



UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO.

Dissertação de Mestrado

**Aplicação para o Problema de Roteamento de Veículos na
entrega *Last Mile* baseada em algoritmos tradicionais e
*Benchmark Loggi BUD***

Patricia Almeida da Silva

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande - FURG, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Computação.

Orientadora: Prof^ª. Dr^ª. Regina Barwaldt
Coorientador: Prof. Dr. Giancarlo Lucca

Rio Grande, 2023

Ficha Catalográfica

S586a Silva, Patrícia Almeida da.
Aplicação para o Problema de Roteamento de Veículos na entrega *Last Mile* baseada em algoritmos tradicionais e *Benchmark Loggi BUD* / Patrícia Almeida da Silva. – 2023.
66 f.

Dissertação (mestrado) – Universidade Federal do Rio Grande – FURG, Programa de Pós-Graduação em Computação, Rio Grande/RS, 2023.

Orientadora: Dra. Regina Barwaldt.

Coorientador: Dr. Giancarlo Lucca.

1. Problema de roteamento de veículos 2. Última milha
3. Algoritmo de *Dijkstra* 4. *Benchmark Loggi BUD* I. Barwaldt, Regina
II. Lucca, Giancarlo III. Título.

CDU 004

Catálogo na Fonte: Bibliotecário José Paulo dos Santos CRB 10/2344

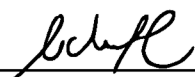
DISSERTAÇÃO DE MESTRADO

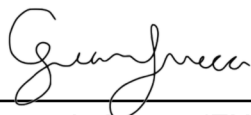
**Aplicação para o Problema de Roteamento de Veículos na entrega Last
Mile baseada em algoritmos tradicionais e Benchmark Loggi BUD**

Patricia Almeida da Silva

Banca examinadora:

Documento assinado digitalmente
ROBERTO DOS SANTOS RABELLO
Data: 31/05/2023 17:12:37-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Roberto Dos Santos Rabello (UPF)

Profa. Dra. Helida Salles Santos (FURG)

Prof. Dr. Giancarlo Lucca (FURG)
CoorientadorDocumento assinado digitalmente
REGINA BARWALDT
Data: 04/06/2023 18:12:00-0300
Verifique em <https://validar.iti.gov.br>

Profa. Dra. Regina Barwaldt (FURG)
Orientadora

AGRADECIMENTOS

Este trabalho de pesquisa é dedicado em especial ao meu marido Alessandro Righi que sempre me ajudou com sua experiência e conhecimentos, sempre me apoiou em todos os aspectos. As incontáveis horas de troca de ideias valeram a pena. Muito obrigado!

Agradeço aos meus pais que, desde cedo, me ensinaram o valor da educação para se entender o mundo e que me mostraram, com muita paciência e amor, que não há limites para a busca de um sonho, para se querer sempre mais da vida e ser feliz.

Agradeço também a empresa Loggi pelo apoio e incentivo por meio do Programa de Bolsas de Pesquisa (PBP - Loggi).

Por fim, mas não menos importante, agradeço aos meus orientadores Regina Barwaldt e Giancarlo Lucca pela orientação, parceria, dedicação, paciência e profissionalismo.

“A persistência é o caminho do êxito”.
—CHARLES CHAPLIN

RESUMO

SILVA, Patricia Almeida da. **Aplicação para o Problema de Roteamento de Veículos na entrega *Last Mile* baseada em algoritmos tradicionais e *Benchmark Loggi BUD***. 2023. 66 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande - FURG, Rio Grande.

Problemas de Roteamento de Veículos (PRV) começaram a ser estudados há mais de cinquenta anos a fim de encontrar soluções para minimizar distâncias percorridas nas demandas de entrega. Com a globalização, o número de entregas e a complexidade das rotas aumentaram, e consequentemente os custos operacionais são elevados. Esse cenário se reflete também nas rotas de transporte escolar que atualmente exigem grandes investimentos dos órgãos públicos. Considerando esse contexto, o presente trabalho teve como objetivo desenvolver uma aplicação para tratar o PRV. Para isso, foi utilizado um conhecido algoritmo de menor caminho, o *Dijkstra*. Tal abordagem foi submetida em seus testes iniciais no *Benchmark Loggi BUD*, conjunto de dados reais para problemas de entrega urbana disponibilizados pela *Loggi*. Nos testes seguintes, foi aplicada em conjuntos de dados XML do *OpenStreetMap* e em dados de rotas de transporte escolar obtidos em sites de acesso público. A aplicação desenvolvida permite que o usuário faça *upload* de um arquivo XML com os pontos de origem e destino definidos, visualize o mapa da região, a rota de caminho mínimo e a distância total percorrida. Como resultado constatou-se que é efetiva nos cálculos de rotas mínimas, tendo suas rotas comparadas com a ferramenta *Google Maps*, no entanto, seu desempenho limitou-se à capacidade de processamento do *hardware*.

Palavras-chave: Problema de Roteamento de veículos, Última milha, Algoritmo de *Dijkstra*, *Benchmark Loggi BUD*.

ABSTRACT

SILVA, Patricia Almeida da. *Application to the Vehicle Routing Problem in Last Mile delivery based on traditional algorithms and Benchmark Loggi BUD*. 2023. 66 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande - FURG, Rio Grande.

Vehicle Routing Problems (VRP) began to be studied over fifty years ago in order to find solutions to minimize distances traveled on delivery demands. With globalization, the number of deliveries and the complexity of routes have increased, and consequently operating costs are high. This scenario is also reflected in the school transport routes that currently demand large investments by public bodies. Considering this context, the present work had with the aim of developing an application to treat PRV. For this, we used a well-known shortest path algorithm called Dijkstra. Such an approach was observed in its initial tests on Benchmark Loggi BUD which is a real data set for problems urban delivery service provided by Loggi. In the following tests, it was applied to OpenStreetMap XML datasets and school bus route data obtained from publicly accessible websites. The developed application allows the user to upload an XML file with defined origin and destination points, view the map of the region, the shortest path route and the total distance traveled. As a result it was verified that it is effective in the calculations of minimum routes, having its unexpected routes with the Google Maps tool, however, its performance was limited to the hardware processing ability.

