time Windows* - VRPTW), além das noções de testes de desempenho no contexto de algoritmos de otimização combinatória.

2.1 Sistema Logístico

Uma das principais funções do sistema logístico é o transporte. Os procedimentos de fornecimento e distribuição exigem uma gestão eficaz da frota utilizada para o transporte. A gestão de forma eficaz dos transportes vem acompanhada de muitos desafios, que estão vinculados a três níveis de planejamento, sendo eles o estratégico, tático e operacional (Anbuudayasankar; Ganesh; Mohapatra, 2014).

Como exemplo, a escolha da localização dos depósitos de fornecimento se encaixa no planejamento estratégico, problemas táticos estão vinculados a fatores como problemas de tamanho da frota de veículos e por último, a roteirização e planejamento das rotas dos veículos se encaixam no planejamento operacional (Anbuudayasankar; Ganesh; Mohapatra, 2014).

Nos últimos anos, o mercado de logística vem se expandindo com uma crescente demanda, o que pode estar vinculado ao aumento de compras online, que cresce de forma constante devido à acessibilidade de dispositivos móveis (Lu; Yang, 2019). Uma parte considerável do valor final dos produtos vem do seu custo de distribuição, sendo assim, as empresas estão em uma constante busca de redução de custos de entregas, para assim obter uma maior demanda de seus produtos (Konstantakopoulos; Gayialis; Kechagias, 2022).

Soluções que atendam a essas demandas de forma eficiente são computacionalmente complexas, com diversas necessidades específicas para atender a cada caso, porém, o transporte desses bens e serviços por meio de veículos

tem sido uma tarefa imprescindível para a sociedade, já que altos valores são gastos em recursos como combustível, manutenção e operação de veículos, além da mão de obra (Anbuudayasankar; Ganesh; Mohapatra, 2014).

Portanto, melhorias que otimizem essas atividades têm um grande potencial de gerar economias significativas. Um exemplo recente da aplicação da roteirização em sistemas logísticos se encontra no relatório integrado dos Correios (2023, p. 82):

> "implantação da roteirização via sistema de gerenciamento de transporte (*Transportation Management System* - TMS) nas operações de concursos e do Fundo Nacional de Desenvolvimento da Educação - FNDE. A implantação do sistema de roteirização permitiu a otimização das rotas de entrega, com redução de recursos de distribuição superiores a 10%."

Com isso, pode-se notar o impacto e significância dessas soluções para o mundo real. Para entender a base destes sistemas de gerenciamento, é necessário compreender alguns conceitos principais que fazem esse gerenciamento ser possível.

2.2 Sistema de Coordenadas e Pontos

Os sistemas de coordenadas são utilizados em mapas para identificar determinadas posições no espaço geográfico real. O globo possui divisões compostas pelos meridianos e paralelos, que respectivamente são representados pela latitude e longitude. (Barczyszyn, 2019). Com isso, temos um sistema onde cada coordenada corresponde a um ponto no espaço.

Esse sistema facilita muito a identificação de forma única de elementos em um mapa, já que a partir do georreferenciamento cada elemento recebe uma coordenada de latitude e longitude, o que garante a identificação da localização desta entidade (Barczyszyn, 2019). Com isso é possível construir soluções digitais que espelham o mundo real e possibilitem a manipulação desses dados no meio digital.

Com a capacidade de identificação dos pontos é construído então um grafo contendo as conexões (estradas) que unem cada um desses pontos. Por meio deste grafo, é determinada a sequência de estados que subdividem a trajetória, com o sistema de planejamento as trajetórias visando encontrar o melhor caminho que una os estados inicial e final de um determinado trajeto, de forma que respeite as restrições impostas pelo problema (Souza, 2023).

2.3 Cálculos de Distância

Um dos fatores a serem considerados quando se trata de um problema de roteamento é a distância entre dois pontos. Com o auxílio da identificação dos pontos por meio de coordenadas geográficas, existem métodos utilizados para calcular essa distância. A utilização destes cálculos para os testes de desempenho dos algoritmos é comumente limitada à distância euclidiana.

2.3.1 Distância Euclidiana

A distância euclidiana calcula a distância entre dois pontos utilizando como base o Teorema de Pitágoras, pelo valor da linha diagonal do triângulo (MARIA *et al*., 2020). Considerando (x_1 , y_1) e (x_2 , y_2) dois pontos no espaço, a distância é calculada com a seguinte fórmula:

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (1)$$

onde:

- D é a distância resultante em graus
- x_1 é a latitude do ponto inicial
- x_2 é a latitude do ponto final
- y_1 é a longitude do ponto inicial
- y_2 é a longitude do ponto final

O resultado da fórmula pode ser convertido para unidades de distância multiplicando-se os graus pela distância correspondente no globo, com $1\,grau \simeq 111{,}319\,km = 69\,milhas$, assim obtendo o resultado em distância. (Maria *et al*., 2020).

2.3.2 Fórmula de Haversine

Com a fórmula de Haversine é possível calcular a distância de forma mais precisa, já que a mesma calcula o círculo resultante entre os dois pontos, para isso assumindo o raio da terra como 6367,45 km (Maria *et al*., 2020). Na sua definição, de forma geral, o cálculo é feito por meio da seguinte fórmula (Maria *et al*., 2020; Andreou *et al*., 2023):

$$d = 2 \times R \times arcsen(\sqrt{sen^2 \times (\frac{x_2 - x_1}{2}) + cos(x_1) \times cos(x_2) \times sen^2 \times (\frac{y_2 - y_1}{2})}) \quad (2)$$

onde:

- d é a distância resultante
- R é o raio da esfera, dado por 6367,45 km
- x_1 é a latitude do ponto inicial
- x_2 é a latitude do ponto final
- y_1 é a longitude do ponto inicial
- y_2 é a longitude do ponto final

A distância de Haversine consegue atingir um maior grau de precisão, porém ainda existe um erro associado, que é de aproximadamente 0,5%, devido ao formato de uma elipsóide irregular que o planeta terra possui, com o formato de esfera neste caso gerando apenas resultados aproximados, existindo ainda soluções mais precisas como a formula de Vincenty (Andreou *et al*., 2023).

2.4 Cálculo de Menor Caminho

A maioria das ferramentas para o Problema de Roteamento de Veículo utilizam algoritmos de busca de menor caminho, como algoritmo Dijkstra, para definir distâncias e tempo de percurso entre pontos (Toth; Vigo, 2014). Esses valores são frequentemente pré-calculados e armazenados em uma matriz de distâncias.

O algoritmo Dijkstra foi criado em 1959 por Edsger Wybe Dijkstra, para calcular o menor caminho entre um nó inicial e n nós finais existentes, considerando um grafo sem arestas com valores negativos. Para isso, o algoritmo constrói interativamente uma árvore de dados contendo a sequência de pontos com o menor custo para alcançar os nós finais (Dijkstra, 1959). O algoritmo pode ser adaptado para encontrar o menor caminho entre um número limitado de nós, finalizando suas iterações ao encontrar todos os menores caminhos.

2.5 Problema do Caixeiro Viajante (TSP)

O problema do caixeiro viajante (*Traveling Salesman Problem* - TSP), é um dos problemas de otimização combinatória mais conhecidos, pode ser definido como um número n de cidades que devem ser visitadas apenas uma vez por um caixeiro

viajante, considerando que a rota deve ser finalizada na mesma cidade em que foi iniciada (El-Sherbeny, 2010).

O TSP possui uma definição simples, mas ao mesmo tempo foi um dos primeiros problemas a serem provados como *NP-Hard* por Karp em 1972, no desenvolvimento da teoria *NP-complete* (Jünger; Reinelt; Rinaldi, 1995). O principal objetivo deste problema é minimizar a distância total percorrida pelo caixeiro viajante.

Além da versão clássica, existe também uma versão expandida, que se trata do problema de múltiplos caixeiros viajantes (*multiple Traveling Salesman Problem* - mTSP), em que m caixeiros viajantes visitam n cidades (nó), considerando também que cada nó deve ser visitado apenas uma vez, com ambos os modelos sendo puramente problemas de roteamento (Anbuudayasankar; Ganesh; Mohapatra, 2014).

2.6 Problema do Roteamento de Veículos (VRP)

O problema de roteamento de veículos (*Vehicle Routing Problem* - VRP) possui os mesmos conceitos do mTSP, porém com uma frota de veículos ao invés dos caixeiros viajantes, considerando também a capacidade dos veículos na sua forma padrão (Anbuudayasankar; Ganesh; Mohapatra, 2014).

Na definição do problema, existem n requisições de entregas para clientes, onde cada requisição é definida como pontos em uma rede de estradas (Toth; Vigo, 2014), podendo ser representada em forma de grafos, onde cada ponto representa um vértice e as estradas representam as arestas. A aplicação do VRP no mundo real por meio de soluções computacionais tem se provado eficiente, gerando economias consideráveis em custos globais em transporte (Toth; Vigo, 2014).

2.7.1 Problema de Roteamento de Veículos Capacitados (CVRP)

O problema de roteamento de veículos capacitados (*Capacitated Vehicle Routing Problem* - CVRP), possui como principal característica considerar a capacidade de pacotes que cada veículo pode acomodar, assumindo que todos os veículos contém a mesma capacidade de transporte de pacotes, sendo uma das variantes mais simples, geralmente utilizada juntamente com outras variantes como o VRPTW (Konstantakopoulos; Gayialis; Kechagias, 2022).

O CVRP pode ser considerado como a versão básica do VRP, consistindo na busca de uma coleção de rotas, onde cada cliente é visitado apenas por uma rota,

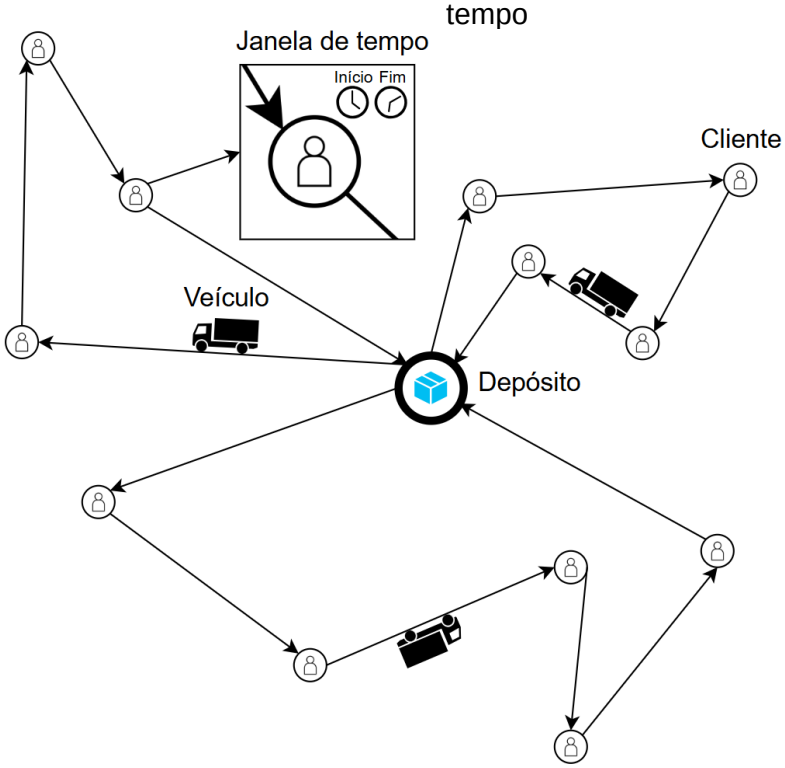
com cada rota partindo e retornando a um depósito 0 e servindo n clientes, sem violar o limite de capacidade de cada veículo. (Toth; Vigo, 2002).

2.7.2 Problema de Roteamento de Veículos com Janela de Tempo (VRPTW)

O Problema de roteamento de veículos com janela de tempo (*Vehicle Routing Problem with Time Windows* - VRPTW) é uma extensão do CVRP, onde cada cliente possui um intervalo de tempo inicial e final, chamado de janela de tempo. Essa janela de tempo delimita o tempo em que esse cliente pode ser servido, essa janela de tempo pode ser rígida ou suaves. No caso rígido o veículo deve respeitar as restrições, e esperar até o tempo inicial de serviço caso chegue muito cedo, já na forma suave, esse veículo pode violar essa janela de tempo, com uma penalidade associada (Toth; Vigo, 2014).

O VRPTW é uma das variantes mais estudadas na área de roteamento de veículos, por se adequar a problemas da vida real (Konstantakopoulos; Gayialis; Kechagias, 2022). A Figura 1 demonstra de forma simplificada como o VRPTW funciona, contendo os consumidores, que devem ser atendidos dentro de uma janela de tempo determinada, por múltiplos veículos que iniciam e finalizam sua rota em um depósito.

Figura 1 - Ilustração do problema do roteamento de veículos com janela de tempo



Fonte: *Autoria própria.*

O VRPTW pode ser definido de forma simplificada como um grafo $G = (V, A)$, com o depósito representando como o primeiro e último vértices 0 e $n + 1$, portanto $|V| = n + 2$. Os clientes são os demais vértices $C = V \setminus \{0, n + 1\}$. Cada um dos clientes possui uma janela de tempo $[a_i, b_i]$ que delimita o período em que o cliente deve ser servido, além de um tempo necessário para finalização de serviço s_i e demanda de carga q_i. A demanda e tempo de serviço para 0 e $n + 1$ é zero, $q_0 = q_{n+1} = s_0 = s_{n+1} = 0$.

É definido ainda uma frota de veículos K, em que cada veículo possui uma capacidade máxima Q. Todas as rotas de veículos válidas possuem o depósito como início e fim $[0, \ldots, n + 1]$, porém com essas rotas podendo ser ainda considerada inválidas caso violem as janelas de tempo delimitada como $a_i \leq T_{ik} \leq b_i$ considerando T_{ik} o tempo de início de serviço no cliente i, ou que violem a capacidade máxima dos veículos $q_i + q_j \geq Q$. (Cordeau *et al*., 2007; Toth; Vigo, 2014).

2.8 Trabalhos Correlatos

A *12th Implementation Challenge* do DIMACS (2021) tratou do estudo do VRP, com pesquisas demonstrando algoritmos de solução para 8 variantes do VRP, que incluem o VRPTW. Com os participantes demonstrando seus resultados e pesquisas por meio de artigos curtos e vídeos, os trabalhos são listados na sua plataforma web (DIMACS, 2022).

Os principais trabalhos do evento no contexto do VRPTW foram o Kool *et al*. (2022) que adaptou a implementação *open-source* HGS *for the Capacitated Vehicle Routing Problem* (HGS-CVRP) para adicionar suporte a janela de tempo, além utilizar conceitos como heurística construtiva, operador de cruzamento *Selective Route Exchange* (SREX) e busca local SWAP*.

Li *et al*. (2022) desenvolve uma solução baseada em um algoritmo memético de busca local, de dois níveis, com a primeira parte minimizando o número de rotas e a segunda à distância das mesmas. Jiang *et al*. (2022) constrói uma solução por meio de ajustes dinâmicos no número de rotas, com um algoritmo memético de população de base iterativa, que inicia com uma população menor, e expande a população a cada convergência.

Na revisão sistemática da literatura de Liu *et al*. (2023) que analisou 10.556 artigos, do período de 2018 a 2022, nas bases de dados: *Sciencedirect*, ACM, Wiley,

Springer Link, IEEE Xplore. Após a filtragem de critérios foram selecionados 64 artigos, os resultados apresentados mostram que 86% dos trabalhos utilizaram métodos aproximados, sendo 17% destes heurísticos e 69% meta-heurísticos.

O estudo ilustra os algoritmos de cada artigo revisado, onde se pode notar que a maior recorrência de Otimização por Colônia de Formigas (*Ant Colony Optimization* - ACO), Algoritmo Genético (*Genetic Algorithm* - GA), Busca Adaptativa em Grande Vizinhança (*Adaptive Large Neighborhood Search* - ALNS), Busca Tabu (*Tabu Search* - TS) e Busca de Vizinhança Variável (*Variable Neighborhood Search* - VNS) considerando também os usos de forma híbrida (Liu *et al*., 2023).

Elshaer e Awad (2020) em sua revisão taxonômica sobre o VRP, classificou 299 artigos sobre o tema, entre 2009 e 2017, a partir da amostragem da classificação, feita a partir dos algoritmos relacionados às variantes do VRP. Analisando especificamente o VRPTW, sem considerar suas sub variantes apresentadas no trabalho, a maior quantidade utiliza de Algoritmo Genético, Busca Tabu básica, Recozimento Simulado (*Simulated Annealing* - SA), Busca em Grande Vizinhança e Algoritmo Memético (*Memetic Algorithm* - MA). Quanto às sub variantes do VRPTW a maior quantidade se relaciona com TS básico, GA, LNS, ALNS e ACO.

No trabalho de Toth e Vigo (2014), que analisa a literatura a partir do ano 2000 até a data da sua publicação, é ilustrado que heurísticas mais bem sucedidas para VRPTW envolvem busca local. Para a classe de meta-heurísticas são demonstrados os melhores resultados registrados no período da pesquisa para os *datasets* de Solomon (1987) e Gehring e Homberger (1999).

É destacado o uso de alguma variante de busca local em todas as soluções, o autor destaca que um fator chave para o sucesso da meta-heurística é a possibilidade de visitar soluções não válidas no decorrer da execução. Com a amostragem dos dados, é possível notar um destaque para técnicas de busca local *Neighborhood filtering, 2-opt e 2-opt*,* além de algoritmos LNS, Algoritmo Evolucionários (*Evolutionary algorithms* - EA) ou estratégias dessa classe, Busca local Iterada (*Iterated local search* - ILS*)*, TS, Busca local Guiada (*guided local search* - GLS*)* e SA.

A revisão de Konstantakopoulos, Gayialis e Kechagias (2022) em sua análise selecionou 263 artigos, dentre esses 122 com a variante VRPTW, em sua correlação com os algoritmos utilizados, nota-se um grande uso de algoritmos construtivos geralmente utilizados para criação de solução inicial e heurísticas de melhoria local.