Keywords: *Vehicle Routing Problem, Last Mile, Dijkstra's Algorithm, Benchmark Loggi BUD.*

LISTA DE FIGURAS

Figura 1	Esquema gráfico do VRP clássico (Gupta, A. and Saini, S., 2017) [29]	18
Figura 2	Esquema gráfico da entrega <i>Last Mile</i> (Chu, H., Zhang, W., et al., 2021) [16]	22
Figura 3	Exemplo de grafo	25
Figura 4	Exemplo de grafo direcionado e grafo não direcionado	26
Figura 5	Grafo não direcionado e lista de adjacências correspondente	26
Figura 6	Grafo não direcionado e matriz de adjacências correspondente	27
Figura 7	Exemplo de caminho em um grafo	27
Figura 8	Pseudocódigo do algoritmo de <i>Dijkstra</i> (Zanatto, 2021) [69]	29
Figura 9	Exemplo de uso do algoritmo de <i>Dijkstra</i> e matriz de distâncias	29
Figura 10	Exemplo de utilização do algoritmo de <i>Floyd-Warshall</i> e matriz de distâncias mínimas	30
Figura 11	Pseudocódigo do algoritmo de <i>Floyd-Warshall</i> (2021)	30
Figura 12	Exemplo execução do algoritmo de <i>Bellman-Ford</i> (Stevenard, 2022)[65]	31
Figura 13	Quadro comparativo dos algoritmos de <i>Dijkstra</i> , <i>Floyd-Warshall</i> e <i>Bellman-Ford</i>	32
Figura 14	Interface <i>OpenStreetMap</i>	33
Figura 15	Fases da RSL, conforme modelo proposto por <i>Denyer</i> e <i>Tranfield</i>	35
Figura 16	Resultado da <i>string</i> de busca nos repositórios	39
Figura 17	Resultado após critérios de inclusão e exclusão	39
Figura 18	Tipos de produções científicas	39
Figura 19	Nuvem de palavras relacionadas com os países dos estudos selecionados	40
Figura 20	Número de instituições de ensino por país	40
Figura 21	Lista de produções por fontes internacionais de publicações	41
Figura 22	Lista de produções de instituições de ensino brasileiras	42
Figura 23	Número de publicações por ano	42
Figura 24	Nuvem de palavras relacionadas com as palavras-chave nos estudos selecionados	43
Figura 25	Categorização de soluções VRP	43
Figura 26	Categorização de soluções VRP <i>Last Mile</i>	44
Figura 27	Arquitetura do modelo	46
Figura 28	Estrutura dos Dados <i>JSON</i>	47
Figura 29	Docker <i>OSRM</i>	47
Figura 30	Mapa pontos de entrega	48
Figura 31	Resultado Requisição HTTP para distâncias	48

Figura 32	Dados obtidos com o processamento dos dados da Região 0 do RJ . .	51
Figura 33	Mapa dos pontos de entrega classificados no <i>dataset</i> como Região 0 do RJ	51
Figura 34	Mapa pontos de entrega da amostra selecionada	51
Figura 35	Distâncias obtidas com o Algoritmo de <i>Dijkstra</i>	52
Figura 36	Tela inicial do <i>OpenStreetMap</i>	53
Figura 37	Tela inicial da aplicação	53
Figura 38	Rotas mínimas entre os nós	54
Figura 39	Página web contendo menu de opções	55
Figura 40	Mapeamento da cidade de Mata	55
Figura 41	Grafo gerado da cidade de Mata	55
Figura 42	Rota gerada pelo algoritmo de <i>Dijkstra e Bellman-Ford</i>	56
Figura 43	Mapeamento da cidade de Crateús	57
Figura 44	Grafo gerado da cidade de Crateús	57
Figura 45	Rota gerada com <i>Dijkstra e Bellman-Ford</i>	58
Figura 46	Rota gerada com o <i>Maps</i>	58
Figura 47	Rota gerada com <i>Dijkstra e Bellman-Ford</i>	59
Figura 48	Rota gerada com o <i>Maps</i>	59

LISTA DE TABELAS

Tabela 1	Repositórios utilizados na pesquisa	36
Tabela 2	Resultado inicial da pesquisa nos repositórios de dados.	37
Tabela 3	Resultado da pesquisa nos repositórios pesquisados após os critérios.	38

LISTA DE ABREVIATURAS E SIGLAS

ACM	<i>Association for Computing Machinery</i>
ACO	<i>Ant colony optimization</i>
ALNS	<i>Adaptive Large Neighborhood Search</i>
BDTD	Biblioteca Digital Brasileira de Teses e Dissertações
BUD	<i>Benchmark Urban Deliveries</i>
D2D	<i>Door to Door</i>
HTML	<i>HyperText Markup Language</i>
HTTP	<i>HyperText Transfer Protocol</i>
IA	Inteligência Artificial
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
ILS	<i>Iterated Local Search</i>
JSON	<i>JavaScript Object Notation</i>
OSM	<i>Openstreetmap</i>
OSRM	<i>Open Source Routing Machine</i>
PRV	Problema de Roteamento de Veículos
RSL	Revisão Sistemática da Literatura
VNS	<i>Variable neighborhood search</i>
VRP	<i>Vehicle Routing Problem</i>
UFSCar	Universidade Federal de São Carlos
XML	<i>eXtensible Markup Language</i>

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Motivação	15
1.2	Objetivos	16
1.2.1	Objetivo geral	16
1.2.2	Objetivos específicos	16
1.3	Contribuições da pesquisa	16
1.4	Organização do texto	17
2	REFERENCIAL TEÓRICO	18
2.1	<i>Vehicle Routing Problem</i>	18
2.1.1	Variantes VRP	19
2.2	Soluções VRP <i>Last Mile</i>	23
2.3	Técnicas e Métodos aplicados ao VRP <i>Last Mile</i>	23
2.3.1	Métodos exatos	23
2.3.2	Heurísticas	24
2.3.3	Metaheurísticas	24
2.3.4	Métodos híbridos	25
2.3.5	Inteligência Artificial	25
2.4	Grafos	25
2.4.1	Representação de grafos	26
2.4.2	Caminhos em grafos	27
2.5	Algoritmos de Caminhos mínimos	28
2.5.1	Algoritmo de <i>Dijkstra</i>	28
2.5.2	Algoritmo de <i>Floyd-Warshall</i>	29
2.5.3	Algoritmo de <i>Bellman-Ford</i>	31
2.5.4	Comparação entre os Algoritmos de <i>Dijkstra</i> , <i>Floyd-Warshall</i> e <i>Bellman-Ford</i>	31
2.6	Matriz de distâncias	32
2.7	<i>Open Source Routing Machine (OSRM)</i>	32
2.8	<i>OpenStreetMap (OSM)</i>	33
2.9	OSMnx	33
3	TRABALHOS RELACIONADOS	35
3.1	Revisão Sistemática da Literatura	35
3.1.1	Formulação da questão de pesquisa	36
3.1.2	Localização dos estudos	36