As meta-heurísticas mais pesquisadas utilizam GA, TS, (A)LNS, VNS, EA e ACO, com os autores destacando um aumento no uso de EA's multiobjetivo.

2.9 Algoritmos Heurísticos

Algumas das soluções heurísticas importantes no contexto do VRP são as heurísticas construtivas e de busca local, já que conseguem gerar bons resultados com um baixo custo computacional (Bräysy; Gendreau, 2005). O que ocorre devido a sua estrutura mais simples, que exige menos complexidade computacional.

2.9.1 Algoritmos Construtivos

As heurísticas construtivas criam soluções viáveis por meio da seleção de clientes com base em critérios de minimização, que devem ser controlados para não violar as restrições de janela de tempo e capacidade (Bräysy; Gendreau, 2005). Algumas das principais heurísticas construtivas utilizadas são apresentadas a seguir:

- *Savings algorithm*: Inicia com uma solução em que cada cliente é servido por uma rota individual, e posteriormente combina essas rotas buscando gerar melhores economias "*savings*" (Solomon, 1987).
- *Nearest Neighbor*: Busca sempre encontrar o cliente não roteado mais próximo ao último elemento da rota ou ao depósito caso seja o início de uma nova rota (Solomon, 1987).
- *Cheapest Insertion*: Tenta inserir iterativamente o cliente na rota, entre cada cliente i e j, de maneira que gere o menor aumento de custo. Iniciando uma nova rota com base em um critério predefinido, como a heurística I1 apresentada por Solomon (Solomon, 1987).
- *Sweep algorithm*: Particiona os clientes de acordo com seu ângulo polar em torno de um ponto central (Bräysy; Gendreau, 2005). Para o VRPTW pode se utilizar de duas fases, como o *time-oriented sweep* de Solomon (1987), que em sua primeira fase os clientes são vinculados aos veículos utilizando o algoritmo *Sweep*, em seguida, esses clientes são roteados por outra heurística, como uma heurística de inserção (Bräysy; Gendreau, 2005).

2.9.2 Busca Local (Neighborhood search)

A busca local utiliza de heurísticas para melhoria de rota, para isso, realiza buscas em vizinhanças a partir de uma solução, por meio de modificações na sua estrutura, visando encontrar melhorias. Na sua forma tradicional pode ser dividida em duas categorias: dentro da rota (*intra-route*) e entre rotas (*inter-route*) (Toth; Vigo, 2014).

A categoria *intra-route* envolve uma busca local dentro de apenas uma rota, enquanto *inter-route* envolve a busca local entre múltiplas rotas (Liu *et al*. 2023). Esses operadores podem ser usados em conjunto com outros algoritmos, ajudando na melhora da solução, mas com um custo computacional adicional.

Alguns dos algoritmos que são comumente utilizados em meta-heurísticas importantes são apresentados a seguir, baseando-se em (El-Sherbeny, 2010, Toth; Vigo, 2014, Bräysy; Gendreau, 2005):

- *Relocate* (*inter-route*), que remove um cliente de uma rota, e insere em outra rota.
- *Exchange* (*inter-route*), faz a troca de dois clientes entre duas rotas diferentes.
- 2-opt (*intra-route*), considerado parte das heurísticas λ-opt, que realizam a remoção de λ arcos e substitui por outros λ arcos possíveis para conectar a rota, considerando também rotas que mudem a orientação do trajeto. Possui uma complexidade de $O(n^{\lambda})$ para n clientes, a melhor solução encontrada é denominada λ-ótima.
- Or-opt (*intra-route*), realiza a troca de uma sub-rota para uma posição diferente na mesma rota, considerando a preservação da orientação da rota. A vizinhança geralmente é reduzida utilizando uma quantidade limitada dos clientes ou considerando soluções próximas à original.
- 2-opt* (*inter-route*), possui uma definição similar ao λ-opt, mas realizando a troca entre duas rotas diferentes, ou seja, realiza a troca de um arco de uma rota com outro arco de outra rota.
- *Cross-exchange* (*inter-route*), troca posições de duas sub-rotas, removendo 4 arcos e substituindo eles por outros 4 arcos. Geralmente usando um número limitado de clientes para diminuir o número de operações.
- *Path Relocation Neighborhood* (*inter-route*), realoca sub-rotas de uma rota para a outra, removendo 3 arcos e adicionando em outra rota, pode permitir inversão

da orientação da rota, normalmente com uma quantidade de realocações de sub-rotas limitada.

O 2-opt, Or-opt, 2-opt*, *Cross-exchange* e *Path Relocation Neighborhood* são os principais operadores utilizados em meta-heurísticas importantes segundo Toth e Vigo (2014), existem muitos outros operadores de busca local, como por exemplo os operadores, *λ-interchange*, *shift-sequence*, GENI-*exchange*, *Cycle-Transfer*, K-*node interchange* (Bräysy; Gendreau, 2005; El-Sherbeny, 2010).

2.10 Algoritmos Meta-heurísticos

As Meta-heurísticas são outro método muito aplicado para o planejamento de rotas, já que contém procedimentos avançados como busca populacional, além de métodos de busca local e têm grande ênfase dentro do contexto de VRP (Konstantakopoulos; Gayialis; Kechagias, 2022). Métodos meta-heurísticos utilizam de técnicas para evitar resultados mínimos locais, para se obter melhores resultados em tempo de execução em comparação com as variações que buscam resultados exatos (Labadie; Prins; Prodhon, 2016). Além disso, uma Meta-heurística é um algoritmo mais geral, com a possibilidade de ser aplicado em diferentes problemas de otimização com relativamente poucas modificações (Dorigo; Birattari; Stutzle, 2006). A seguir são apresentadas as principais meta-heurísticas utilizadas no contexto do VRPTW.

2.10.1 Algoritmos Genéticos (GA)

Introduzido por Holland (1975), trata-se de um algoritmo baseado em população que utiliza mecanismos baseados na seleção natural. O algoritmo inicia definindo um conjunto de soluções que serve como uma população inicial, cada uma dessas soluções é definida como um cromossomo. Cada um desses cromossomos é avaliado de acordo com uma função de aptidão que avalia o quão boa é seu resultado, a partir daí são criadas novas gerações utilizando dos melhores indivíduos da população com mutação e cruzamento de cromossomos (Kumar *et al*., 2010).

O GA é um dos algoritmos meta-heurísticos mais comumente utilizados para solucionar variantes VRP (Konstantakopoulos; Gayialis; Kechagias, 2022), os cromossomos nesse caso representam a solução de rotas. Variantes que hibridizam

algoritmos genéticos com estratégias como busca local podem ser denominadas como algoritmos meméticos (Toth; Vigo, 2014).

Alguns dos operadores de cruzamento comumente utilizados no contexto do VRPTW são o cruzamento por montagem de arestas (*Edge Assembly Crossover*) que tem um foco nas conexões entre clientes, gera soluções possivelmente não viáveis, e necessita de processos de reparação da solução, outro operador é o cruzamento por ordem (*Order Crossover*) juntamente com a representação por rota gigante que considera a solução como uma rota que abrange todos os clientes sequenciados, realizando o cruzamento por meio dessa representação, e utilizando um processo de separação da rota gigante posteriormente (Toth; Vigo, 2014).

2.10.2 Busca Local Iterada (ILS)

A busca local iterada (*Iterated Local Search* - ILS) tem como base a busca local de forma iterativa, porém para fugir dos ótimos locais, após realizar a melhoria inicial da solução, são aplicadas perturbações na solução, aplicando movimentos aleatórios na solução, e posteriormente a busca local é reaplicada e a nova solução obtida avaliada de acordo com a estratégia utilizada, assim o algoritmo itera até atingir algum critério de parada definido (Lourenço; Martin; Stützle, 2003).

2.10.3 Otimização da Colônia das Formigas (ACO)

Em 1992, Dorigo elaborou a ideia do algoritmo de otimização por colônia de formigas (Abu *et al*., 2022). Este algoritmo meta-heurístico se baseia no comportamento de formigas, que na procura de uma fonte de comida, liberam feromônios pelos caminhos percorridos ao encontrar alimento, este feromônio influencia as demais formigas a seguir o mesmo caminho, e serve como um meio de comunicação entre as formigas, facilitando que as próximas formigas encontrem um melhor caminho (Nha; Djahel; Murphy, 2012).

No meio computacional, o algoritmo cria formigas artificiais como procedimento de construção probabilístico, utilizando uma informação heurística do problema e a distribuição artificial dos feromônios que são dispersos pelas formigas. O algoritmo consiste em três principais passos: simulação das formigas para construção de soluções por meio da heurística e do feromônio, as soluções geradas

podem ser melhoradas com uma fase de busca local, por fim as trilhas de feromônios são adaptadas a partir das soluções encontradas (Dorigo; Stützle, 2019).

A escolha de uma solução gerada pode ser dada pela seguinte fórmula, baseado em Dorigo e Stützle (2019):

$$P_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}} \quad (3)$$

Onde:

- $\tau_{ij}(t)$ é a quantidade de feromônio na aresta (i, j) no instante t.
- η_{ij} é o valor heurístico associado à aresta (i, j).
- α e β são constantes que controlam a influência do feromônio e da heurística, respectivamente.
- N_i^k é o conjunto de nós não visitados por k a partir do nó i.

Com a atualização dos feromônios depositados dado por:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum \quad \Delta\tau_{ij}^{k}(t) \qquad 0 < \rho < 1 \quad (4)$$

Onde:

- ρ é a constante de taxa de evaporação do feromônio.
- m é o número total de formigas.
- $\Delta\tau_{ij}^{k}(t)$ é a quantidade de feromônio depositado pela formiga k na aresta (i, j), que é definida por:

$$\Delta\tau_{ij}^{k}(t) = \frac{Q}{L_k} \quad (5)$$

Onde:

- Q constante de quantidade de feromônio lançado.
- L_k é o custo total da solução.

Assim, por meio dessas fórmulas é possível construir as formigas artificialmente de forma iterativa, com cada uma das formigas construindo as soluções e depositando os feromônios para guiar as novas formigas da próxima iteração.

2.10.4 Recozimento Simulado (SA)

Introduzido por Kirkpatrick, Gelatt e Vecchi (1983) para otimização combinatória, o recozimento simulado (*Simulated Annealing* - SA) foi um das primeiras soluções meta-heurísticas para o problema CVRP, em um artigo de 1993 que Osman introduziu o conceito de λ-*interchance moves* (Labadie; Prins; Prodhon, 2016).

O algoritmo se baseia em conceitos da termodinâmica, onde se é observado que metais que são esquentados e posteriormente resfriados lentamente, dão aos átomos tempo para se estabelecer de uma forma mais organizada e ótima, assim, o SA é desenvolvido com a “Temperatura” que representa uma variável predefinida, que diminui gradualmente ao longo das interações (Labadie; Prins; Prodhon, 2016).

Essa variável de temperatura é utilizada para determinar a probabilidade de uma solução que não melhora a função objetivo ser escolhida, o cálculo da probabilidade de escolha é determinado pela seguinte equação, com base em (Labadie; Prins; Prodhon, 2016):

$$P = exp\left(-\frac{\triangle E}{T}\right) \tag{6}$$

Onde:

- $\triangle E$ representa o valor da solução analisada.
- T representa a temperatura atual.

A variação da temperatura é feita a partir da fórmula:

$$T_{novo} = \alpha \cdot T_{atual} \qquad 0 < \alpha < 1 \tag{7}$$

Com base desses conceitos, o algoritmo consegue iniciar com uma maior capacidade de exploração de soluções, e ao final possui uma maior chance de convergência para uma solução ótima local, e possivelmente, ótima global.

2.10.5 Tabu Search (TS)

Proposto por Glover (1986) como uma estratégia para resolver problemas combinatórios, utiliza de uma tabela “Tabu” que contém movimentos proibidos de serem selecionados, mas que podem se sobrepor a essa regra caso atinja um certo critério, como uma melhoria global da solução. Uma solução adicionada a lista tabu sai desse estado após um período curto de iterações, possibilitando novamente a sua

escolha. A ideia do algoritmo busca a não repetição de soluções semelhantes, com isso possivelmente saindo de ótimos locais (Glover, 1986). No contexto do VRP, o TS utiliza de busca local para explorar as vizinhanças e gerar possíveis novos resultados, que são avaliados e selecionados de acordo com sua qualidade e lista tabu.

2.10.6 Busca em Vizinhança Variável (VNS)

Introduzido por Mladenović e Hansen (1997), explora vizinhanças mais distantes da solução atual com rotinas de pesquisa local repetidamente, até atingir um critério de parada, aliado com mudanças dos operadores de busca local. Os principais passos do algoritmo incluem uma fase de perturbação da solução atual, que ajuda o algoritmo a escapar de ótimos locais, a busca local para melhoria da solução e a troca do operador de busca local (Hansen *et al*., 2017). A troca de operadores pode ser feita utilizando algumas estratégias, algumas delas são, baseado em (Hansen *et al*., 2017):

- Sequencial, onde as vizinhanças são exploradas em ordem fixa e o processo é reiniciado a cada melhoria encontrada.
- Cíclico, em que são alternadas em um ciclo contínuo, sem reiniciar o processo.
- *Pipe,* mesmo com melhorias encontradas, a exploração continua no mesmo operador, de forma encadeada.
- Skewed, que avalia as soluções considerando a distância entre a solução atual e a encontrada, possibilitando aceitar soluções com piora, caso seja distante o suficiente da atual, ajudando na diversificação.

2.10.7 Busca em Grande Vizinhança (LNS)

Formulado por Shaw (1998) para solucionar VRPs, explora grandes vizinhanças por meio da destruição e reconstrução da solução, removendo e reinserindo um conjunto de N clientes. Os principais fatores que influenciam o algoritmo são a escolha dos clientes a serem removidos e qual processo será utilizado para reinserir na reconstrução (Shaw, 1998).

A escolha de destruição pode ser feita de forma completamente aleatória, ou considerar algum critério como maior distância entre clientes, onde se deve também definir a porcentagem de destruição. Já a reconstrução pode ser feita por meio de

algoritmos construtivos, como o método *greedy*, o que faz o algoritmo LNS facilmente aplicável em conjunto com outros algoritmos (Pisinger; Ropke, 2019; Shaw, 1998).

2.10.8 Busca Adaptativa em Grande Vizinhança (ALNS)

Se baseia nos conceitos de destruição e reconstrução do LNS, realizando esse processo de forma iterativa. Porém o ALNS, são definidos múltiplos operadores de destruição e reconstrução de soluções, onde a cada iteração são selecionados dois operadores para efetuar essas operações, aumentando assim a diversificação do algoritmo. Um dos métodos para escolha dos operadores é a seleção por roleta, onde cada operador possui uma probabilidade de ser escolhido, essas escolhas são balanceadas com base no histórico de desempenho de cada operador (Mara *et al*., 2022).

Para a estratégia de escolha de soluções se usa do Critério de Metropolis para facilitar a chances do algoritmo de escapar de ótimos locais, com conceitos semelhantes ao já citado SA, utilizando de uma temperatura T que determina a chance de escolha de solução com piora, que vai gradualmente diminuindo durante a execução do algoritmo, assim garantindo uma melhor exploração da solução inicialmente, e um bom refinamento da solução ao final das iterações (Mara *et al*., 2022).

2.11 Testes de Desempenho

Testes de *software* são uma maneira de avaliar a qualidade e bom funcionamento de um sistema, com processos que validam se um sistema satisfaz a requisitos especificados ou não, visando encontrar *bugs*, erros ou falta de funções, fornecendo assim um conhecimento sobre a qualidade do *software* (Jamil *et al*., 2016). Os testes são uma forma consolidada e amplamente utilizada para avaliar diversos critérios específicos. No contexto de algoritmos de otimização, são utilizados testes de desempenho para avaliar a qualidade das soluções obtidas, essa tarefa envolve avaliar alguns critérios para gerar soluções justas (Beiranvand; Hare; Lucet, 2017).

O presente trabalho visa avaliar algoritmos que contém em seus processos, lógicas aleatórias, por isso é essencial desenvolver e analisar os testes considerando esses fatores. Um algoritmo randômico pode ser muito afetado pelo acaso, em alguns

casos encontrando soluções ótimas rapidamente ao mesmo tempo que tem a possibilidade de nunca convergir para uma solução aceitável, sendo importante estudar a distribuição das probabilidades dos seus resultados ou métricas de desempenho (Arcuri; Briand, 2011).

Outro ponto importante a ser considerado ao realizar testes em algoritmos de otimização é o viés da solução inicial fornecida para o algoritmo. Fornecer soluções iniciais diferentes para cada algoritmo, o que pode resultar em resultados enviesados (Beiranvand; Hare; Lucet, 2017). Portanto, o teste de algoritmos no contexto de meta-heurísticas requer balanceamentos para não enviesar os resultados do experimento, assim gerando resultados justos para serem avaliados.

Para realizar essa análise dos resultados no contexto de algoritmos de otimização, utilizam-se de muitas estratégias como Tabelas, para apresentar dados brutos ou trabalhados, *Boxplots* para ilustrar destruições estatísticas e perfis de desempenho que possibilitam comparar de forma gráfica algoritmos em múltiplas instâncias do problema, assim avaliando a eficácia e eficiência dos algoritmos (Beiranvand; Hare; Lucet, 2017).

3 METODOLOGIA

Neste capítulo serão apresentados os métodos utilizados para o desenvolvimento da pesquisa, tais como a escolha da metodologia, seleção de algoritmos, bases de dados, configuração de ambiente de testes e implementação do protótipo final.

3.1 Métodos Utilizados

O trabalho busca comparar meta-heurísticas amplamente utilizadas no Problema de Roteamento de Veículos com Janela de Tempo (VRPTW), analisando dados obtidos por meio de testes de desempenho sobre os algoritmos *Ant Colony Optimization* (ACO), *Genetic Algorithm* (GA) e *Tabu Search* (TS). Portanto, a pesquisa pode ser classificada de natureza primária, já que este tipo de pesquisa busca apresentar novos conhecimentos, utilizando para isso observações e criação de teorias para explicar os dados obtidos (Wazlawick, 2009).

De acordo com Wazlawick (2009, p. 43), ao classificar as pesquisas quanto ao objetivo, ele afirma: "A pesquisa explicativa é a mais complexa e completa. É a pesquisa científica por excelência porque, além de analisar os dados observados, busca suas causas e explicações, ou seja, os fatores determinantes desses dados.” Com isso, pode-se notar que a pesquisa possui também caráter explicativo, buscando descrever tanto o comportamento dos algoritmos quanto explicar quais os fatores que contribuem para os resultados de desempenhos encontrados.

Os resultados foram analisados com base em métricas específicas do problema, permitindo avaliar a qualidade dos resultados obtidos em relação a resultados do estado da arte obtidos via site do *Capacitated Vehicle Routing Problem Library* (CVRPLIB) (CVRPLIB [...], 2025), além da análise de tempo de execução necessário.

A pesquisa também pode ser classificada pelo uso de procedimentos experimentais, com uma abordagem quantitativa, já que as principais características da pesquisa experimental giram em volta da manipulação de aspectos da realidade, utilizando de variáveis experimentais controladas pelo pesquisador e utilizando de técnicas rigorosas de amostragem e testes de hipóteses, o que contribui para que o trabalho obtenha resultados estatisticamente aceitáveis e replicáveis (Wazlawick,

2009). Os experimentos buscam medir variáveis de desempenho dos algoritmos sob condições e restrições logísticas, coletando os dados e analisando os resultados.

O Quadro 1 apresenta de forma resumida a classificação da metodologia do trabalho, que com esses métodos a pesquisa é formulada visando a exploração de soluções para o VRPTW.

Quadro 1 - Classificação da metodologia do trabalho

Quanto à/ao	**Classificação**
Natureza	Primária
Objetivos	Explicativa
Abordagem	Quantitativa
Procedimentos	Experimental

Fonte: Autoria própria.

Os algoritmos são testados em ambientes simulados, utilizando *benchmarks* relevantes da variante VRPTW de Solomon (1987), executados múltiplas vezes para se obter uma representação mais realista e completa, principalmente em técnicas meta-heurísticas, que contam com fatores de aleatoriedade.

Por fim, o trabalho também explora métodos para implementação desses algoritmos em contextos reais, demonstrando tecnologias que podem ser usadas para se desenvolver sistemas que resolvam problemas reais.

3.2 Ferramentas Utilizadas

Para o ambiente de desenvolvimento é utilizado o Python como linguagem de programação, que é escolhido devido ao seu ecossistema de bibliotecas que facilitam o desenvolvimento de algoritmos de otimização, análise de desempenho e visualização dos resultados. As bibliotecas utilizadas para desenvolvimento são:

- NumPy e Pandas: Utilizadas para a manipulação e análise dos dados e resultados, processamento e preparação dos dados gerados pelos algoritmos, com o primeiro sendo ainda utilizado no desenvolvimento dos algoritmos em si.
- NetworkX: Biblioteca utilizada para a criação e manipulação de grafos, usada para representar os mapas e resolver problemas relacionados a grafos.
- Matplotlib: Ferramenta de visualização de dados que permitem a geração de gráficos e plotagens, facilitando a análise visual dos resultados dos algoritmos.

- time e timeit: Módulos padrão do Python utilizados para medir o tempo de execução de blocos de código.
- Numba: Utilizado para otimizar seções de código que são muito utilizadas pelos algoritmos, como função de cálculo de desempenho de soluções encontradas.

Para o contexto do protótipo de roteamento, uma parte fundamental se trata da representação, para isso é necessário utilizar-se de bases de dados reais que contenham o mapeamento do planeta. Algumas dessas plataformas são colaborativas e possibilitam o auxílio dos usuários na construção dos mapas, que são as Informações Geográficas Voluntárias (GIS - Geographic Information System).

O OpenStreetMaps (OSM), fundado em 2004, destaca-se como uma dessas plataformas significantes na área de GIS na última década, por conseguir construir um ecossistema com dados úteis e documentados por meio de páginas web, o que facilita sua usabilidade e aplicação (Fonte *et al*., 2017).

Essas fontes de dados são de grande utilidade, pois a sua licença permite o uso, modificação e construção, com a necessidade de citação ao OSM e seus contribuintes, além do compartilhamento com a mesma licença em caso de modificação dados (Minghini; Frassinelli, 2019). O formato de dados disponibilizado pelo OSM por padrão é o formato XML, que podem ser manipulados com ferramentas, as quais são listadas na página da documentação oficial do OSM (OpenStreetMaps, 2025, Fonte *et al*., 2017).

Para facilitar a extração dos dados do OSM, são desenvolvidas tecnologias como o Overpass Turbo, que se trata de uma ferramenta baseada em web que permite a visualização dos dados do OSM, para isso faz o uso de *queries* por meio da linguagem Overpass *Query Language*. Esta ferramenta possibilita a extração de dados de forma mais delimitada e com maior flexibilidade, selecionando partes do OSM que sejam de interesse (Fonte *et al*., 2017; OpenStreetMaps, 2024; Minghini; Frassinelli, 2019).

3.3 Ambiente de Testes

Na pesquisa de revisão de datasets para problemas de roteamento de veículos, Gunawan *et al*. (2021) destaca a plataforma *Vehicle Routing Problem Repository* (VRP-REP), que consiste em um web site que comporta *datasets* de testes para VRP. Na data de sua apresentação Gunawan *et al*. (2021) destaca que a plataforma contava com bases de dados para 41 variantes do VRP, sendo que as

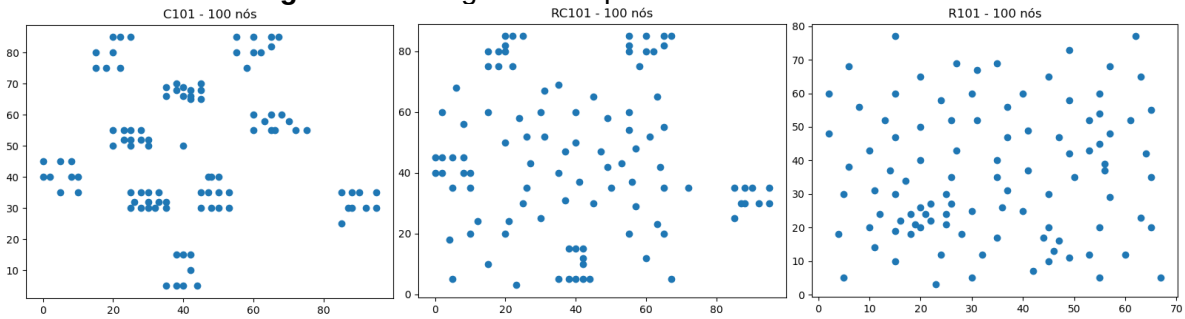
variantes que contavam com mais instâncias eram *Green Vehicle Routing Problem* (GVRP) com 1,045 instâncias e *Multi-Depot Vehicle Routing Problem with Time Windows* (MDVRPTW) com 959 instâncias (Gunawan *et al*., 2021). Sendo assim, a utilização dessa plataforma torna-se muito útil para pesquisas que buscam trabalhar com o problema de roteamento de veículos.

Com o objetivo de avaliar e comparar os algoritmos de roteamento, é importante utilizar bases de dados com cenários reais de aplicação, para isso utiliza-se de *datasets* reconhecidos pela comunidade. Para a presente pesquisa é utilizada a fonte de dados do site VRP-REP (VEHICLE ROUTING PROBLEM REPOSITORY, 2014), com os *datasets* de Solomon (1987) que contém problemas semelhantes ao selecionado pelo trabalho, além de serem datasets consolidados e muito utilizados pela literatura.

Os datasets selecionados de Solomon (1987) contém 56 instâncias que variam entre 3 classes, plotadas para ilustração da dispersão das classes na Figura 2:

- C (agrupadas): contém uma distribuição dos consumidores de forma agrupada, formando *clusters.*
- R (randômicas): consiste em uma distribuição uniforme de forma aleatória dos consumidores.
- RC (semi-agrupadas): uma forma combinada das instâncias anteriores.

Figura 2 - Plotagem da dispersão dos nós de instâncias



Fonte: *Gerado a partir das instâncias de Solomon (1987).*

De acordo com o que é apresentado no Quadro 2, para cada classificação mencionada existem 2 variantes. A primeira variante (R1, C1, RC1) contém janelas de tempo mais curtas, que juntamente com as restrições de capacidade fazem com que seja necessário a utilização de mais veículos. A segunda variante (R2, C2, RC2) contém maiores janelas de tempo e grande capacidades para os veículos, com isso

possibilitando que cada veículo atenda a mais clientes ao mesmo tempo (Solomon, 1987). Cada variante pode ser usada em arranjos menores, como 25, 50 ou 75 clientes, para reduzir a quantidade de resultados, serão utilizadas apenas a quantidade padrão de 100.

Quadro 2 - Informações sobre instâncias

Variante	Distribuição	Janela de tempo	Capacidade	Veículo percorre
C1	Nós aglomerados	Reduzida	Menor	Rotas menores
C2	Nós aglomerados	Ampla	Maior	Rotas maiores
R1	Nós dispersos	Reduzida	Menor	Rotas menores
R2	Nós dispersos	Ampla	Maior	Rotas maiores
RC1	Mistura entre C e R	Reduzida	Menor	Rotas menores
RC2	Mistura entre C e R	Ampla	Maior	Rotas maiores

Fonte: Baseado nas descrições de (Solomon, 1987).

Cada instância de teste contém um único depósito (definido pelo nó 0) e N clientes (representado pelos nós 1 a N), cada um desses com uma coordenada associada. Existem N requisições de clientes das quais possuem uma janela de tempo (início e fim) que deve ser respeitada, caso o veículo chegue muito cedo, deve esperar até a janela inicial de tempo.

Outra variável definida é a capacidade necessária para transportar o pacote e o tempo de serviço necessário para finalizar o serviço. A frota de veículos para esse problema é homogênea, definida com N veículos idênticos, que partem do único depósito existente e precisam finalizar sua rota retornando ao depósito, com uma capacidade de transportar uma quantidade Q de carga e um tempo máximo de percurso permitido para realizar as entregas.

Existem ainda os datasets de Gehring e Homberger (1999) que são uma extensão dos trabalhos de Solomon, aumentando a quantidade de consumidores a serem atendidos, com instâncias que contém 200, 400, 600, 800 e 1000 consumidores por instância, seguindo as subdivisões já citadas dos datasets de Solomon: C1, C2, R1, R2, RC1, RC2 (Gehring; Homberger, 1999).

O ambiente de teste é preparado utilizando as instâncias que buscam simular condições reais e replicar a complexidade dos problemas encontrados nas soluções

logísticas. As soluções serão implementadas em Python, utilizando bibliotecas já citadas. As simulações dos testes serão executadas em um computador com processador Intel-12400F possuindo 12 cores e 4.40 GHz, 16GB de memória, 480 GB de armazenamento de SSD e sistema operacional Linux Mint 21.3, além do Python 3.12.7.

Os testes serão executados paralelamente utilizando a biblioteca *multiprocessing* do python, onde após a extração dos dados dos *datasets*, se abre um processo para cada uma classe do *dataset* para o algoritmo por vez, otimizando assim o tempo necessário para executar os testes. Os resultados gerados pelo algoritmo são avaliados, calculando a distância total, tempo total, quantidade de veículos, e tempo de execução total.

3.4 Implementação de Algoritmos

Os algoritmos utilizados neste trabalho foram selecionados com base na frequência de uso em pesquisas que buscam selecionar o VRPTW. Conforme apresentado na seção 2.8, pode-se notar uma maior frequência dos algoritmos Busca Tabu (*Tabu Search* - TS), Algoritmos genéticos (*Genetic Algorithm* - GA), Otimização por colônia de formigas (*Ant Colony Optimization* - ACO), Busca Adaptativa em Grande Vizinhança (*Adaptive Large Neighborhood Search* - ALNS) e Busca em Grande Vizinhança (*Large Neighborhood Search* - LNS).

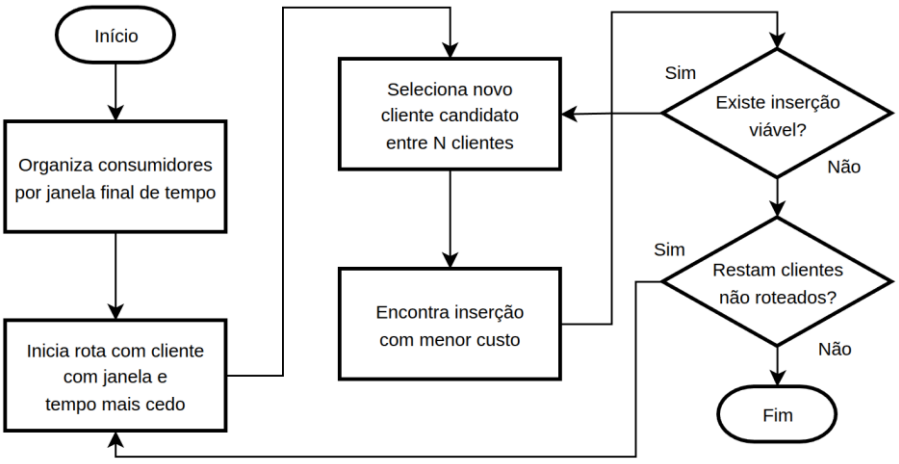
Nesta pesquisa os algoritmos ACO, GA e TS serão utilizados para testes e comparações. Todos esses iterando até atingir mesmos critérios parada, que são por número máximo de iterações $max = 100$ ou não ter melhoria por 10 iterações consecutivas.

3.4.1 Heurística Construtiva de Iniciação

A heurística de iniciação tem como objetivo fornecer uma solução inicial de boa qualidade, o que é de grande utilidade já que o problema possui um grande espaço amostral de soluções possíveis. O algoritmo escolhido para inicialização se baseia na heurística I1 de Solomon (1987), que constrói soluções sequencialmente a partir de uma cliente inicial, que é adicionado conforme o menor tempo inicial de serviço, e considerando critérios de custo para inserir cada cliente entre outros clientes i e j adjacentes da rota.

A Figura 3 ilustra o fluxo de execução da heurística construtiva I1 desenvolvida no trabalho, demonstrando os principais passos de sua execução.

Figura 3 - Ilustração do fluxograma da heurística construtiva I1



Fonte: Criado com base em (Solomon, 1987).

No fluxo da Figura 3, o algoritmos inicia as soluções a partir dos clientes que contém janela de tempo que se inicia mais cedo, ou seja, com menor valor, a partir daí é selecionado um novo candidato que ainda não foi roteado, procurando a inserção do cliente u com o menor custo utilizando:

$$d_{iuj} = d_{iu} + d_{uj} - \mu d_{ij} \qquad \mu \geq 0 \qquad (8)$$

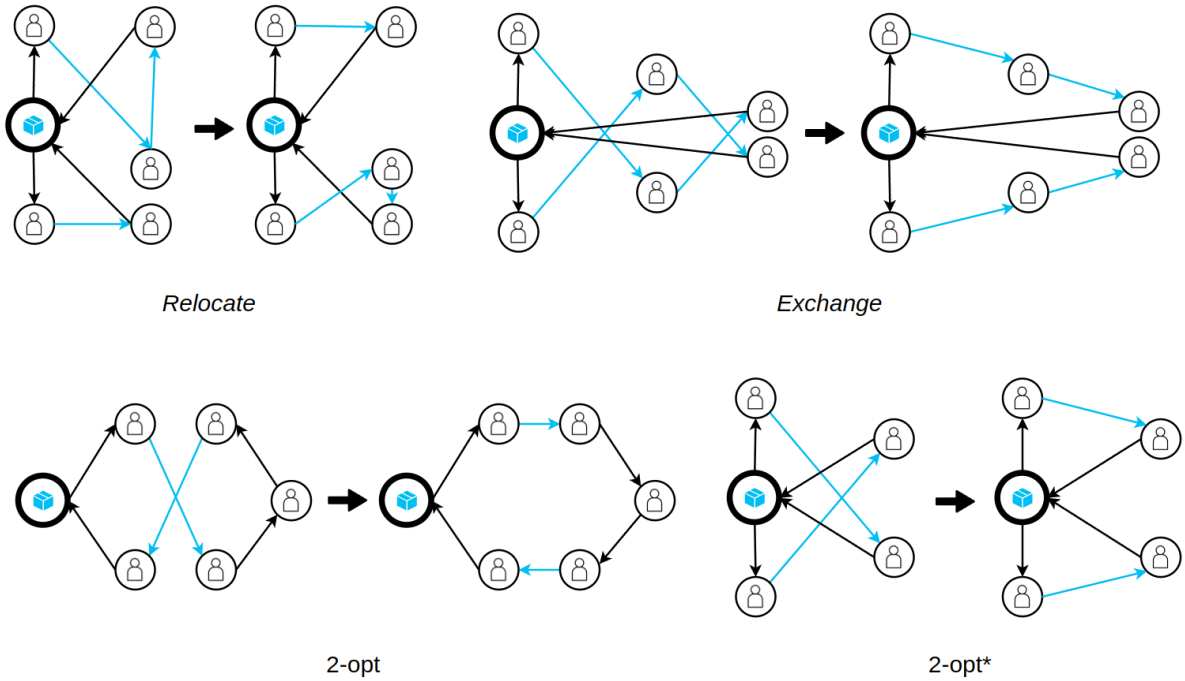
Ao encontrar os menores custos de inserção, é verificado quais não violam as restrições, e é selecionado a melhor inserção. Caso o algoritmo não encontre novas inserções viáveis, ele cria uma nova e repete o processo. A implementação utilizada neste trabalho explora uma quantidade N de clientes ao invés de explorar todos que ainda não foram roteados, sendo definida como 20% do número total de clientes, utiliza-se desse critério para otimizar o tempo de execução do algoritmo.