3.1.3	Avaliação e seleção	37
3.1.4	Análise e síntese	38
3.1.5	Relatos dos resultados	38
4	PROCEDIMENTOS METODOLÓGICOS	45
4.1	Classificação da pesquisa	45
4.2	Proposta da aplicação	46
4.3	Modelo de arquitetura proposta	46
4.4	<i>Dataset Benchmark Loggi BUD</i>	47
4.5	Validação do modelo	48
5	ANÁLISE E DISCUSSÃO DOS DADOS	50
5.1	Testes com dados do <i>Benchmark Loggi BUD</i>	50
5.2	Testes com dados dinâmicos	52
5.3	Teste com dados de transporte escolar	56
6	CONSIDERAÇÕES FINAIS	60
	REFERÊNCIAS	61

1 INTRODUÇÃO

O Problema de Roteamento de Veículos (PRV), conhecido na literatura como *Vehicle Routing Problem* (VRP), foi originalmente proposto no final da década de 50, por *Dantzig e Ramser* [14]. Tal problema consiste em estabelecer os trajetos de custos mínimos a serem percorridos por veículos com capacidade determinada, que partem de um mesmo ponto de origem, e têm como objetivo atender a demanda de clientes em localizações geograficamente dispersas [14].

No Brasil, no ano de 2020, o comércio eletrônico (*e-commerce*) teve um crescimento de 73,88%, conforme índices do Comitê de Métricas da Câmara Brasileira de Comércio Eletrônico Camara-e.net¹ em parceria com o Movimento Compre e Confie MCC-ENET² [44]. Grande parte desse índice de crescimento se deu devido às restrições impostas no contexto da Pandemia do Covid-19, que gerou uma mudança no comportamento dos consumidores.

O crescimento do *e-commerce* e a subsequente pressão sobre as operações logísticas, aliado à exigência do nível de serviço por parte dos clientes, geram uma demanda por soluções inovadoras no processo de entrega de mercadorias [54].

O Problema de Roteamento de Veículos possui algumas variantes, por isso, embora existam pesquisas a respeito do tema, ainda é muito difícil lidar com algumas aplicações práticas.

Muitas das pesquisas existentes concentraram-se em propriedades analíticas de diferentes variantes. Vários modelos matemáticos foram desenvolvidos, no entanto há incertezas relativas aos VRPs, sendo que a maioria são limitados a teóricos ou estudos empíricos de pequena escala. A implementação desses modelos em dados reais ainda enfrentam desafios consideráveis [2].

Um dos agravantes dos VRPs atuais é a entrega na última milha, também conhecida por *last mile*, ou seja, o problema de roteamento de veículos no último trecho de uma en-

¹Camara-e.net - principal entidade multissetorial da América Latina e entidade brasileira de maior representatividade da Economia Digital. Acesso: <https://www.camara-e.net/>

²MCC-ENET - indicador de consumo do mercado online, conjunto de indicadores de vendas do mercado online brasileiro, que inclui todas as vendas de bens de consumo e entretenimento, excluindo dados de Mercado Livre, viagens/turismo, anúncios (veículos, imóveis, etc.), e aplicativos de transportes e alimentação.

trega de pacote empresa-consumidor até o destinatário final que deve receber a mercadoria em casa ou em um ponto de coleta [24].

A logística da última milha é considerada um dos segmentos mais caros de todo o processo de distribuição de mercadorias, e o menos eficiente. Esse custo deve-se, principalmente, ao elevado índice de entregas com falhas, ocasionadas pela ausência do cliente no endereço indicado [34].

Na Ciência da Computação uma das estruturas de dados existentes mais conhecidas é o grafo [17]. Um grafo é definido por $G = (V, E)$, onde V são os chamados vértices e E as arestas. Analogamente, é possível entender um vértice como um endereço (seja uma residência ou um ponto de coleta) e as arestas os caminhos.

Na abordagem de grafos, temos também os chamados caminhos mínimos, que consistem em encontrar um caminho de um vértice origem até um vértice destino com menor custo em um grafo. Custos, nesse caso, podem se referir às distâncias percorridas de um ponto ao outro.

Dado um grafo, muitas soluções já foram propostas na literatura para encontrar o menor caminho entre vértices, sendo exemplos, *Dijkstra* [43], *Floyd-Warshall* [58], *Bellman-Ford* [49], entre outros.

Considerando as circunstâncias mencionadas, esta pesquisa caracteriza-se pela implementação de aplicação para o VRP, baseada em algoritmos para cálculos de caminhos mínimos, visando otimização.

1.1 Motivação

Os custos no setor logístico são elevados e o transporte, em geral, representa o maior percentual desses gastos, por isso, empresas começam a repensar os processos logísticos, surgindo a necessidade de aplicações capazes de reduzir custos e suprir de maneira satisfatória as demandas.

No transporte escolar, o cenário não é diferente, pois os custos operacionais também são elevados. Segundo dados divulgados pelo Ministério da Educação ³, no período de janeiro de 2021 a abril de 2022, foram repassados para estados e municípios o total de R\$ 909.729.855,27 reais, por meio do Programa Nacional de Apoio ao Transporte do Escolar (PNATE). Vale ressaltar que a verba repassada pelo governo federal é apenas suplementar, ou seja, estados e municípios devem fornecer mais recursos para garantir o transporte escolar [22].

Rotas bem planejadas e calculadas com a maior precisão possível, são importantes aliadas no processo de redução dos custos, diminuição na emissão de gases poluentes e melhoria no fluxo de veículos.

³Dados divulgados pelo MEC no Painel de Investimentos em Educação Básica, link de acesso: <https://www.gov.br/fnde/pt-br/assuntos/noticias/mec-lancou-o-painel-de-investimentos-em-educacao-basica>.

1.2 Objetivos

Para definir melhor o escopo dessa dissertação, os objetivos são divididos em duas partes, geral e específicos. A seguir, são detalhados cada um deles.

1.2.1 Objetivo geral

Este trabalho teve como principal objetivo, desenvolver uma aplicação para o problema de roteamento de veículos na última milha, apoiada no algoritmo de *Dijkstra* e em dados reais obtidos no *Benchmark Loggi BUD (Benchmark Urban Deliveries)* e no *OpenStreetMap*, que possibilite fornecer rotas otimizadas em relação às distâncias percorridas, minimizando percursos e consequentemente custos operacionais.