3.4.2 Fase de Busca Local

Os algoritmos implementados utilizam de uma fase de busca local, que é feita por meio dos operadores de busca em vizinhança, para a implementação foram utilizados os métodos listados a seguir, com exemplos das trocas ilustrados graficamente na Figura 4:

- Operador de realocação: Removendo um cliente de uma rota e inserindo entre dois clientes de outra rota.
- *Exchange*: Realize a troca de dois clientes entre duas rotas diferentes.
- 2-opt: Realiza a troca de arestas entre 4 clientes dentro da mesma rota.
- 2-opt*: Realiza a troca de arestas entre 2 clientes de duas rotas diferentes.

Figura 4 - Operadores de busca local



Fonte: *Criado com base em (Toth; Vigo, 2014, Bräysy; Gendreau, 2005).*

Por meio desses operadores de troca, são exploradas soluções próximas de uma solução inicial, pela exploração de soluções vizinhas. Esse passo consegue refinar ainda mais uma solução encontrada, mas possui um custo de desempenho a ser considerado, já que realiza todas as trocas possíveis dentro das regras de cada operador.

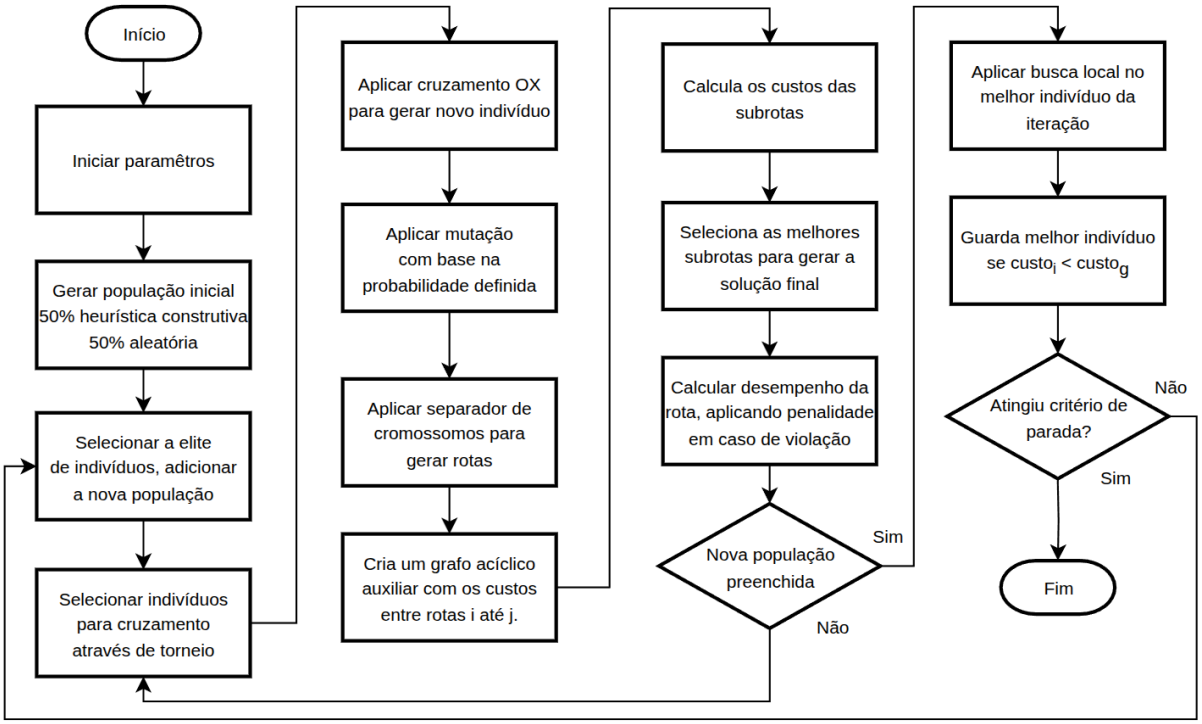
Para reduzir esse custo, são armazenadas os tempos finais previstos para cada rota da solução inicial, para evitar rotas que não sejam viáveis, avaliando se a janela de tempo no ponto de troca não é violada, assim não processando uma troca inviável, e reduzindo o processamento necessário. Além disso, cada operador tem sua execução finalizada após encontrar uma melhoria na solução, seguindo para o próximo de forma encadeada, na ordem apresentada na Figura 4.

Os operadores de busca local são usados em dois contextos diferentes, para melhoria de soluções locais e na implementação da busca tabu, já que a exploração desse método avalia soluções vizinhas, para ambos se utilizam os mesmos operadores de busca local, porém com a vizinhança implementada na Busca Tabu não encerrando ao encontrar melhoria, com o desempenho de rota calculado posteriormente.

3.4.3 Algoritmo Híbrido Genético

O Algoritmo Genético (GA) possui base em população e evolução das soluções por meio da seleção dos melhores indivíduos da população, realizando o cruzamento e mutação para obter novos indivíduos. Para o problema do VRPTW, cada cromossomo representa uma solução completa de todas as rotas sequenciadas, utilizando a estratégia de cromossomo *giant tour* definida por Prins (2004), onde se considera a solução por completo para realizar as operações, e a partir disso outros indivíduos são criados e construídos para avaliação de seu desempenho. A lógica completa está apresentada na Figura 5 em forma de fluxograma.

Figura 5 - Ilustração do fluxograma do GA



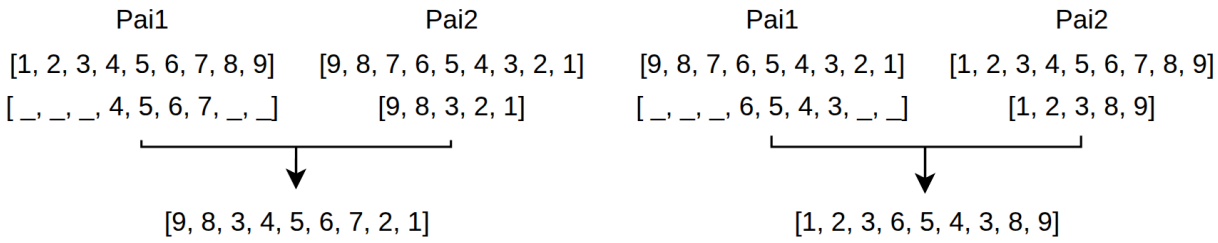
Fonte: Autoria própria.

O algoritmo genético inicia sua população composta por 50% de indivíduos criados com a heurística construtiva I1 de Solomon e os outros 50% de forma aleatória, aplicando o operador de separação. Seguindo, primeiro calcula-se o custo de cada indivíduo, em seguida ordenando os mesmos de acordo com o custo. A partir daí, é selecionada a elite, que tem seu tamanho definido $e = 2$, com essa parcela de indivíduos sendo adicionados à nova geração.

Para preencher o restante do espaço da população é realizado o torneio de seleção, onde a cada rodada que indivíduos são escolhidos aleatoriamente, e de acordo com a probabilidade de cruzamento definida por $c = 0.8$. Dois desses indivíduos são selecionados e aplicados no operador de cruzamento por ordem.

O cruzamento extrai um segmento de uma sequência Pai1[1], remove os genes do segmento extraído do Pai2[2], e preenche o restante da rota mantendo a ordem relativa dos genes do Pai2, preservando assim partes importantes das rotas. A Figura 6 ilustra esse processo, com o ponto de corte $i = 3, j = 6$.

Figura 6 - ilustração do cruzamento por ordem

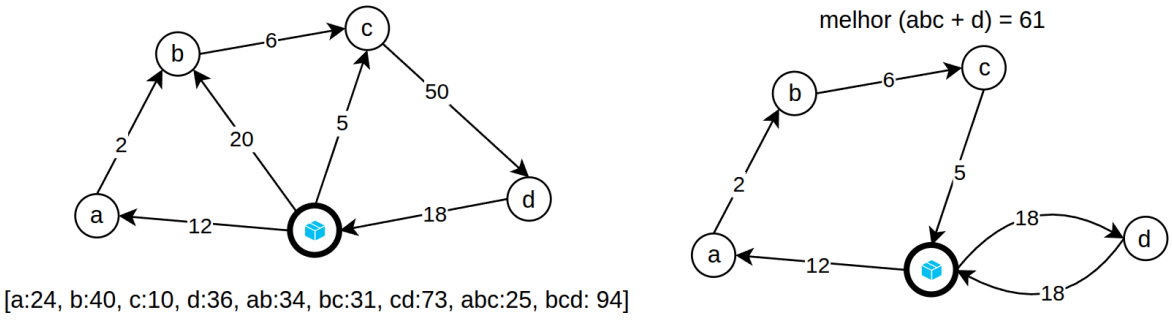


Fonte: Criado com base em (Toth; Vigo, 2014).

Após o cruzamento, aplica-se uma mutação aleatória com cada descendente com a probabilidade de $m = 0.15$, efetuando a troca de depois genes do cromossomo, promovendo diversidade genotípica. Em seguida, cada novo cromossomo criado é submetido ao processo de divisão baseado no procedimento de separação de Prins (2004) que é adaptado para o problema considerando as restrições de janela de tempo, o algoritmo reconstrói as rotas com base em um grafo acíclico auxiliar, que contém sub-rotas viáveis e seus custos associados, a partir deste grafo é selecionado conjunto de sub-rotas que respeitem as restrições do problema e tenha o menor custo, a seguir a Figura 7 demonstra de forma simplificada o processo de separação.

[1] Pai1: Cromossomo principal que tem seu segmento reduzido.
[2] Pai2: Cromossomo secundário com elementos semelhantes excluídos e aplicados no Pai1.

Figura 7 - ilustração de processo de separação de cromossomo



[a:24, b:40, c:10, d:36, ab:34, bc:31, cd:73, abc:25, bcd: 94]

Fonte: Baseado em (Prins, 2004).

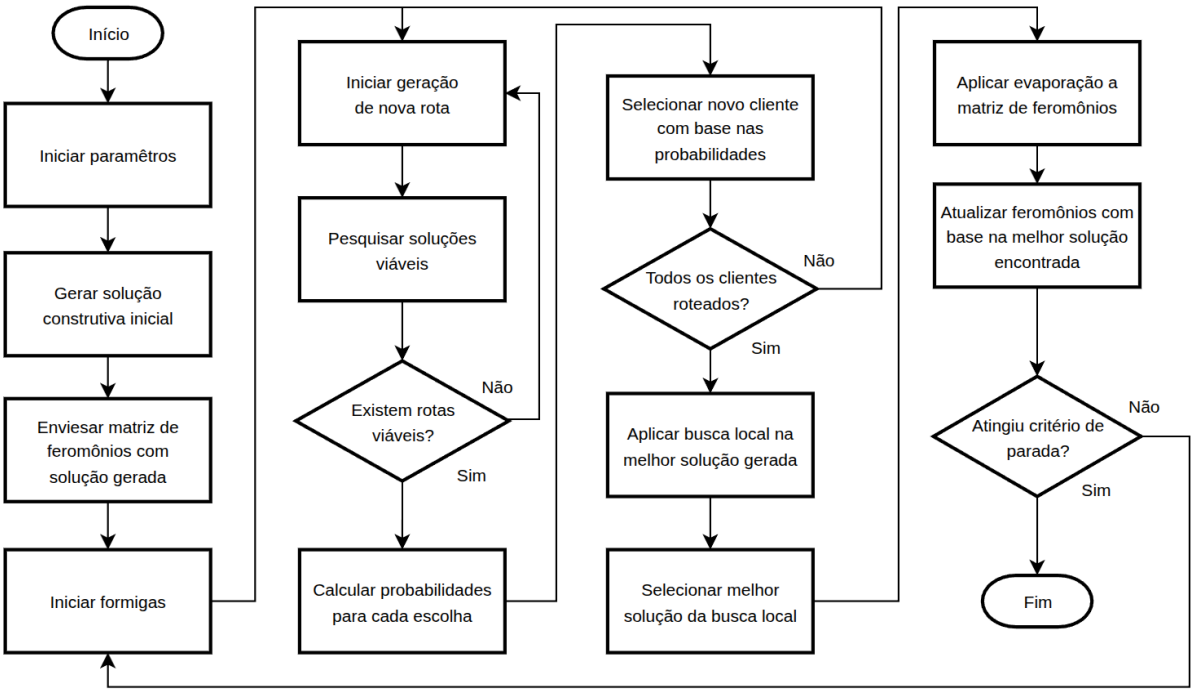
O algoritmo realiza o cálculo dos custos de cada sub-rota a, b, c, d, ab, bc, cd, abc e bcd. A partir daí é selecionado a melhor combinação que gere o menor custo. Considerando também que algumas das sub-rotas podem não respeitar as restrições do problema, e, portanto, podem ser eliminadas.

Por fim, a população resultante é então reavaliada com base no custo total de cada indivíduo, ordenando a lista e selecionando o melhor indivíduo da geração, para então ser submetido a busca local já detalhada anteriormente, para encontrar melhorias no melhor indivíduo, que então é comparado com a melhor solução global, e caso haja melhora, é armazenado. O algoritmo é iterado até atingir algum critério de parada. O algoritmo utiliza $p = 15, e = 2, i = 100, m = 0.15, c = 0.8$.

3.4.4 Algoritmo de Otimização por Colônia de Formigas Híbrido

O algoritmo de Otimização por Colônia de Formigas (ACO) possui sua característica construtiva, guiada por meio da distribuição de feromônios das formigas artificiais e heurística, o fluxo do algoritmo desenvolvido é apresentado na Figura 8.

Figura 8 - Ilustração do fluxograma do ACO



Fonte: Autoria própria.

O algoritmo ACO inicia gerando uma matriz de feromônios de dimensão igual ao espaço amostral do problema, inicialmente preenchida com o valor padrão $\tau_0 = 1$. Em seguida, essa matriz é enviesada por uma solução inicial obtida pela heurística construtiva I1 de Solomon, para cada aresta percorrida nessa rota inicial, aumenta-se o feromônio correspondente segundo a fórmula (4) da seção 2.10.3, reforçando trajetos promissores antes das iterações principais serem iniciadas.

As formigas são então inicias formigas e constroem soluções factíveis. A cada passo de construção de rota, cada formiga escolhe o próximo cliente j a partir de *i* com base nas probabilidades definidas na fórmula (3) da seção 2.10.3, que então são calculadas com τ_{ij} sendo o nível de feromônio e η_{ij} a heurística baseada em distância, que, no caso do problema, é calculada por meio da distância euclidiana. O expoente α controla a influência do feromônio e β a influência da heurística.

Após todas as formigas completarem suas rotas, respeitando restrições de capacidade de carga e janelas de tempo, aplica-se uma busca local descrita anteriormente na melhor solução com base no critério de menor distância, aprimorando o resultado encontrado. Após a busca local, é feita a atualização da matriz de feromônios, com base no melhor resultado encontrado, primeiro ocorrendo a evaporação global dos feromônios, após a evaporação, a matriz é atualizada com

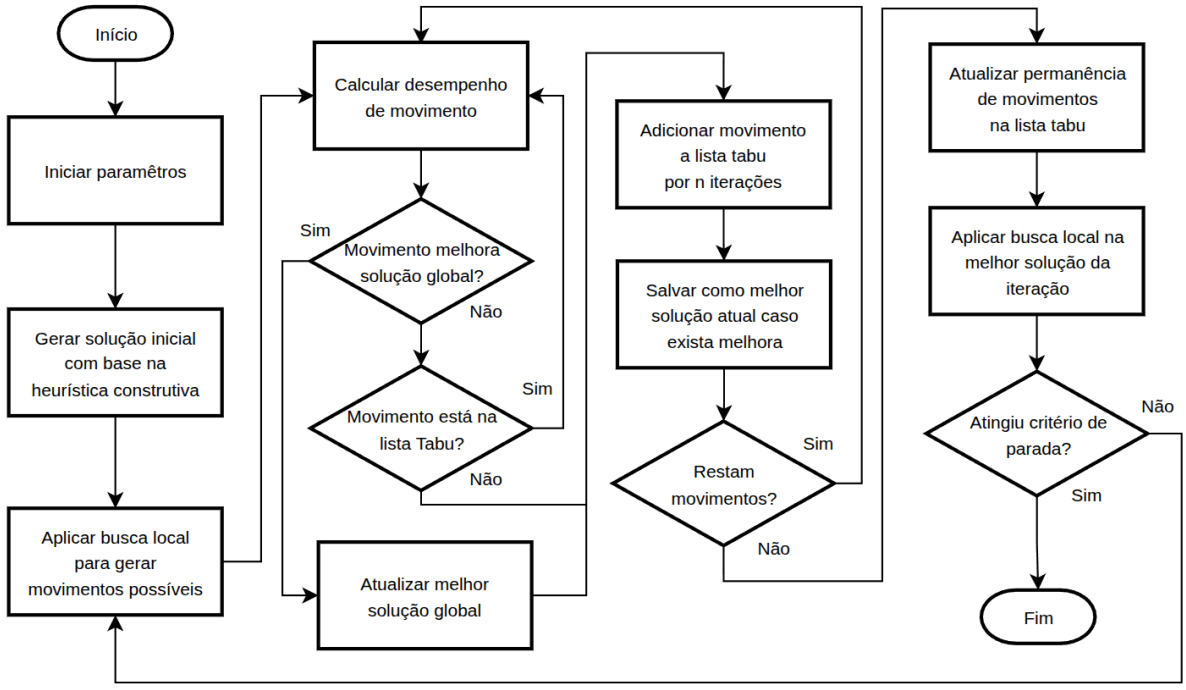
base nos conceitos da equação já citada anteriormente, depositando o feromônio extra apenas nos arcos usados pelas melhores formigas, reforçando boas soluções.

Por fim, o ciclo se repete até atingir o critério de parada, construindo soluções e atualizando o feromônio de acordo com os melhores resultados, dessa forma a informação acumulada na matriz guia as formigas a encontrar regiões mais promissoras do espaço de busca. Os parâmetros utilizados para o algoritmo são $\alpha = 1.5, \beta = 2.5, i = 100, p = 30$.

3.4.5 Algoritmo de Busca Tabu Híbrido

A Busca Tabu (TS) possui sua tabela de restrições garantindo que soluções previamente visitadas sejam proibidas por um período de iterações, assim possibilitando que o algoritmo escape de ótimos locais. A busca de novas soluções é feita por meio da busca local. A demonstração do fluxo de execução do algoritmo é apresentada na Figura 9 a seguir.

Figura 9 - Ilustração do fluxograma do TS



Fonte: Autoria própria.

O algoritmo de Busca Tabu inicia-se a partir de uma solução inicial gerada pela heurística construtiva I1 de Solomon. Em seguida, executa-se um processo iterativo de busca local utilizando os operadores definidos anteriormente, retornando

todos os movimentos de troca viáveis, que são então listados e avaliados de acordo com a função de custo total de distância.

Para cada um movimento candidato de cada operador, é verificado primeiro se o movimento é aceito pelo critério de aspiração. Se o movimento resultar na melhora da solução global encontrada até o momento, é aceito, mesmo que esteja na lista tabu. Caso contrário, é verificado se o movimento está na lista tabu, se estiver, é ignorado e o algoritmo segue para avaliar o próximo movimento. Se o movimento não estiver na lista tabu, é então avaliado. Se o movimento resultar na melhora da solução atual, é aceito e salvo como melhor solução para essa iteração.

Ao final de cada iteração, seleciona-se a melhor solução encontrada para esse ciclo, entre as soluções resultantes de trocas não pertencentes a lista tabu ou as aspirações. A lista tabu é então atualizada, reduzindo o fator de permanência de soluções passadas e removendo movimentos de acordo com o tamanho do tabu. E ao fim, a melhor solução do ciclo é aplicada na busca local, sendo salvo caso melhore o resultado global. Assim o algoritmo garante uma melhor exploração e evitando retornos a soluções recentemente visitadas.

O procedimento de geração de vizinhança é novamente aplicado em uma nova iteração e avaliado de acordo com a lista tabu. O algoritmo se repete até atingir o critério de parada. Com os parâmetros de $i = 100, a = 5$.

3.5 Análise De Desempenho

Os testes são propostos com o intuito de avaliar a eficiência dos algoritmos selecionados para resolver o VRPTW, identificando qual algoritmo apresenta melhor desempenho. Para garantir a consistência, cada algoritmo será executado cinco vezes para assegurar a consistência dos resultados e minimizar a influência de fatores aleatórios. Para avaliar o desempenho dos algoritmos, são determinadas métricas, descritas a seguir:

- Distância total;
- Tempo total das rotas;
- Quantidade de veículos utilizados;
- Tempo de execução;
- Qualidade da solução;

Os algoritmos serão executados com uma quantidade máxima de 100 iterações ou até convergirem para uma solução estável que se repita de forma sequencial durante 10 iterações, estas limitações são impostas para não inviabilizar os testes caso o algoritmo seja se torne muito lento, o que é importante de ser considerado por se tratar de um problema np-completo.

Após a etapa da execução dos testes, os dados são tabulados e analisados por meio de métodos estatísticos e amostrais, com um dos métodos mais completos de gerar relatórios para resultados de *benchmarks* se tratando das tabelas numéricas, mas que podem se tornar extensivas para grandes quantidades de resultados (Beiranvand; Hare; Lucet, 2017). Para esses casos, se utilizam de outras técnicas para facilitar o entendimento, utilizando da mediana dos valores obtidos para cada *dataset* e comparação com resultados do estado da arte.

3.6 Desenvolvimento do Protótipo de Roteamento

Para meios demonstrativos da aplicabilidade prática dos algoritmos em aplicações reais, desenvolveu-se um exemplo da aplicabilidade do OSM em conjunto com outras tecnologias. O protótipo de software de roteamento de veículos é implementado utilizando algoritmos desenvolvidos e selecionando a melhor solução após a análise dos resultados encontrados nos experimentos. Para o desenvolvimento do protótipo são utilizadas tecnologias e ferramentas de código aberto, para se criar uma solução de forma mais prática.

Como fonte de dados geográficos foi utilizado o OpenStreetMaps, que possibilita a extração de dados de forma simples por meio de suas ferramentas. Para o trabalho, são extraídos dados para exemplificar a usabilidade na plataforma *web* Overpass-Turbo, já que essa aplicação facilita a seleção de partes do mapa de forma simples com sua linguagem de *query*. São extraídos dos dados informações referentes a vias, e convertidas em formato GeoJSON, para que posteriormente possam ser utilizados no *software* de roteamento como dados de exemplo.

Para a visualização das rotas, é utilizado o Leaflet, por ser uma biblioteca de código aberto, que possui uma fácil integração com a base de dados do OpenStreetMaps (OSM). A combinação das duas ferramentas possibilita a visualização dinâmica aliado com uma interface moderna e simples.

Para a plataforma de desenvolvimento, é utilizada a web, por se tratar de um *software* para utilização multiplataforma, facilitando acesso para o usuário, já que não necessita de instalação. Para esse fim, é utilizado o *framework* Vite para o desenvolvimento do *software*, como responsável pela construção do *front-end* da plataforma, já que facilita a criação de interações fluídas para o usuário.

Para a comunicação entre pares, o formato de dados JavaScript Object Notation (JSON) é utilizado, já que, tem se tornado um formato de dados cada vez mais popular para a comunicação na internet, ultrapassando a popularidade de outros formatos como o XML (Fosci; Psaila, 2023).

Porém, para que seja possível definir dados mais específicos, como dados de georeferencia, é necessário que haja uma estrutura padrão e bem definida, para finalidades de uso em GIS, foi desenvolvido pela comunidade o GeoJSON com suas especificações definidas no RFC 7946, a ferramenta tem o intuito de fornecer uma estrutura padronizada que considera tipos de geometria para cada dimensão (Fosci; Psaila, 2023; Zhu; Tan, 2018). Os principais tipos de geometrias são apresentados no Quadro 3.

Quadro 3 - Tipos de geometria que o GeoJSON suporta

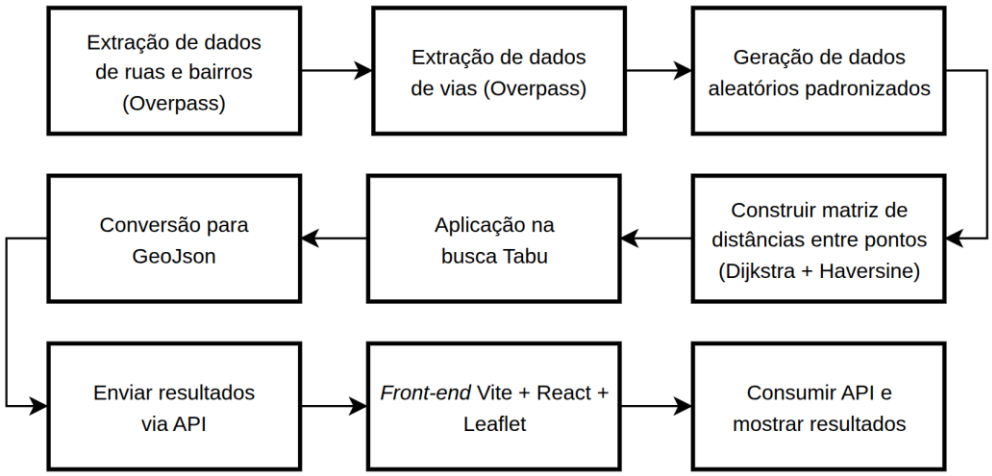
1 dimensão	**2 dimensões**	**3 dimensões**	**dimensões heterogêneas**
Point	*LineString*	*Polygon*	*GeometryCollection*
MultiPoint	*MultiLineString*	*MultiPolygon*	

Fonte: Criado a partir das descrições de (Horbiński; Lorek, 2022).

Com os elementos apresentados no Quadro 3 é possível criar pontos, linhas e polígonos que podem ser interpretados utilizando o Leaflet em conjunto com o GeoJSON, para criar mapas interativos (Horbiński; Lorek, 2022) que fornece um ambiente de desenvolvimento interativo e funcional para o usuário final.

A Figura 10 demonstra em forma de fluxograma os passos utilizados no trabalho para construção do protótipo de apresentação.

Figura 10 - Fluxograma de implementação de protótipo



Fonte: Autoria própria.

A extração dos dados é feita por meio da plataforma do Overpass-Turbo contendo as vias, ruas e bairros, exportado em forma GeoJSON, a partir daí são extraídos os dados principais, a partir dos dados são selecionados endereços de forma aleatória, selecionando uma via e selecionando uma coordenada aleatória dessa via, selecionando também o endereço relacionado a esse ponto. Assim, com dos dados gerados, são selecionadas as coordenadas de cada ponto, e aplica-se então o algoritmo Dijkstra para encontrar todas as rotas entre o ponto atual e os N outros pontos selecionados, utilizando para isso o cálculo de Haversine, para calcular a distância entre cada um dos pontos.

Em seguida, os dados pré-formatados são então aplicados no algoritmo no TS, que então gera os resultados do roteamento de menor custo encontrado. Na sequência os dados são convertidos para formato GeoJSON para que assim possam ser disponibilizados pela API, que é desenvolvida com o *framework* flask. Os dados são então consumidos pelo *Front-end*, que utiliza as tecnologias citadas, demonstrando os resultados obtidos na forma de *interface* interativa.

Assim, por meio desse conjunto de ferramentas é possível desenvolver as soluções de testes e *software* necessárias para este trabalho, dessa forma oferecendo o suporte necessário para atingir os objetivos propostos na pesquisa.

4 RESULTADOS E DISCUSSÕES

Nesta seção serão apresentados os resultados obtidos por meio dos testes descritos. Após a execução, os resultados dos critérios de avaliação são apresentados na Tabela 1. Os resultados foram calculados com base nos critérios descritos na seção 3.5, realizando a mediana dos valores por datasets para simplificar a amostragem, organizados por distância, tempo e veículos, com ACO, TS e GA representando os algoritmos Otimização Colônia de Formigas, Busca Tabu e Algoritmo Genético, respectivamente.

Tabela 1 – Mediana dos resultados obtidos nos benchmarks de Solomon

Algoritmo	Distância	Tempo	Veículos	*Dataset*
ACO	892,50	9903,67	10,20	C1
TS	1072,29	10559,55	10,80	C1
GA	1119,29	10447,67	10,80	C1
ACO	650,75	9752,78	3,30	C2
GA	699,24	11907,41	4,00	C2
TS	718,58	11723,37	4,00	C2
TS	1318,29	2502,84	13,80	R1
GA	1361,54	2531,34	13,60	R1
ACO	1420,79	2500,34	13,30	R1
TS	1078,08	3098,09	4,00	R2
GA	1088,13	3022,29	4,00	R2
ACO	1314,47	2855,12	4,00	R2
TS	1537,71	2773,57	14,80	RC1
GA	1608,48	2745,71	14,40	RC1
ACO	1626,97	2753,01	13,70	RC1
GA	1240,78	3314,27	4,60	RC2
TS	1317,82	3254,79	4,10	RC2
ACO	1492,85	3248,66	4,60	RC2

Fonte: *Autoria própria.*

Por meio dos dados da Tabela 1, pode-se notar que o algoritmo ACO apenas consegue ter algum destaque em instâncias aglomeradas (C1, C2), principalmente na instância com janelas de tempo reduzidas, que acabam por facilitar a construção das rotas para as formigas, já que acabam por “limitar” a quantidade de possibilidades a serem escolhidas pelas próximas formigas, já que após a escolha do primeiro cliente,

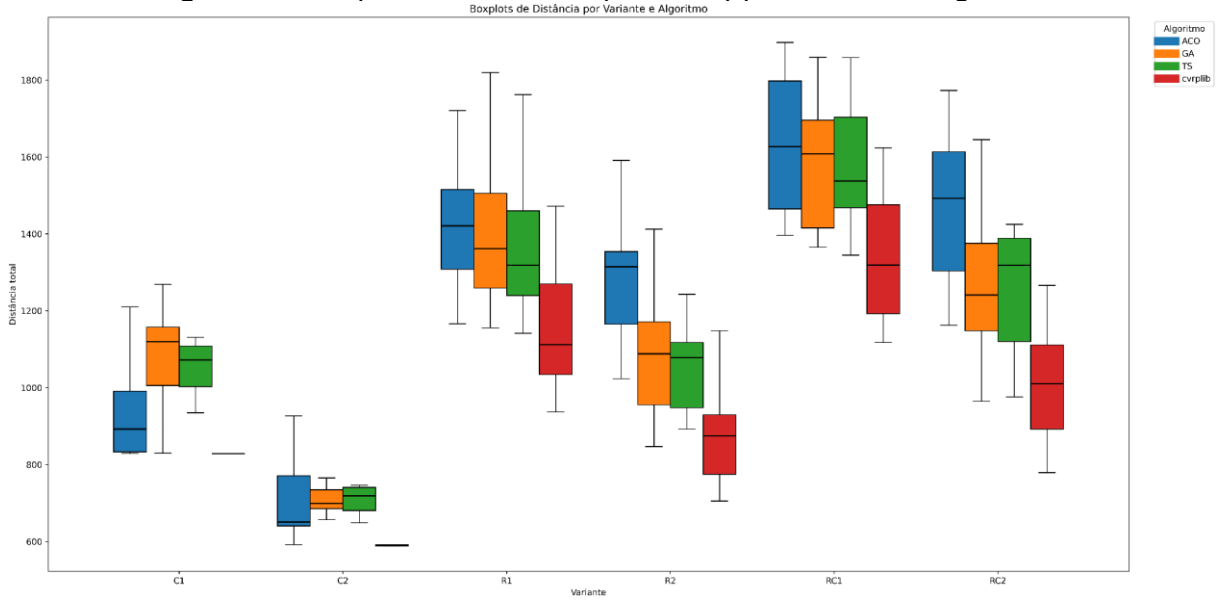
os demais que possuem um bom fator de heurística são os clientes daquela aglomeração, assim facilitando a incrementação e dos feromônios em soluções próximas.

Além disso, também percebeu-se que os algoritmos TS e GA possuem resultados, no geral, próximos, já que apesar das implementações apresentarem estratégias híbridas, ainda carecem de sofisticações modernas. Com isso, ambos os algoritmos acabam por apresentarem resultados semelhantes, já que o GA evita explorar soluções inviáveis, uma de suas principais vantagens, o que se comprova como uma característica importante para gerar boas soluções, mas que não foi explorada para padronizar o experimento entre os algoritmos.

Para uma avaliação comparativa com resultados do estado da arte, são utilizados os dados disponibilizados pelo site *Capacitated Vehicle Routing Problem Library* (CVRPLIB) (CVRPLIB [...], 2025), que possui resultados para as instâncias de Solomon, além das soluções listadas de cada resultado, facilitando assim a manipulação para comparação com os resultados do experimento.

Para uma melhor ilustração dos resultados obtidos de uma maneira geral, utilizou-se do *boxplot* na Figura 11 para demonstrar a variação e amplitude dos resultados obtidos em relação aos disponíveis no site do CVRPLIB.

Figura 11 - *Boxplot* de resultados (distância) por variante e algoritmo



Fonte: Autoria própria (utilizando dados do CVRPLIB).

A Figura 11 possibilita visualizar como as instâncias aglomeradas possuem resultados finais quase que idênticos para os resultados do CVRPLIB, e apesar do

ACO apresentar melhores soluções no geral melhores para essas instâncias, o TS consegue atingir resultados com uma menor variação geral, tendo mais consistência em seus resultados, o que se dá pela sua maior intensificação da busca local, se mantendo em soluções mais concisas, que também pode ser notado nos resultados do R2 e RC2.

Para avaliar a diferença percentual entre os resultados obtidos em relação aos valores de referência, que são apresentados na Tabela 2 com os valores de mediana da diferença entre os resultados obtidos e os resultados do CVRPLIB para cada *dataset*.

Tabela 2 - Mediana da diferença entre os resultados e CVRPLIB

Algoritmo	Distância %	Tempo total %	Veículos %	*Dataset*
ACO	7,67	0,76	2,00	C1
TS	30,01	7,43	8,00	C1
GA	35,03	6,30	8,00	C1
ACO	10,60	1,71	10,00	C2
GA	18,54	24,08	33,33	C2
TS	21,47	21,79	33,33	C2
TS	18,81	11,45	13,33	R1
GA	20,26	12,42	8,78	R1
ACO	24,28	11,65	9,55	R1
GA	23,01	-14,15	-25,00	R2
TS	23,79	-11,14	-20,00	R2
ACO	45,04	-11,10	-25,00	R2
TS	16,88	13,72	17,34	RC1
GA	17,74	13,31	17,42	RC1
ACO	22,34	12,36	13,33	RC1
GA	25,58	-11,66	-18,34	RC2
TS	30,24	-16,45	-29,16	RC2
ACO	45,23	-18,18	-23,34	RC2

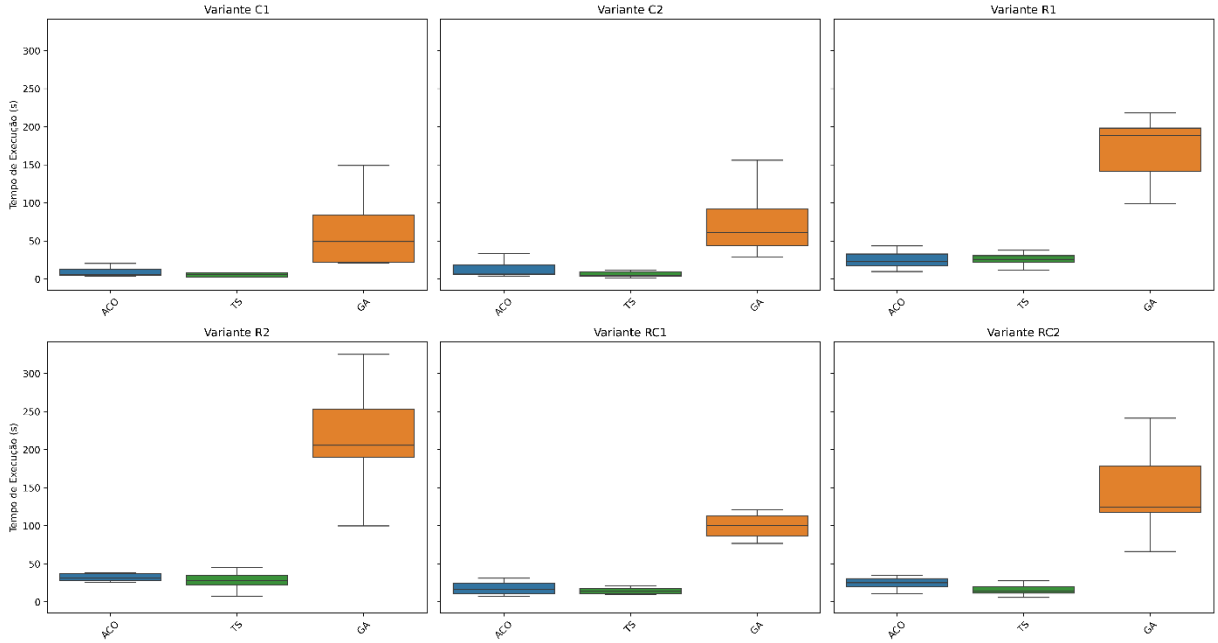
Fonte: Autoria própria.