1.2.2 Objetivos específicos

Esse trabalho constituiu-se a partir dos seguintes objetivos específicos:

- Revisão Sistemática da Literatura sobre o Problema de Roteamento de Veículos na entrega *Last Mile*;
- Compreensão da estrutura do *Dataset Benchmark Loggi BUD*;
- Estudo dos algoritmos para cálculo de caminhos mínimos e ferramentas para geração de mapas e rotas;
- Proposta de desenvolvimento de uma aplicação para cálculo de caminhos mínimos baseada nos algoritmos de *Dijkstra* e *Bellman-Ford*;
- Testes e validação da aplicação com dados reais e disponibilização de rotas gráficas.

1.3 Contribuições da pesquisa

Os elementos apresentados nessa pesquisa evidenciam contribuições científicas e sociais, tais como:

- Contribuir para pesquisas operacionais logísticas;
- Cooperar na ampliação bibliográfica sobre o tema;
- Propiciar conhecimento de novas abordagens aplicadas na solução do Problema de Roteamento de Veículos;
- Reduzir custos operacionais com transporte escolar e nos demais processos logísticos.

1.4 Organização do texto

Este trabalho está organizado em seis capítulos, estruturados da seguinte forma:

- **Capítulo I:** Introdução, composta por Motivação do trabalho, Objetivos gerais e específicos, Contribuições da Pesquisa e Organização do texto.
- **Capítulo II:** Referencial Teórico, composto por: *Vehicle Routing Problem*, Variantes VRP (CVRP - Problema de Roteamento de Veículos Capacitados, MDVRP - Problema de Roteamento de Veículos com Múltiplos Depósitos, VRPPD - Problema de Roteamento de Veículos com Coleta e Entrega, VRPTW - Problema de Roteamento de Veículos com Janela de Tempo, VRPMTW - Problema de Roteamento de Veículos com Múltiplas Janelas de Tempo, G-VRP - Problema de Roteamento de Veículos Verdes e VRP *Last Mile*), Soluções VRP *Last Mile*, Técnicas e Métodos aplicados ao VRP *Last Mile* (Métodos exatos, Heurísticas, Metaheurísticas, Métodos híbridos e Inteligência artificial), Grafos (Representação de grafos, Caminhos em grafos e Caminhos mínimos), Algoritmos de caminhos mínimos (Algoritmo de *Dijkstra*, *Algoritmo de Floyd-Warshall* e Algoritmo de *Bellman-Ford*, Comparação entre os Algoritmos de *Dijkstra*, *Floyd-Warshall* e *Bellman-Ford*), Matriz de distâncias, *Open Source Routing Machine*, *OpenStreetMap* e OSMnx.
- **Capítulo III:** Trabalhos relacionados, composto pela Revisão sistemática da literatura pautada nas subseções: Formulação da questão de pesquisa, Localização dos estudos, Avaliação e seleção, Análise e síntese e Relatos dos resultados.
- **Capítulo IV:** Procedimentos Metodológicos, aborda a classificação da pesquisa, a proposta da aplicação, o Modelo de arquitetura proposta, o *Dataset Benchmark Loggi BUD* e a Validação do Modelo.
- **Capítulo V:** Análise e Discussão dos dados dividido em três seções: Testes com dados do *Benchmark Loggi BUD*, Testes com dados dinâmicos e Testes com dados de transporte escolar.
- **Capítulo VI:** Considerações Finais.

Por fim, apresentamos as Referências utilizadas para desenvolvimento da pesquisa.

2 REFERENCIAL TEÓRICO

2.1 *Vehicle Routing Problem*

O problema de roteamento de veículos do inglês *Vehicle Routing Problem* (VRP), visa rotear um conjunto de veículos para atender a demanda de um conjunto de clientes. Tem como objetivo minimizar a distância total percorrida pelos veículos de forma que cada veículo inicie e termine sua rota em um determinado depósito e cada cliente deve ser visitado uma vez por um único veículo [60].

O problema de roteamento consiste em uma rede com fluxos de tráfego, iniciados em um nó de origem para um nó de destino. Os veículos dos fluxos de tráfego entram continuamente na rede nas origens, escolhem suas rotas até o destino e saem da rede no destino [67].

O VRP é um importante problema de otimização combinatória no qual busca-se encontrar a configuração ideal de rotas para uma frota de veículos que deve realizar entregas a partir de uma origem (depósito) para um determinado conjunto de clientes [9].

Podemos observar na Figura 1, o esquema gráfico do VRP clássico, com um depósito, os veículos, clientes e as rotas.

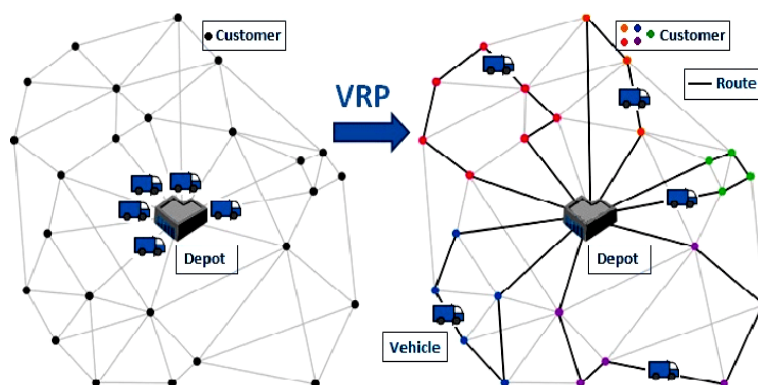


Figura 1: Esquema gráfico do VRP clássico (Gupta, A. and Saini, S., 2017) [29]

Problemas VRP são comprovadamente considerados problemas NP-difícil, embora existam algoritmos exatos capazes de encontrar soluções ótimas em um tempo razoável para instâncias relativamente pequenas, esses problemas são computacionalmente difíceis

e requerem alto custo computacional [37].

Desde a sua primeira formulação, surgiram outras variantes do VRP, que apresentam algumas restrições adicionais ao problema, e representam condições presentes nas operações logísticas.

Nos últimos anos, evidenciou-se o problema VRP *Last Mile* ou entrega na última milha. Existem algumas definições concorrentes de entrega na última milha, mas um entendimento amplamente aceito é que o termo se refere a todas as atividades de logística relacionadas à distribuição de encomendas *online*, para famílias de clientes particulares em áreas urbanas.

A entrega na última milha começa quando uma remessa chega a um ponto de partida em uma área urbana, como um depósito central por exemplo, após o transporte de longa distância, e termina quando a remessa chega com sucesso ao ponto de destino do cliente final [10].

A logística da última milha pode ser definida como a última parte do trânsito nas cadeias de abastecimento, em que as remessas são entregues dos centros de entrega até a porta dos clientes; entrega porta a porta (D2D¹) [39].

2.1.1 Variantes VRP

O Problema de Roteamento de Veículos é bastante amplo e conforme as características das demandas e necessidades dos clientes torna-se necessária a criação de variações do VRP mais adequadas a essas situações.

O surgimento das variantes VRP está relacionado à natureza dos produtos transportados, à qualidade dos serviços requeridos e às características tanto de clientes como dos veículos [3].

Abordaremos na sequência, as variantes: CVRP - Problema de Roteamento de Veículos Capacitados (*Capacited Vehicle Routing Problem*), MDVRP - Problema de Roteamento de Veículos com Múltiplos Depósitos (*Multiple Depot Vehicle Routing Problem*), VRPPD - Problema de Roteamento de Veículos com Coleta e Entrega (*Vehicle Routing Problem with Pickup and Delivery*), VRPTW - Problema de Roteamento de Veículos com Janelas de Tempo (*Vehicle Routing Problem with Time Windows*), VRPMTW - Problema de Roteamento de Veículos com Múltiplas Janelas de Tempo (*Vehicle Routing Problem with Multiple Time Windows*), G-VRP - Problema de Roteamento de Veículos Verdes (*Green Vehicle Routing Problem*) e VRP *Last Mile* - Problema de Roteamento de Veículos na última milha.

2.1.1.1 CVRP - Problema de Roteamento de Veículos Capacitados

O Problema de Roteamento de Veículos Capacitados (CVRP) é um problema clássico de otimização combinatória, que tem recebido muita atenção devido aos seus princi-

¹D2D: Acrônimo do termo em inglês *Door to Door* que significa de porta em porta

pais desafios como distribuição, logística e transporte. Visa determinar o menor custo de roteirização de veículos similares que atendem clientes com demandas conhecidas [1].

O CVRP é um problema NP-difícil amplamente investigado, que visa determinar as rotas para uma frota de veículos atender a um grupo de clientes com custo mínimo de viagem e restrição de capacidade [35].

2.1.1.2 MDVRP - Problema de Roteamento de Veículos com Múltiplos Depósitos

O Problema de Roteamento de Veículos com Múltiplos Depósitos (MDVRP) busca determinar simultaneamente as rotas para vários veículos de mais de um depósito para um conjunto de clientes e, em seguida, retornar ao mesmo depósito sem exceder a restrição de capacidade de cada veículo. Geralmente, o problema visa minimizar o custo total das rotas combinadas para uma frota de veículos [42].

O MDVRP envolve a minimização da distância total percorrida por veículos originários de vários depósitos, de modo que os veículos juntos visitem os locais especificados pelo cliente exatamente uma vez. Este problema tem sido utilizado na literatura como referência para o desenvolvimento de esquemas de otimização [50].

O MDVRP é amplamente aplicável a muitas situações do mundo real, incluindo problemas de distribuição logística para otimizar os custos totais de transporte. Afinal, uma rota ótima pode minimizar a distância total de cada rota, levando a economia de custos [47].

2.1.1.3 VRPPD - Problema de Roteamento de Veículos com Coleta e Entrega

O Problema de Roteamento de Veículos com Coleta e Entrega é uma extensão do VRP e consequentemente também é classificado como NP-difícil. Nessa variante os veículos além da entrega de produtos para os clientes também realizam coletas.

O VRPPD consiste no transporte de cargas de consumidor a consumidor. Tem como objetivo reduzir a distância total percorrida pela frota de veículos [48].

Menezes (2019) conceitua o VRPPD como:

Essa versão se trata de outra extensão do CVRP e nela cada cliente possui duas quantidades, representando a demanda de mercadorias que devem ser entregues e coletadas, nessa ordem. Para cada vértice i , O_i representa o vértice que é a origem da demanda de entrega e D_i representa o vértice que é o destino da retirada [46].

2.1.1.4 VRPTW - Problema do Roteamento de Veículos com Janela de Tempo

O VRPTW é uma variação do VRP clássico, um problema no qual uma frota de veículos parte de um depósito central com o objetivo em atender um conjunto de clientes com suas respectivas demandas. Cada cliente define o horário para receber sua mercadoria, ou seja, a janela de tempo em que deve ser atendido [62].

Esta variante é relevante para situações com longos tempos de serviço quando comparada com os tempos de viagem e é derivada de aplicações da vida real para as quais os pedidos diários devem ser entregues no mesmo dia, a operação total não pode ser concluído dentro do tempo máximo de roteamento e as violações a este último são altamente indesejáveis [55].

Em geral, um VRPTW refere-se à obtenção de um conjunto de rotas para uma frota de veículos para fornecer os serviços a um número de clientes com janelas de tempo pré-definidas e demandas conhecidas. Cada cliente deve ser atendido por apenas um veículo dentro da janela de tempo específica, e a demanda total de cada rota não deve exceder a capacidade máxima do veículo de serviço. O objetivo do VRPTW é encontrar rotas que minimizem os custos de entrega sob todas as restrições [12].

2.1.1.5 VRPMTW - Problema do Roteamento de Veículos com Múltiplas Janelas de Tempo

O Problema de Roteamento de Veículos com Múltiplas Janelas de Tempo é uma generalização do Problema de Roteamento de Veículos, onde os clientes têm uma ou mais janelas de tempo nas quais eles podem ser visitados.

O Problema do Roteamento de Veículos com Múltiplas Janelas de Tempo consiste em, dado um conjunto de veículos com a mesma capacidade, um conjunto de clientes com demandas, um tempo de serviço e uma ou mais janelas de tempo, encontrar um conjunto de rotas que começam e terminam em um único depósito, onde todos os clientes sejam visitados uma única vez e o tempo de serviço inicie a partir de uma das respectivas janelas de tempo.

O VRPMTW objetiva minimizar uma função de custo que combina o tamanho da frota utilizada e o tempo gasto para visitar todos os clientes e retornar ao depósito [8].

2.1.1.6 G-VRP - Problema de Roteamento de Veículos Verdes

Devido ao cumprimento de regulamentos ambientais e políticas energéticas, novos modelos de roteirização de veículos são propostos para questões de logística sustentável. O problema de roteamento de veículos verdes é denotado como uma Variante VRP com consideração dos efeitos ambientais das estratégias logísticas [38].

O G-VRP visa considerar a redução de emissões de gases de efeito estufa, durante o roteamento dos veículos. Pode ser por meio da adoção de Veículos de Combustível Alternativo ou com veículos convencionais de combustível fóssil existentes nas frotas. O G-VRP também leva em consideração a sustentabilidade ambiental no transporte e na logística [59].