Na análise da Tabela 2 é possível notar que os algoritmos conseguiram atingir resultados satisfatórios, considerando a implementação mais simples adotada para facilitar a comparação e diminuir o escopo final do trabalho, mais uma vez o destaque do ACO em instâncias de clientes aglomerados é vista, o que reforça ainda mais a hipótese já citada.

Outro ponto importante a se notar é que os resultados obtidos para as instâncias R2 e RC2 atingiram um número de veículos e tempo total percorrido menores que os resultados do estado da arte, isso acontece devido a métrica central dos algoritmos em diminuir a distância total, mesmo que se sacrifique outros pontos. Neste caso, os resultados com menores distâncias necessitam de maior número de veículos, o que acarreta também em um maior tempo total de percurso das rotas.

No geral os resultados obtidos para o TS e GA possuem valores semelhantes, apenas de existir uma menor distribuição dos resultados notada para o TS em relação ao GA. Porém, o mesmo não se permanece quando comparado os tempos de execução necessários para as duas estratégias. A Figura 12 apresenta os tempos de execução para cada um dos algoritmos.

Figura 12 - *Boxplot* de tempo de execução dos algoritmos



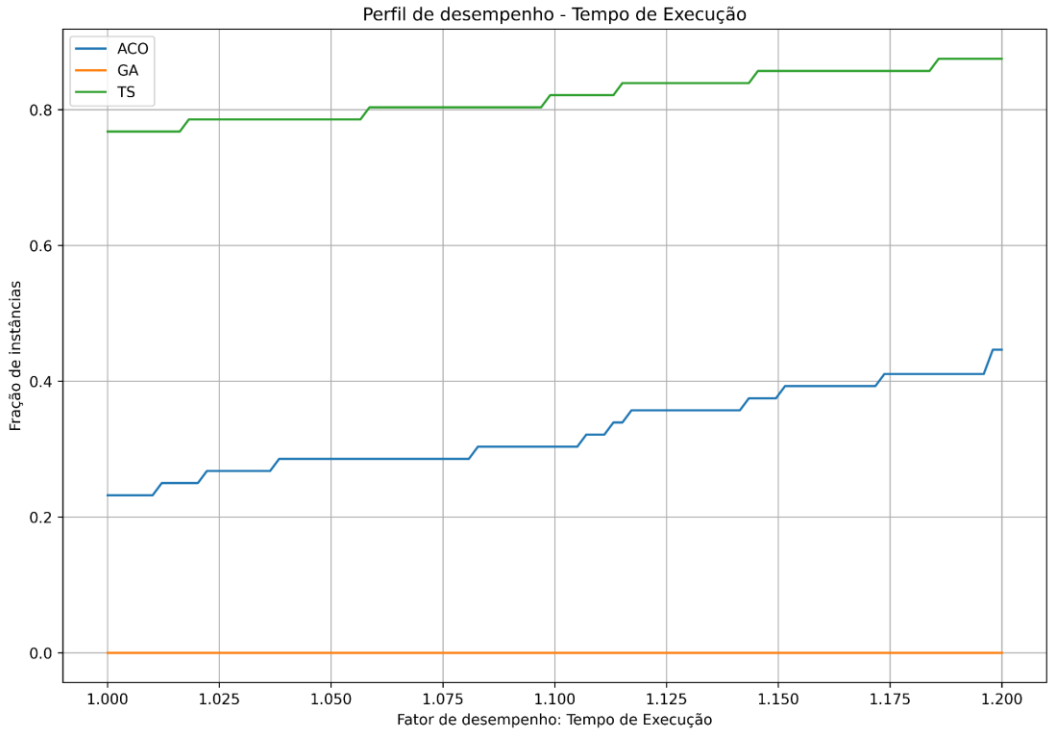
Fonte: Autoria própria.

Analisando a Figura 12, pode-se perceber que o TS possui uma rápida convergência para os resultados finais em comparação às demais estratégias, também pode-se notar um maior tempo de execução necessário para o GA até sobre o próprio ACO, isso se deve a necessidade de separação dos cromossomos de cada indivíduo, que se mostrou um processo custoso para a implementação utilizada no trabalho, além da população inicial que necessita ser inicializada para N indivíduos,

fazendo com que o algoritmo possua uma maior complexidade em relação aos demais analisados.

Isso gera uma maior diferença entre essas duas soluções promissoras, assim destacando o TS sobre o GA, devido seu menor tempo de execução necessário para gerar resultados semelhantes, para as estratégias utilizadas no presente trabalho. Por fim, para demonstrar uma maneira simplificada e mais ilustrativa, são apresentados os perfis de desempenho para os resultados obtidos, o que analisam de maneira geral entre as instâncias, assumindo como valor de referência o melhor dos resultados obtido dentre os candidatos. Os resultados são apresentados na Figura 13, considerando que quanto mais alto e mais à esquerda, melhor.

Figura 13 - Perfis de desempenho dos algoritmos utilizados em função do tempo total de execução

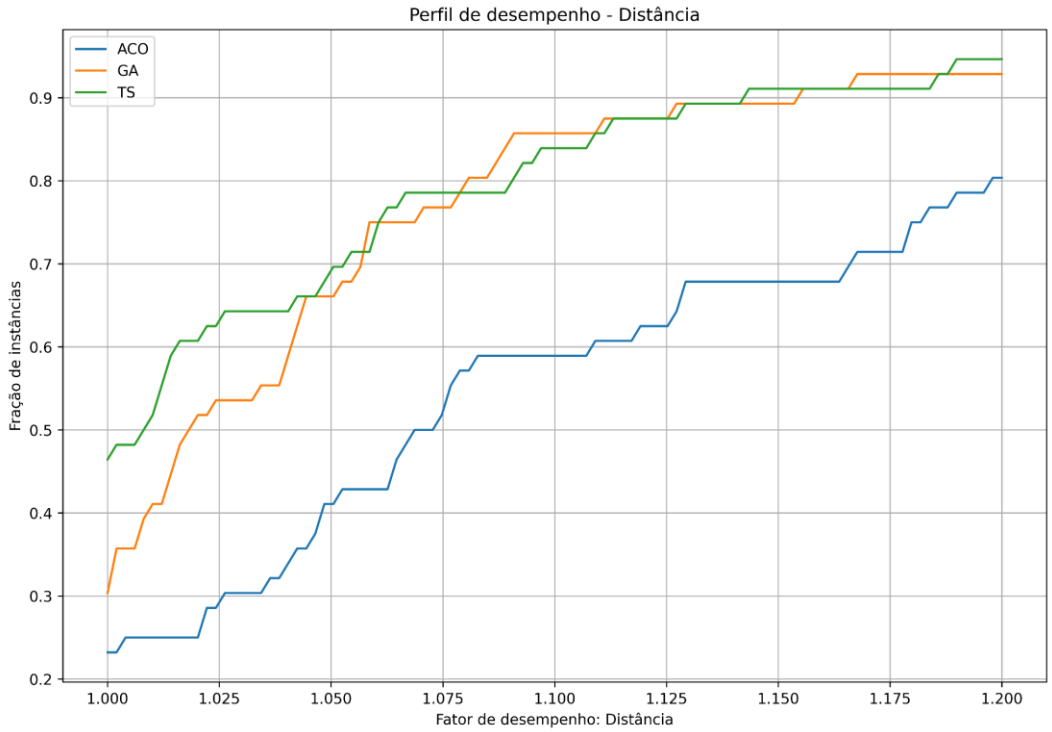


Fonte: Autoria própria.

A Figura 13 ilustra de forma simplificada os resultados obtidos quanto ao tempo de execução, com o TS apresentando os melhores resultados de forma muito clara, devido a sua estratégia simplista, mas que consegue gerar resultados satisfatórios, convergindo rapidamente para boas soluções, aprimorando a busca local por meio da sua lista de movimentos proibidos, o que ajuda a aumentar o espaço de exploração. Outro fato que se destaca é o baixo desempenho do GA, por possuir uma

estrutura mais complexa para a separação dos cromossomos e ter que processar uma população de indivíduos, acaba por precisar de uma maior quantidade de tempo de execução. Seguindo, é apresentado na Figura 14 os perfis de desempenho em função da distância total percorrida.

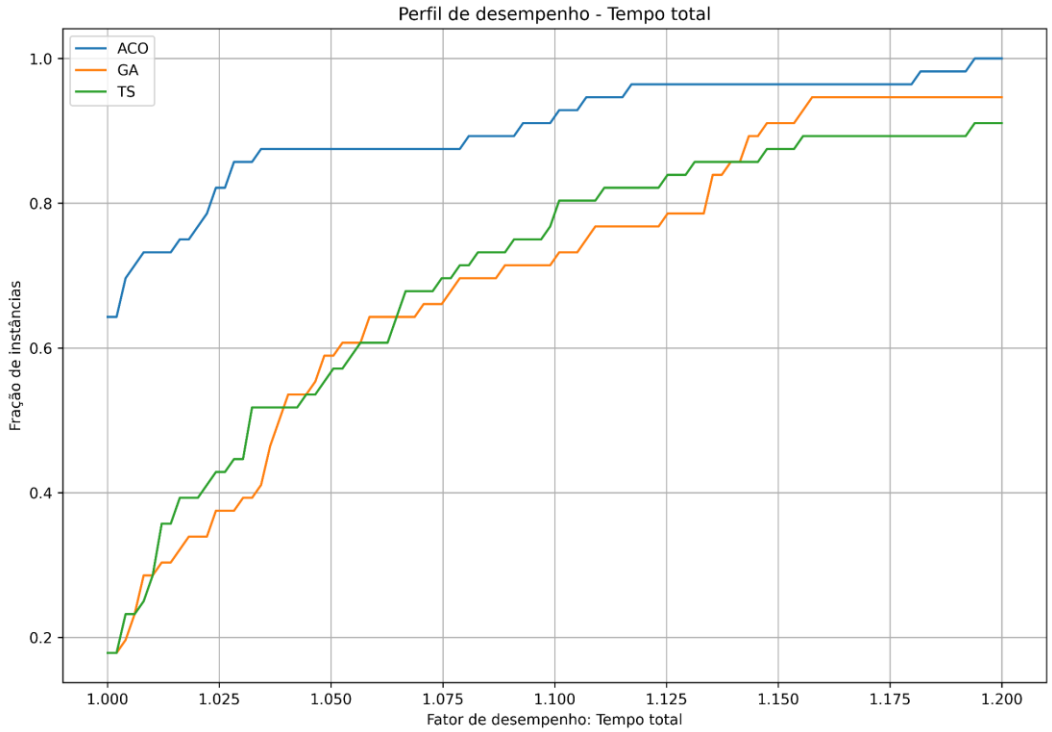
Figura 14 - Perfis de desempenho dos algoritmos utilizados em função da distância total percorrida



Fonte: Autoria própria.

Na Figura 14 é possível notar que os algoritmos GA e TS obtiveram resultados próximos, já o ACO teve um desempenho inferior aos demais, como visto anteriormente, com resultados superiores apenas para instâncias aglomeradas, o que resulta em um pior desempenho geral em relação aos demais. A Figura 15 apresenta os resultados de perfis de desempenho em relação ao tempo total de percurso percorrido por cada veículo.

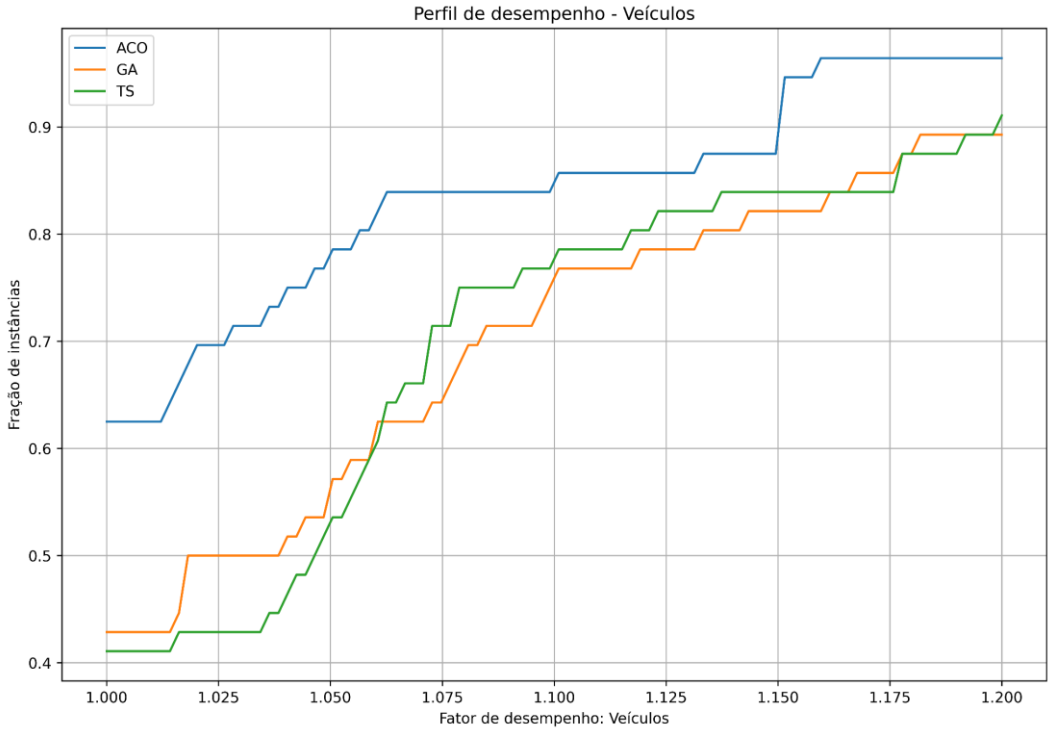
Figura 15 - Perfis de desempenho dos algoritmos utilizados em função do tempo total percorrido



Fonte: Autoria própria.

Neste caso, a Figura 15 apresenta um melhor desempenho para o ACO, isso ocorre pelos seus melhores resultados quanto aos números de veículos utilizados e consequentemente menores tempos total das rotas, já que um número menor de veículos utilizados, geralmente gera também menores tempos totais. Novamente os algoritmos GA e TS possuem resultados gerais semelhantes. Por fim, o último a Figura 16 apresenta os perfis de desempenho em função da métrica de número de veículos total utilizados por solução.

Figura 16 - Perfis de desempenho dos algoritmos utilizados em função número de veículos

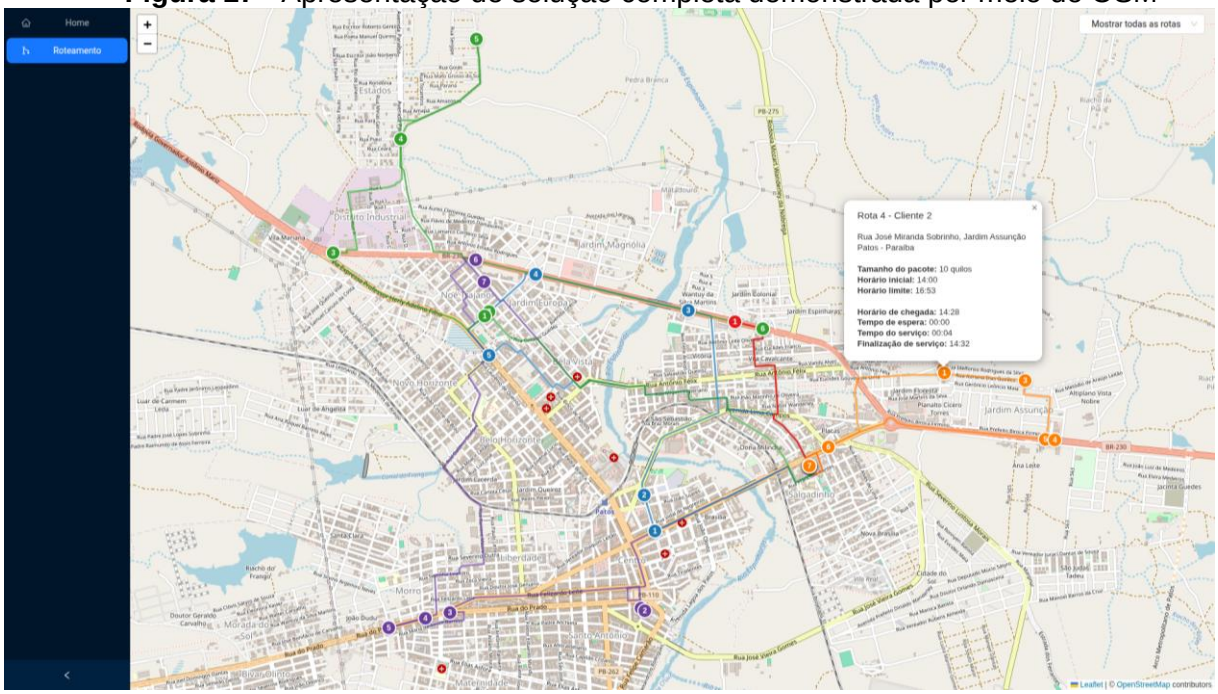


Fonte: Autoria própria.

Na Figura 16 o ACO demonstra um melhor resultado geral, por possuir mais flexibilidade para reduzir o número de veículos, devido à reconstrução das rotas pelas formigas em cada iteração, não mantendo soluções semelhantes como o TS e o GA, isso possibilita que o algoritmo reduza a quantidade de rotas totais ao decorrer da execução, diferentemente dos demais, que possuem uma menor chance de redução do número de rotas ao decorrer das iterações.

A seguir, para demonstrar de forma clara como os resultados obtidos pelos algoritmos desenvolvidos neste trabalho podem ser aplicados em cenários reais, e fornece uma melhor visualização das rotas geradas pelos algoritmo, utiliza-se de um protótipo que usa o algoritmo TS como gerador das rotas finais e coordenadas geradas de forma aleatória com base nos dados do OpenStreetMaps. Para a demonstração foram utilizados os dados geográficos da cidade de Patos - PB, com o depósito definido no local da UEPB, que são apresentados na Figura 17.

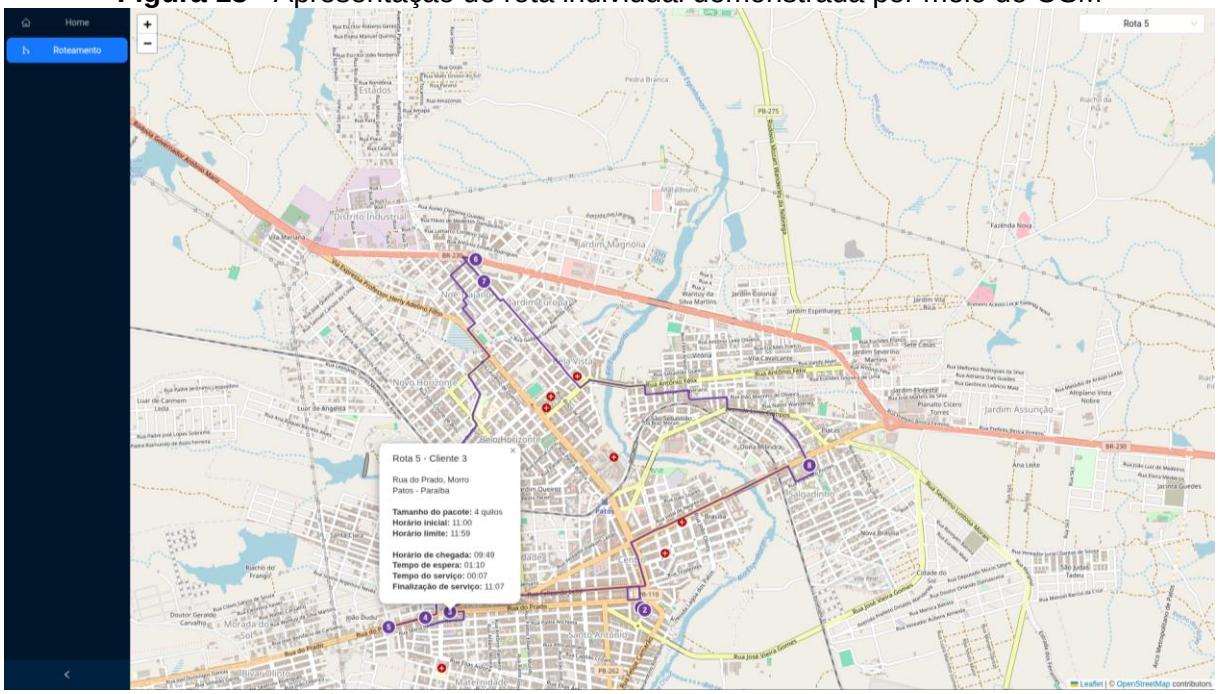
Figura 17 - Apresentação de solução completa demonstrada por meio do OSM



Fonte: Autoria própria.

A Figura 17 demonstra a apresentação das rotas utilizando as tecnologias citadas na seção 3.6, sendo possível visualizar o roteamento completo gerado pela Busca Tabu, além dos dados relacionados a cada um dos clientes das rotas. As rotas entre clientes são previamente geradas utilizando o algoritmo Dijkstra com auxílio da fórmula de Haversine descrita na seção 2.3.2 do presente trabalho, assumindo uma velocidade média de $v = 40km$. Dessa forma, este protótipo demonstra a aplicabilidade das tecnologias selecionadas, em conjunto com os algoritmos desenvolvidos, na construção de uma solução de roteamento. Para uma melhor visualização de uma rota gerada individual, a Figura 18 apresenta uma das rotas geradas pela solução.

Figura 18 - Apresentação de rota individual demonstrada por meio do OSM



Fonte: Autoria própria.

A Figura 18 demonstra a solução de uma rota individual, onde é possível selecionar cada uma das rotas e visualizar os dados calculados pelo algoritmo TS, como previsão de tempo de chegada no cliente, previsão de tempo de espera necessário, finalização do serviço, gerando assim uma melhor forma de visualização dos resultados para um usuário final, definindo as melhores rotas para minimizar as distâncias com base na frota de veículos utilizada, não excedendo os tempos de chegada de cada cliente, capacidade máxima dos veículos e não excedendo o horário máximo que o veículo pode realizar as entregas.

Por fim, com a análise geral, a Busca Tabu se destaca aos demais algoritmos, já que possui um menor tempo de execução necessário, atingindo resultados semelhantes ou melhores que os demais. Isso acontece já que, na sua forma original, o algoritmo genético apresenta limitações de intensificação e diversificação em relação a meta-heurísticas como a busca tabu (Labadie; Prins; Prodhon, 2016). Para contornar essas deficiências, utiliza-se as estratégias propostas por Prins (2004). Contudo, a ausência de exploração de soluções inviáveis limitou a capacidade de exploração do Algoritmo Genético implementado no presente trabalho.

A evolução das implementações dos algoritmos ao longo do tempo demonstra que ambas as estratégias têm grande potencial, com trabalhos como Bräysy e Gendreau (2002) que destacavam como a Busca Tabu mostrava ter o melhor

desempenho na solução do VRPTW. No entanto, em trabalhos recentes, o Algoritmo Genético tem se destacado. Um exemplo é o estudo de Vidal *et al*. (2013), que é destacado por Toth e Vigo (2014) por seus bons resultados, usando o cruzamento por ordem, considerando soluções viáveis e inviáveis de forma separada, aliado com uma fase de reparação de soluções inviáveis.

Além disso, o evento do DIMACS (2022), mencionado na seção 2.8, classificou as implementações submetidas, e os melhores resultados utilizavam estratégias evolucionárias, o que reforça ainda mais o destaque do Algoritmo Genético atualmente. O algoritmo de Otimização por Colônia de Formigas, apesar de apresentar resultados promissores para estratégias aglomeradas, não obteve um desempenho semelhante nas demais classes, não mostrando uma constância satisfatória em relação ao principal critério de avaliação, assim, de forma geral, ficando abaixo dos demais.

5 CONCLUSÃO

O presente trabalho contribuiu para a literatura por meio da descrição dos principais algoritmos meta-heurísticos do estado da arte aplicados ao problema de roteamento de veículos com janela de tempo, além de aplicar algumas das soluções mais pesquisadas segundo estudos de revisões da literatura, identificando que a Otimização por Colônia de Formigas possui um potencial para resolver soluções que tem clientes aglomerados, o Algoritmo Genético e Busca Tabu propostos apresentando resultados numéricos semelhantes, porém com o Busca Tabu gerando resultados de forma mais eficiente, com um menor tempo total de execução.

Os resultados, apesar de não se aproximarem dos resultados do estado da arte, o que se dá muito pela implementação de forma mais simples, não utilizando de estratégias de diversificação sofisticadas, como busca local variável ou busca local iterada, que podem ser aplicadas aos algoritmos melhorando significativamente os resultados. Além disso, a limitação por explorar apenas soluções viáveis, parece não gerar resultados tão promissores quanto soluções que trabalham com o espaço não viável, o que se dá devido a necessidade de os algoritmos escaparem de ótimos locais.

Com isso, foram evidenciados os pontos positivos e negativos dos algoritmos e contribuiu-se para que trabalhos futuros possam selecionar os algoritmos de acordo com suas necessidades. Além disso, a elaboração e ilustração do protótipo por meio do *framework* para aplicação do problema em cenários reais, por meio da ferramenta OpenStreetMaps, contribuindo para que trabalhos futuros que busquem desenvolver soluções para otimizar, possam utilizar da *stack* de tecnologias e algoritmos para solucionar problemas logísticos reais.

5.1 Trabalhos Futuros

Como trabalhos futuros, os testes dos algoritmos podem ser aperfeiçoados ao utilizar orçamento computacional, limitando a quantidade de avaliações de qualidade de uma solução resultante. Essa abordagem pode ser útil para igualar as condições entre os algoritmos, avaliando com maior precisão, mas que no contexto de múltiplas meta-heurísticas, necessitam de uma adaptação para casos específicos, como avaliação de soluções no meio do processo do algoritmo.

Outra extensão dos testes é a utilização de *datasets* maiores, como as instâncias de Gehring e Homberger (1999), que contém até 1000 consumidores, analisando a escalabilidade em condições mais desafiadoras dos algoritmos propostos no trabalho. A execução dessas extensões pode necessitar de altos tempos computacionais, o que é um desafio a ser considerado.

Além disso, os algoritmos propostos podem servir de base para implementação de soluções para outras variantes do VRP, como por exemplos variantes que consideram mudanças dinâmicas, com múltiplos depósitos, frota heterogênea. As soluções também podem ser aperfeiçoadas por meio de técnicas de outros algoritmos, como desconstrução e reconstrução de rotas, ou combinando conceitos entre si, como usar conceitos da lista tabu juntamente com a população do algoritmo genético, visando diversificar mais ainda a exploração.

Por fim, uma extensão mais clara é o aperfeiçoamento do protótipo desenvolvido, melhorando a integração com ferramentas de dados geográficos em tempo real, visando sua adaptação a cenários reais de logística, assim gerando utilidade e resolvendo problemas da sociedade.

REFERÊNCIAS

ABU, N. et al. A comprehensive overview of classical and modern route planning algorithms for self-driving mobile robots. **Journal of Robotics and Control (JRC)**, Yogyakarta, v. 3, n. 5, p. 666–678, 2022. Disponível em: https://doi.org/10.18196/jrc.v3i5.14683. Acesso em: 09 ago. 2024.

ANBUUDAYASANKAR, S. P.; GANESH, K.; MOHAPATRA, S. Models for practical routing problems in logistics: Design and Practices. **Cham: Springer**, 2014. 165 p. Disponível em: https://doi.org/10.1007/978-3-319-05035-5. Acesso em: 07 out. 2024.

ANDREOU, A. et al. UAV Trajectory Optimisation in Smart Cities Using Modified A* Algorithm Combined With Haversine and Vincenty Formulas. *In:* **IEEE Transactions on Vehicular Technology**, Thessaloniki, Grécia: IEEE, v. 72, n. 8, p. 9757–9769, 2023. Disponível em: https://doi.org/10.1109/TVT.2023.3254604. Acesso em: 12 out. 2024.

ARCURI, A.; BRIAND, L. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In: 33rd International Conference on Software Engineering (ICSE), 2011, Waikiki, Honolulu, HI, United States. **Anais** […] New York: ACM, maio 2011. p. 1-10. Disponível em: https://doi.org/10.1145/1985793.1985795. Acesso em: 26 nov. 2024.

BARCZYSZYN, G. L. Sistema colaborativo para planejamento de rotas para cadeirantes. 2019. 71 f.Dissertação (Pós-Graduação em Computação Aplicada) – Universidade Tecnológica Federal Do Paraná, Curitiba, 2019. Acesso em: 11 ago. 2024.

BEIRANVAND, V.; HARE, W.; LUCET, Y. Best practices for comparing optimization algorithms. **Optimization and Engineering**, v. 18, n. 4, p. 815–848, set. 2017. disponível em: https://doi.org/10.1007/s11081-017-9366-1. Acesso em: 03 abr. 2025.

BRÄYSY, O.; GENDREAU, M. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. **Transportation Science**, v. 39, n. 1, p. 104–118, fev. 2005. Disponível em: https://doi.org/10.1287/trsc.1030.0056. Acesso em: 20 fev. 2025.

BRÄYSY, O.; GENDREAU, M. Tabu search heuristics for the vehicle routing problem with time windows. **Top**, v. 10, n. 2, p. 211–237, dez. 2002. Disponível em: https://doi.org/10.1007/BF02579017. Acesso em: 11 mar. 2025.

CORDEAU, J.-F. et al. Vehicle routing. Amsterdam: Elsevier, 2007. **Handbooks in operations research and management science**, v. 14, p. 367–428, 2007. Disponível em: https://doi.org/10.1016/S0927-0507(06)14006-2. Acesso em: 28 fev. 2025.

CVRPLIB Capacitated Vehicle Routing Problem Library [online]. 2025. Disponível em: http://vrp.atd-lab.inf.puc-rio.br. Acesso em: 08 abr. 2025.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. **Management science**, Informs, v. 6, n. 1, p. 80–91, 1959. Disponível em: https://doi.org/10.1287/mnsc.6.1.80. Acesso em: 29 set. 2024.

DORIGO, M.; BIRATTARI, M.; STUTZLE, T. Ant colony optimization. **IEEE computational intelligence magazine**, v. 1, n. 4, p. 28–39, nov. 2006. Disponível em: https://doi.org/10.1109/MCI.2006.329691. Acesso em: 11 nov. 2024.

DORIGO, M.; STÜTZLE, T. Ant Colony Optimization: Overview and Recent Advances. *In*: GENDREAU, M.; POTVIN, J.-Y. (eds.). **Handbook of Metaheuristics**. International Series in Operations Research & Management Science. Cham: Springer International Publishing, 2019. v. 272. p. 311–351. Disponível em: https://doi.org/10.1007/978-3-319-91086-4_10. Acesso em: 12 nov. 2024.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische mathematik**, v. 1, p. 269–271, 1959. Disponível em: https://doi.org/10.1007/BF01386390. Acesso em: 08 set. 2024.

DIMACS. Final papers & videos [online]. [s.l.]: DIMACS, [2022]. Disponível em: http://dimacs.rutgers.edu/programs/challenge/vrp/papers-videos/. Acesso em: 08 mar. 2025.

EL-SHERBENY, N. A. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. **Journal of King Saud University-Science**, v. 22, n. 3, p. 123–131, jul. 2010. Disponível em: https://doi.org/10.1016/j.jksus.2010.03.002. Acesso em: 15 mar. 2025.

FONTE, C. C. et al. **Mapping and the citizen sensor**. London: Ubiquity Press, 2017. Disponível em: https://doi.org/10.5334/bbf. Acesso em: 06 set. 2024.

FOSCI, P.; PSAILA, G. Soft querying features in geojson documents: The geosoft proposal. **International Journal of Computational Intelligence Systems**, Springer, v. 16, n. 1, p. 163, 2023. Disponível em: https://doi.org/10.1007/s44196-023-00325-3. Acesso em: 07 set. 2024.

GE, B.; HU, S.; ZHENG, P. Research on full traversal path planning based on improved reciprocating algorithm. *In*: **2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)**. Chongqing. Proceedings. Piscataway: IEEE, 2020. v. 9, p. 922–926. Disponível em: https://doi.org/10.1109/ITAIC49862.2020.9338939. Acesso em: 27 out. 2024.

GEHRING, H.; HOMBERGER, J. **A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows**. *In*: Proceedings of EUROGEN99. **Anais** […]. Citeseer, 1999. p. 57-64. Disponível em: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e41b80d075429403f569980d278f6d4f91da2594. Acesso em: 29 dez. 2024.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. **Computers & operations research**, v. 13, n. 5, p. 533–549, 1986. Disponível em: https://doi.org/10.1016/0305-0548(86)90048-1. Acesso em: 17 mar. 2025.

GUNAWAN, A. et al. Vehicle routing: Review of benchmark datasets. **Journal of the Operational Research Society**, Taylor & Francis, v. 72, n. 8, p. 1794–1807, 2021. Disponível em: https://doi.org/10.1080/01605682.2021.1884505. Acesso em: 15 fev. 2025.

HANSEN, P. et al. Variable neighborhood search: basics and variants. **EURO Journal on Computational Optimization**, v. 5, n. 3, p. 423–454, 2017. Disponível em: https://doi.org/10.1007/s13675-016-0075-x. Acesso em: 11 mar. 2025.

HOLLAND, J. H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. Cambridge: MIT Press, 1975.

HORBIŃSKI, T.; LOREK, D. **The use of leaflet and GeoJSON files for creating the interactive web map of the preindustrial state of the natural environment**. *Journal of Spatial Science*, v. 67, n. 1, p. 61–77, 2022. Disponível em: https://doi.org/10.1080/14498596.2020.1713237. Acesso em: 07 set. 2024.

JAMIL, M. A. et al. **Software testing techniques: A literature review**. International conference on information and communication technology for the Muslim world (ICT4M), 6., 2016, Jakarta. **Anais** [...], IEEE, 2016. Disponível em: https://doi.org/10.1109/ICT4M.2016.045. Acesso em: 26 nov. 2024.

JIANG, S. et al. FHDSolver: Fast and Effective VRPTW Solver. In: *12th DIMACS Implementation Challenge Workshop*, 2022. Disponível em: http://dimacs.rutgers.edu/events/details?eID=2076. Acesso em: 08 abr. 2025.

JÜNGER, M.; REINELT, G.; RINALDI, G. The traveling salesman problem. *In*: BALL, M. O.; MAGNANTI, T. L.; MONMA, C. L.; NEMHAUSER, G. L. (eds.). **Handbooks in operations research and management science**, Amsterdam: Elsevier Science, v. 7, p. 225–330, 1995. Disponível em: https://doi.org/10.1016/S0927-0507(05)80121-5. Acesso em: 30 mar. 2025.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. Science, v. 220, n. 4598, p. 671–680, 13 maio 1983. Disponível em: https://doi.org/10.1126/science.220.4598.671. Acesso em: 11 mar. 2025.