2.1.1.7 VRP *Last Mile*

A distribuição urbana de cargas é um dos principais responsáveis pelo trânsito urbano, degradação ambiental, consumo de combustíveis e pela congestão do espaço público. O VRP *Last Mile* é uma das principais abordagens para otimizar a distribuição de cargas na última milha, com objetivo de minimizar a distância e o custo de entrega.

Na entrega da última milha, os pacotes são entregues de filiais locais de empresas a clientes individuais. A entrega da última milha também inclui as solicitações de coleta e entrega dentro da cidade [13].

O problema da entrega na última milha pode ser observado no esquema apresentado na Figura 2, que contém um depósito central onde são preparados os pacotes e vários clientes distribuídos ao redor do depósito. Uma frota de veículos visita os clientes e retorna ao depósito central após a entrega de todos os pacotes.

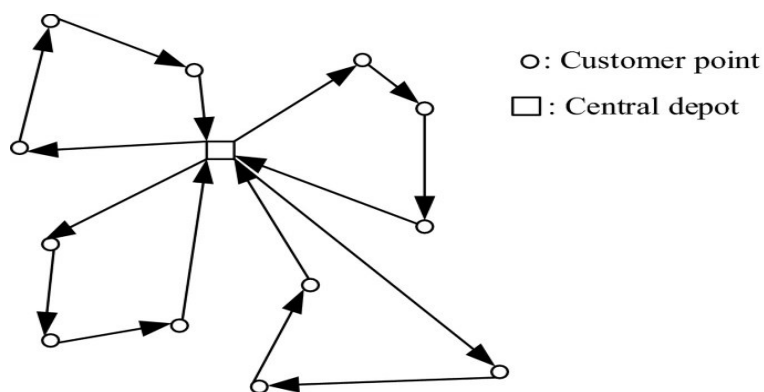


Figura 2: Esquema gráfico da entrega *Last Mile* (Chu, H., Zhang, W., et al., 2021) [16]

2.2 Soluções VRP *Last Mile*

Pesquisas operacionais que buscam otimizar o VRP *Last Mile*, cresceram nos últimos anos, existindo uma variedade de trabalhos com diferentes abordagens.

Os pesquisadores Cai, Zhang, et al. (2018), empregaram o *Algoritmo* de Busca Local Iterada (*ILS - Iterated Local Search*). O *algoritmo* proposto consiste em uma sequência de buscas locais intercaladas por perturbações. As perturbações são baseadas na metaheurística de Busca Adaptativa de Grande Vizinhança (*ALNS - Adaptive Large Neighborhood Search*). Foram utilizadas quatro instâncias de teste, obtendo uma eficácia de 75% [13].

O *Algoritmo* da otimização da Colônia de Formigas (*ACO - Ant Colony Optimization Algorithm*) foi empregado por Zhang e Lee (2016) para resolver o VRP na última milha integrado às opções de entrega a domicílio e ponto de coleta compartilhado, obtendo uma economia em relação a distância a ser percorrida e duração da viagem de 20,87% e 33,99%, respectivamente [70].

Jiang, Zang, et al. (2021), propuseram uma heurística de duas fases baseada em VNS (*Variable Neighborhood Search*) pesquisa de vizinhança, a primeira fase objetivava construir uma solução viável, e a segunda fase era destinada a otimizar a solução encontrada. O algoritmo foi avaliado em 54 instâncias divididas em dois conjuntos, cada um deles com 27 instâncias. O primeiro grupo com instâncias de pequeno porte continha de 51 a 101 nós, e o segundo de médio porte continha de 130 a 200 nós. O algoritmo proposto teve uma eficácia de 98,15%, tendo conseguido encontrar a melhor solução para 53 instâncias [36].

Uma abordagem de otimização baseada em um conjunto de dados reais que combina técnicas de aprendizado de máquina com otimização de roteamento de veículos capacitados, foi proposta por Chu, Zhang, et al. (2021), visando enfrentar os desafios da entrega na última milha [16].

2.3 Técnicas e Métodos aplicados ao VRP *Last Mile*

Neste trabalho, as técnicas e métodos utilizados para resolver o VRP *Last Mile* foram categorizados como: Métodos Exatos, Heurísticas, Metaheurísticas, Técnicas de Inteligência Artificial e Métodos híbridos.

2.3.1 Métodos exatos

Métodos Exatos são métodos matemáticos utilizados para resolver problemas de otimização, buscam a melhor solução para o problema, o que não significa que essa seja a solução mais satisfatória. São utilizados em problemas de pequena ordem, pois, o tempo de processamento computacional torna-se elevado em problemas de ordem maior [30]. Como exemplo de métodos exatos podemos citar: Método *Branch and Bound*, Método

de Decomposição de *Benders* e *Branch-and-Cut*.

2.3.2 Heurísticas

Heurísticas consistem em um conjunto de regras e métodos que conduzem à descoberta, à invenção e à resolução de problemas. Assim como os métodos exatos, fazem uso de métodos matemáticos, no entanto, buscam a solução mais satisfatória [68].

As heurísticas mais utilizadas para resolver o problema VRP são classificadas em: Heurísticas Construtivas, Heurísticas de Duas Fases e Heurísticas de Melhoramento.

- **Heurísticas Construtivas:** São caracterizadas por atuarem de forma gulosa, por meio da formação de um conjunto de soluções definitivas, como a inserção de um nó ou a fusão de duas rotas que não podem ser revertidas posteriormente [45]. As heurísticas de construção de rotas usualmente selecionam nós sequencialmente até que uma solução viável seja criada. Os nós são escolhidos, geralmente, segundo algum critério de minimização de custo desde que não violem a capacidade dos veículos [41].

Algumas heurísticas construtivas mais conhecidas são: o Algoritmo de *Clarke & Wright*, o Método *Cluster-first Route Second* e o Método *Route-first Cluster-second*.

- **Heurísticas de Duas Fases:** Nesse caso, o problema é dividido em duas fases (agrupamento e roteamento). Inicialmente, os clientes são divididos em grupos que posteriormente serão classificados por rotas.
- **Heurística de Melhoramento:** Os algoritmos de busca local são frequentemente usados para melhorar as soluções iniciais geradas por outras heurísticas. A partir de uma determinada solução, um método de busca local aplica modificações simples, como trocas de arco ou movimentos de clientes, para obter soluções vizinhas de custo possivelmente melhor. Se uma solução melhor for encontrada, se tornará a solução atual e o processo será iterado; caso contrário, um mínimo local foi identificado [68].