KOOL, W. et al. Hybrid genetic search for the vehicle routing problem with time windows: a high-performance implementation. In: *12th DIMACS Implementation Challenge Workshop*, 2022. Disponível em: https://wouterkool.github.io/publication/hgs-vrptw/. Acesso em: 08 abr. 2025.

KONSTANTAKOPOULOS, G. D.; GAYIALIS, S. P.; KECHAGIAS, E. P. Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification. **Operational research**, Springer, v. 22, n. 3, p. 2033–2062, jul. 2022. Disponível em: https://doi.org/10.1007/s12351-020-00600-7. Acesso em: 10 ago. 2024.

KUMAR, M. et al. Genetic algorithm: review and application. [S.l.]: SSRN, 1 dez. 2010. Disponível em: https://ssrn.com/abstract=3529843. Acesso em: 11 mar. 2025.

LABADIE, N.; PRINS, C.; PRODHON, C. **Metaheuristics for vehicle routing problems**. [S.l.]: John Wiley & Sons, 2016.

LAPORTE, G. The vehicle routing problem: An overview of exact and approximate algorithms. **European journal of operational research**, v. 59, n. 3, p. 345–358, 1992. Disponível em: https://doi.org/10.1016/0377-2217(92)90192-C. Acesso em: 22 nov. 2024.

LI, Y. et al. Dual-level local search memetic algorithm for the vehicle routing problem with time windows. In: *12th DIMACS Implementation Challenge Workshop*, 2022. Disponível em: http://dimacs.rutgers.edu/events/details?eID=2091. Acesso em: 08 abr. 2025.

LIU, X. et al. A systematic literature review of vehicle routing problems with time windows. **Sustainability**, v. 15, n. 15, p. 12004, 2023. Disponível em: https://doi.org/10.3390/su151512004. Acesso em: 15 fev. 2025.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search. In: GLOVER, Fred; KOCHENBERGER, Gary A. (Orgs.). **Handbook of Metaheuristics**. International Series in Operations Research & Management Science, v. 57. Boston: Springer, 2003. p. 320–353. Disponível em: https://doi.org/10.1007/0-306-48056-5_11. Acesso em: 05 abr. 2025.

LU, E. H.-C.; YANG, Y.-W. A hybrid route planning approach for logistics with pickup and delivery. **Expert Systems with Applications**, Elsevier, v. 118, p. 482–492, 2019. Disponível em: https://doi.org/10.1016/j.eswa.2018.10.031. Acesso em: 19 ago. 2024.

MARA, S. T. W. et al. A survey of adaptive large neighborhood search algorithms and applications. **Computers & Operations Research**, v. 146, p. 105903, 2022. Disponível em: https://doi.org/10.1016/j.cor.2022.105903. Acesso em: 11 mar. 2025.

MARIA, E. et al. Measure distance locating nearest public facilities using Haversine and Euclidean Methods. In: IOP PUBLISHING. **Journal of Physics: Conference Series**. [S.l.], 2020. v. 1450, n. 1, p. 012080. Disponível em: https://doi.org/10.1088/1742-6596/1450/1/012080. Acesso em: 12 out. 2024.

MINGHINI, M.; FRASSINELLI, F. Openstreetmap history for intrinsic quality assessment: Is osm up-to-date? **Open Geospatial Data, Software and Standards**, Springer, v. 4, n. 1, p. 1–17, 2019. Disponível em: https://doi.org/10.1186/s40965-019-0067-x. Acesso em: 04 ago. 2024.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. **Computers & operations research**, v. 24, n. 11, p. 1097–1100, 1997. Disponível em: https://doi.org/10.1016/S0305-0548(97)00031-2. Acesso em: 11 mar. 2025.

NHA, V. T. N.; DJAHEL, S.; MURPHY, J. A comparative study of vehicles' routing algorithms for route planning in smart cities. In: INTERNATIONAL WORKSHOP ON VEHICULAR TRAFFIC MANAGEMENT FOR SMART CITIES (VTM), 1., 2012, Dublin. **Anais** [...]. [S.l.]: IEEE, 2012. p. 1–6. Disponível em: https://doi.org/10.1109/VTM.2012.6398701. Acesso em: 04 ago. 2024.

OPENSTREETMAP. **OpenStreetMap Wiki [online]**. [S.l.]: OpenStreetMap Foundation, 2024. Disponível em: https://wiki.openstreetmap.org/. Acesso em: 06 set. 2024.

OPENSTREETMAP. **OpenStreetMap Wiki: Software/Desktop [online]**. [S.l.]: OpenStreetMap Foundation, 2025. Disponível em: https://wiki.openstreetmap.org/wiki/Software/Desktop. Acesso em: 12 abr. 2025.

PISINGER, D.; ROPKE, S. Large Neighborhood Search. *In*: GENDREAU, M., POTVIN, J-Y. (eds). **Handbook of Metaheuristics**. International Series in Operations Research & Management Science. Cham: Springer International Publishing, 2019. v. 272. p. 99–127. Disponível em: https://doi.org/10.1007/978-3-319-91086-4_4. Acesso em: 11 mar. 2025.

PRINS, C. A simple and effective evolutionary algorithm for the vehicle routing problem. **Computers & Operations Research**, v. 31, n. 12, p. 1985–2002, out. 2004. Disponível em: https://doi.org/10.1016/S0305-0548(03)00158-8. Acesso em: 03 mar. 2025.

SHAW, P. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. *In*: MAHER, M.; PUGET, J.-F. (Eds.). **Principles and Practice of Constraint Programming — CP98**. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. v. 1520. p. 417–431. Disponível em: https://doi.org/10.1007/3-540-49481-2_30. Acesso em: 13 abr. 2025.

SOLOMON, M. M. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. **Operations Research**, v. 35, n. 2, p. 254–265, abr. 1987. Disponível em: https://doi.org/10.1287/opre.35.2.254. Acesso em: 22 dez. 2024.

SOUZA, L. L. d. L. Uso da inteligência artificial no planejamento de trajetórias. 2023. 52 f. Monografia (Graduação em Engenharia de Controle e Automação) - Escola de Minas, Universidade Federal de Ouro Preto, Ouro Preto, 2023. Disponível em: http://www.monografias.ufop.br/handle/35400000/5914. Acesso em: 11 ago. 2024.

TOTH, P.; VIGO, D. (Ed.). **The Vehicle Routing Problem**. Philadelphia: Society for Industrial and Applied Mathematics, 2002. 385 p.

TOTH, P.; VIGO, D. (Ed.). Vehicle routing: problems, methods, and applications. 2. ed. Philadelphia: Society for Industrial and Applied Mathematics, 2014. 480 p. Disponível em: https://doi.org/10.1137/1.9781611973594. Acesso em: 27 fev. 2025.

VEHICLE ROUTING PROBLEM REPOSITORY. **VRP-REP: the vehicle routing problem repository [online].** [S.l.]: s.n., 2014. Repositório de dados. Disponível em: http://www.vrp-rep.org/. Acesso em: 15 nov. 2024.

VIDAL, T. et al. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. **Computers & Operations Research**, v. 40, n. 1, p. 475–489, jan. 2013. Disponível em: https://doi.org/10.1016/j.cor.2012.07.018. Acesso em: 08 mar. 2025.

WAZLAWICK, Raul Sidnei. **Metodologia de pesquisa para Ciência da Computação.** 1. ed. Rio de Janeiro: Elsevier, 2009. 184 p.

ZHU, Z.; TAN, J. A multi-source heterogeneous vector space data integration scheme based on GeoJSON. *In*: **International Conference On Geoinformatics**, 26., 2018, Kunming, China. Anais […] Kunming: IEEE, 2018. p. 1–4. Disponível em: https://doi.org/10.1109/GEOINFORMATICS.2018.8557141. Acesso em: 07 set. 2025.

APÊNDICE A - TABELA DE RESULTADOS COMPLETA

Tabela 3 - Resultados completos

Algoritmo	*Dataset*	Distância	Tempo	Veículos	Tempo execução (s)
ACO	C101	828,94	9828,94	10,00	04,04
ACO	C102	990,42	10413,96	10,20	13,15
ACO	C103	1151,72	10691,48	11,40	20,37
ACO	C104	1210,07	10790,45	11,00	26,22
ACO	C105	828,94	9828,94	10,00	4,49
ACO	C106	853,85	9853,85	10,00	4,63
ACO	C107	832,76	9832,76	10,00	5,94
ACO	C108	892,50	9903,67	10,40	5,49
ACO	C109	982,69	10090,75	10,60	7,25
ACO	C201	591,56	9591,56	3,00	04,05
ACO	C202	719,14	10383,22	3,40	12,99
ACO	C203	926,49	12903,13	4,60	33,28
ACO	C204	1058,55	13372,64	4,60	41,87
ACO	C205	627,28	9777,76	3,40	4,80
ACO	C206	650,90	9727,81	3,20	5,93
ACO	C207	650,60	9650,65	3,20	6,40
ACO	C208	645,03	9645,03	3,20	6,63
ACO	R101	1895,89	3316,02	21,00	9,82
ACO	R102	1720,44	3171,34	19,20	18,83
ACO	R103	1454,47	2861,73	15,80	38,00
ACO	R104	1252,29	2471,78	13,20	41,24
ACO	R105	1611,11	2750,19	16,00	11,94
ACO	R106	1483,05	2625,31	14,40	18,31
ACO	R107	1326,45	2494,81	12,80	27,10
ACO	R108	1165,96	2280,85	11,60	43,30
ACO	R109	1456,10	2505,87	13,40	15,64
ACO	R110	1387,10	2477,07	13,00	20,79
ACO	R111	1340,49	2437,34	13,00	24,66
ACO	R112	1215,52	2243,08	11,20	31,16
ACO	R201	1590,83	3568,24	5,00	14,26
ACO	R202	1403,20	3510,89	4,40	35,20
ACO	R203	1366,73	3291,01	4,20	30,93
ACO	R204	1060,50	3111,74	4,20	58,68
ACO	R205	1324,10	2855,12	4,00	25,90
ACO	R206	1267,96	2816,29	3,80	32,62
ACO	R207	1177,28	3140,08	3,80	38,24

ACO	R208	1023,00	2625,57	3,00	50,79
ACO	R209	1314,47	2824,49	4,00	27,69
ACO	R210	1340,30	2851,28	4,00	28,35
ACO	R211	1153,34	2314,02	3,00	28,82
ACO	RC101	1897,32	3106,29	17,00	7,19
ACO	RC102	1789,63	2941,69	15,40	11,18
ACO	RC103	1599,43	2796,83	13,60	24,45
ACO	RC104	1396,29	2551,44	12,00	31,00
ACO	RC105	1820,62	3016,16	17,00	13,29
ACO	RC106	1654,51	2709,20	13,80	8,29
ACO	RC107	1479,97	2573,66	13,00	19,78
ACO	RC108	1418,72	2506,38	12,20	24,79
ACO	RC201	1772,42	3838,19	5,20	10,75
ACO	RC202	1590,14	3650,70	5,00	25,24
ACO	RC203	1331,99	3460,71	4,60	34,89
ACO	RC204	1162,39	2695,60	3,60	32,89
ACO	RC205	1682,47	3894,43	5,60	25,06
ACO	RC206	1486,42	3036,61	4,60	16,57
ACO	RC207	1499,28	3018,87	4,40	20,96
ACO	RC208	1217,31	2545,59	3,40	29,93
TS	C101	1002,74	10253,60	10,40	2,22
TS	C102	1130,76	10998,17	11,00	7,78
TS	C103	1276,88	11646,24	11,80	18,26
TS	C104	1072,29	11487,38	10,80	31,51
TS	C105	934,60	9937,28	10,00	2,32
TS	C106	1034,30	10084,63	10,00	2,67
TS	C107	1108,17	10602,69	11,00	3,76
TS	C108	977,89	10559,55	11,00	06,03
TS	C109	1093,38	10115,89	10,00	5,97
TS	C201	703,27	12435,27	4,00	1,48
TS	C202	733,89	11224,42	4,00	8,82
TS	C203	739,77	12680,71	4,00	11,81
TS	C204	746,45	11205,50	4,00	26,05
TS	C205	743,21	12472,79	4,00	2,00
TS	C206	648,55	11232,99	4,00	3,74
TS	C207	681,57	12213,75	4,00	4,41
TS	C208	677,12	11052,36	4,00	4,99
TS	R101	1761,50	3363,15	20,00	13,64
TS	R102	1552,03	3161,48	18,20	26,47
TS	R103	1353,93	2885,38	15,60	28,82

TS	R104	1174,58	2483,13	12,80	37,11
TS	R105	1496,47	2834,00	17,00	14,15
TS	R106	1447,32	2752,79	15,00	24,88
TS	R107	1261,75	2522,54	13,40	28,67
TS	R108	1141,35	2303,59	12,00	36,18
TS	R109	1388,95	2456,98	14,00	11,74
TS	R110	1282,66	2406,70	13,20	25,38
TS	R111	1260,94	2402,20	13,60	24,12
TS	R112	1143,74	2183,36	12,00	37,68
TS	R201	1431,07	4036,56	5,00	7,72
TS	R202	1242,77	3900,57	5,00	25,62
TS	R203	1111,77	4070,56	5,00	35,40
TS	R204	910,93	3357,24	4,60	45,18
TS	R205	1123,31	3045,13	4,00	19,23
TS	R206	1078,08	3098,09	4,00	27,84
TS	R207	961,73	2993,93	3,60	33,49
TS	R208	892,38	2385,27	3,00	35,43
TS	R209	1098,61	2759,36	4,00	19,62
TS	R210	1054,39	3489,42	4,00	25,65
TS	R211	933,51	2308,17	3,00	28,53
TS	RC101	1858,07	3203,67	18,00	9,98
TS	RC102	1691,74	3120,55	16,00	11,36
TS	RC103	1488,64	2807,18	14,20	19,40
TS	RC104	1344,43	2550,21	11,80	20,84
TS	RC105	1737,16	3064,83	16,00	12,80
TS	RC106	1583,53	2739,97	15,40	9,24
TS	RC107	1491,89	2654,05	14,00	17,18
TS	RC108	1405,64	2458,06	12,20	14,75
TS	RC201	1796,12	4224,61	5,00	5,75
TS	RC202	1424,49	3825,84	5,00	12,71
TS	RC203	1141,07	3442,06	4,20	22,02
TS	RC204	975,86	2754,59	3,80	27,48
TS	RC205	1365,29	4274,03	6,00	12,12
TS	RC206	1375,95	3067,52	4,00	9,41
TS	RC207	1270,35	2950,87	4,00	19,37
TS	RC208	1055,86	2282,07	3,00	16,27
GA	C101	829,70	9906,87	10,00	20,98
GA	C102	1268,49	11232,62	11,20	61,08
GA	C103	1255,90	11760,35	11,60	148,92
GA	C104	1157,55	11349,53	10,80	242,23

GA	C105	874,28	9889,09	10,00	21,34
GA	C106	1053,36	10711,38	11,00	35,25
GA	C107	1119,29	10219,42	10,00	21,95
GA	C108	1005,68	10140,44	10,80	49,07
GA	C109	1135,44	10447,67	11,60	84,05
GA	C201	689,87	12528,69	4,00	29,16
GA	C202	691,10	12001,07	4,00	67,94
GA	C203	730,42	11813,75	4,00	155,82
GA	C204	746,68	12795,32	4,20	215,58
GA	C205	765,26	12015,28	3,80	35,32
GA	C206	656,68	10194,81	4,00	46,64
GA	C207	707,39	10952,84	4,00	54,54
GA	C208	672,19	11026,54	4,00	70,40
GA	R101	1819,19	3464,48	21,00	135,79
GA	R102	1721,59	3215,26	19,00	188,77
GA	R103	1408,51	2997,73	15,60	199,85
GA	R104	1220,44	2418,30	11,40	197,63
GA	R105	1623,48	2853,67	16,00	99,14
GA	R106	1465,54	2716,85	15,60	188,56
GA	R107	1271,79	2521,45	13,00	194,45
GA	R108	1167,58	2291,43	11,20	217,84
GA	R109	1413,60	2541,24	14,20	125,16
GA	R110	1306,85	2397,99	13,00	143,22
GA	R111	1314,57	2493,84	12,80	183,95
GA	R112	1155,26	2196,70	11,40	213,08
GA	R201	1411,95	4013,90	5,00	99,78
GA	R202	1262,80	4050,72	5,20	179,68
GA	R203	1088,13	3745,63	4,60	251,60
GA	R204	950,43	2812,86	3,40	291,74
GA	R205	1186,90	3022,29	4,00	166,39
GA	R206	1066,81	3113,31	4,00	202,42
GA	R207	960,84	2660,88	3,00	254,80
GA	R208	846,71	2479,93	3,00	324,97
GA	R209	1155,92	2618,11	4,00	200,76
GA	R210	1114,94	3161,19	4,00	205,51
GA	R211	920,12	2373,86	3,40	222,11
GA	RC101	1858,63	3151,76	18,00	87,57
GA	RC102	1677,19	2934,49	15,00	83,78
GA	RC103	1429,80	2721,90	13,00	121,26
GA	RC104	1365,15	2532,52	12,00	153,74

GA	RC105	1749,37	3186,90	17,20	100,46
GA	RC106	1674,82	2769,52	14,80	76,76
GA	RC107	1542,15	2662,25	14,00	99,95
GA	RC108	1370,97	2448,18	12,20	110,22
GA	RC201	1644,78	4103,44	5,00	66,31
GA	RC202	1343,82	4137,67	5,40	120,61
GA	RC203	1219,79	3455,89	4,00	164,89
GA	RC204	964,60	3089,18	4,20	218,63
GA	RC205	1470,43	4686,93	6,40	124,63
GA	RC206	1261,78	3052,47	4,00	109,23
GA	RC207	1199,74	3172,64	5,00	124,03
GA	RC208	991,24	2589,14	4,00	241,02

Fonte: Autoria própria.