2.3.3 Metaheurísticas

Metaheurísticas consistem na aplicação e combinação de heurísticas, que são aplicadas em cada passo da resolução do problema a fim de resolver um problema específico. Heurísticas são modificadas e guiadas, buscando encontrar boas soluções[26].

Uma metaheurística é um conjunto de conceitos que podem ser usados para definir os métodos heurísticos que podem ser aplicados a um vasto conjunto de problemas diferentes. Em outras palavras, uma metaheurística pode ser vista como um quadro geral de algoritmos que

podem ser aplicados a diferentes problemas de otimização com relativamente poucas modificações para torná-los adaptados a um problema específico (BENEVIDES, 2011, p.31)[6].

2.3.4 Métodos híbridos

Os Métodos Híbridos fazem uso de mais de um método de solução, é a junção de dois ou mais métodos, que buscam encontrar uma solução otimizada.

2.3.5 Inteligência Artificial

A Inteligência Artificial (IA²), é a possibilidade de uma máquina, por meio de algoritmos, adquirir capacidade cognitiva semelhante ao ser humano e realizar atividades que antes apenas seres humanos eram capazes [64].

“A Inteligência Artificial é um ramo da Ciência da Computação cujo interesse é fazer com que os computadores pensem ou se comportem de forma inteligente [28].”

Um sistema de IA deve ser capaz de fazer três coisas: armazenar conhecimento, aplicar o conhecimento armazenado para resolver problemas e adquirir conhecimento por meio da experiência. É composto por representação, raciocínio e aprendizagem [31].

2.4 Grafos

Um grafo $G = (V, E)$ é uma estrutura de dados formada por vértices conhecidos também como nódulos ou nós e por arestas também chamadas de arcos. Os vértices são representados por pontos e as arestas são representadas por ligações entre os pontos, como pode ser observado na Figura 3.

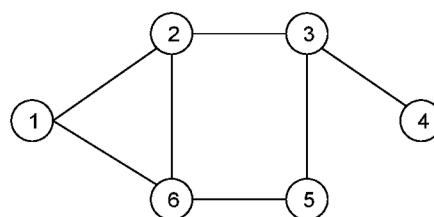


Figura 3: Exemplo de grafo

Conforme Szwarcfiter (1984 apud BARROSO, 2014, p. 52), um grafo é uma estrutura matemática constituída de um conjunto V , finito e não vazio de n vértices, e um conjunto E de m arestas, que são pares não ordenados de elementos de V [5].

²IA - Intelligence Artificial

Grafos podem ser direcionados também conhecidos como grafos orientados ou dígrafo, ou não direcionados. Grafos direcionados, possuem sentido indicado na aresta por uma seta. Grafos não direcionados, por sua vez, apresentam conjunto de arestas constituído de pares de vértices não ordenados. Na Figura 4 é apresentada essa diferença entre grafos direcionados e não direcionados.

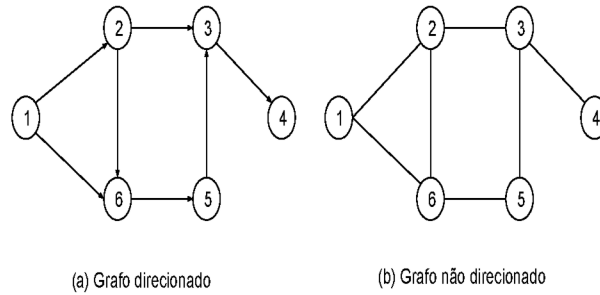


Figura 4: Exemplo de grafo direcionado e grafo não direcionado

2.4.1 Representação de grafos

Além da representação gráfica baseada em vértices e arestas, grafos podem ser representados por listas e matrizes de adjacências. A seguir, cada uma das abordagens são discutidas.

A representação por lista de adjacência apresenta um modo compacto de representar grafos esparsos, pode ser construída atribuindo-se a cada vértice um conjunto de vértices com os quais está conectado. Exemplificamos essa estrutura na Figura 5.

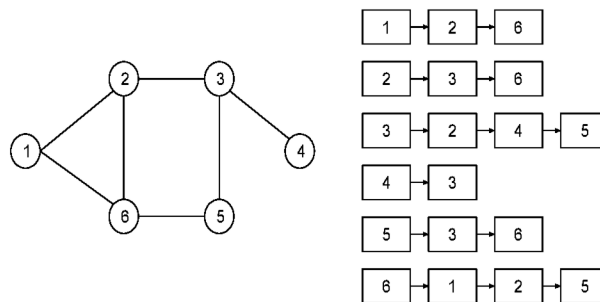


Figura 5: Grafo não direcionado e lista de adjacências correspondente

Uma matriz de adjacência pode ser representada em forma de uma tabela com número de linhas e colunas igual ao número de vértices do grafo. Os números presentes na primeira coluna reproduzem os vértices de origem e os números da primeira linha os vértices de destino.

Do cruzamento entre origem e destino, se houver uma aresta, a posição é preenchida com o peso da aresta, que neste caso é 1. Caso não exista uma aresta, a posição é preenchida com 0, conforme pode ser observado na Figura 6.

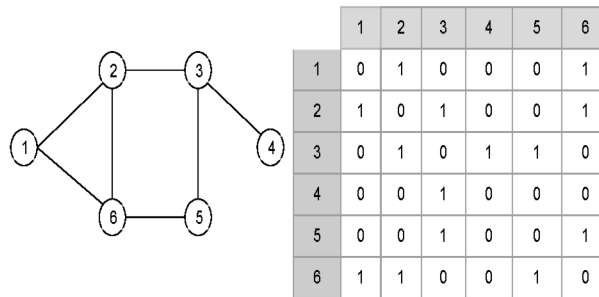


Figura 6: Grafo não direcionado e matriz de adjacências correspondente

2.4.2 Caminhos em grafos

Um caminho entre dois vértices de um grafo é uma sequência de vértices, onde cada vértice está conectado ao vértice seguinte por meio de uma aresta.

Rangel et al. (2018), definem caminho em um grafo como:

Dado um grafo $G(V, A)$, um caminho em G consiste de uma sequência finita alternada de vértices e arestas, começando e terminando por vértices, tal que cada aresta é incidente ao vértice que a precede e que a sucede e não há repetição de vértices. Em outras palavras, um caminho é um trajeto onde não há repetição de vértices [56].

Brito (2014), conceitua caminho como uma sucessão de vértices e arestas, de forma que cada aresta realize a ligação do vértice que a precede ao vértice que a segue, não repetindo arestas. O grafo que contém um único vértice, e nenhuma aresta, também é um caminho [11].

Na Figura 7 exemplificamos uma opção de caminho disponível, partindo-se do vértice 1 para o vértice 5 do grafo. O caminho percorrido nesse exemplo é 1 - 6 - 5.

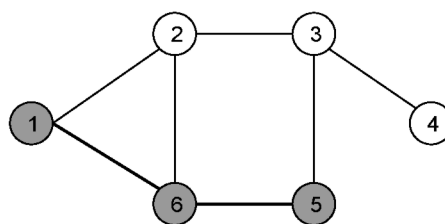


Figura 7: Exemplo de caminho em um grafo

2.4.2.1 Caminhos mínimos

O problema do caminho mínimo consiste na minimização do custo de travessia entre dois vértices de um grafo, obter o caminho de menor custo. O valor do custo é a soma dos pesos de cada aresta percorrida, geralmente, o peso das arestas é a distância entre os vértices, por isso, passou a ser conhecido como problema do caminho mínimo.

O problema do caminho mínimo está associado a várias aplicações nas áreas de roteamento, manufatura, fluxo em redes, dentre outras. Seu principal objetivo é minimizar os esforços de busca. Sendo u e v dois vértices de um grafo $G=(V,A)$, o caminho mínimo entre u e v é uma sequência de arcos que, passando por nós distintos, une u a v de forma a acumular o menor comprimento (distância). Para que haja um caminho mínimo entre dois arcos u e v é necessário que exista uma conexão entre eles, isto é, v seja sucessor de u . Na maioria dos casos, existe mais de um caminho entre dois nós específicos; com isso, o problema do caminho mínimo consiste em encontrar o caminho com menor custo dentre todos os possíveis (HERNANDES, F., BERTON, L., et al., 2009, p. 476) [32].

2.5 Algoritmos de Caminhos mínimos

Nesta seção, apresentamos três dos principais algoritmos de caminhos mínimos estudados na literatura: Algoritmo de *Dijkstra*, Algoritmo de *Floyd-Warshall* e Algoritmo de *Bellman-Ford*.

2.5.1 Algoritmo de *Dijkstra*

O Algoritmo de *Dijkstra* foi desenvolvido em 1959 pelo Cientista da Computação *Edsger Wybe Dijkstra*, utilizado para determinar o caminho mínimo entre dois vértices [4].

O Algoritmo de *Dijkstra* é iterativo e fornece uma solução eficiente para o problema do caminho mínimo, encontrando melhor caminho em um grafo (ponderado e dirigido), do nó de origem ao nó de destino, no entanto, todas as arestas devem ter custos não negativos³ [15].

O algoritmo de *Dijkstra* é o mais conhecido e o mais empregado dos algoritmos para cálculo de caminho de custo mínimo entre uma origem e os demais vértices de um grafo. Tem como objetivo definir um vértice como ponto inicial e calcular o caminho de custo mínimo para percorrer todo o grafo [20].

Escolhido um nó como origem, este algoritmo encontra o caminho mínimo deste nó para todos os demais nós da rede, parte de uma estimativa inicial para o caminho mínimo e vai sucessivamente ajustando esta estimativa, considera que um nó estará fechado quando já tiver sido obtido um caminho mínimo [63].

O algoritmo baseia-se em um processo de rotulação dos vértices do grafo e classificação dos respectivos rótulos. A cada vértice v_i é atribuído um rótulo $(p(v), d(v_i))$, onde $d(v)$ é a distância atual de s até v e $p(v)$ predecessor de v_i , que pode ser permanente ou temporário. Isto é, o rótulo temporário de um vértice representa um limite superior da distância mais curta de s para esse vértice, uma vez que o caminho que lhe está associado

³Custos não negativos: o peso das arestas deve ser positivo.

pode ser ou não o mais curto.

O algoritmo consiste em um processo de fixação dos rótulos dos vértices do grafo. Começando por s , o grafo é ordenado conforme as distâncias de cada vértice a s . Em cada iteração, é escolhido um vértice v_i com rótulo temporário de menor valor $d(v)$ que se torna permanente, para depois varrer todos os vértices adjacentes a v_i , para atualização de seus rótulos. O algoritmo termina quando não existirem vértices com rótulos temporários ou quando o rótulo do último vértice passar a permanente [51].

O Pseudocódigo presente na Figura 8, sintetiza o funcionamento do Algoritmo de *Dijkstra*.

```

1 Dijkstra(s){
2   para cada v em G
3     d[v] = infinito
4   fimpara
5   d[s] = 0
6   Q = V — Conjunto de vértices que a distância ainda não foi calculada
7
8   enquanto Q != vazio
9     u = um vértice em Q, no qual d[u] é a distancia mínima
10    se u == t
11      return
12
13    para cada v adjacente a u
14      se v está em Q
15        e = aresta(v,u)
16        se d[v] > d[u] + l[e]
17          d[v] = d[u] + l[e]
18        fimse
19      fimse
20    fimpara
21    Q.remove(u)
22  fimenquanto
23 }

```

Figura 8: Pseudocódigo do algoritmo de *Dijkstra* (Zanatto, 2021) [69]

Na Figura 9 é apresentado um exemplo de uso do algoritmo de *Dijkstra*, no qual o vértice 1 é a origem do percurso e o vértice 6 o destino. Além disso, apresentamos a matriz de distâncias mínimas obtidas.

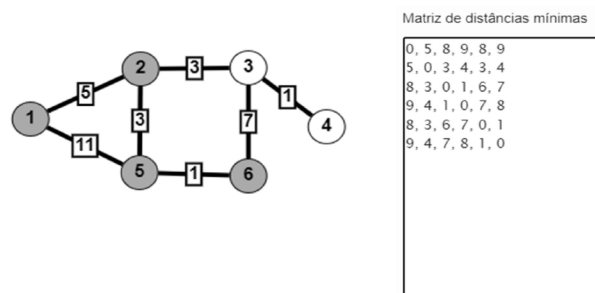


Figura 9: Exemplo de uso do algoritmo de *Dijkstra* e matriz de distâncias

2.5.2 Algoritmo de *Floyd-Warshall*

O algoritmo *Floyd-Warshall* publicado por *Robert Floyd* em 1962 [21], consiste em um algoritmo simples, utilizado para calcular caminhos mais curtos entre todos os pares de vértices em um grafo direcionado ponderado por arestas. Ao contrário do Algoritmo

de *Dijkstra* pode ser usado com ciclos negativos⁴ [33].

Em geral, o algoritmo *Floyd-Warshall* baseia-se em algoritmo de programação dinâmica, um método que busca soluções de problemas, por meio do resultado de decisões inter-relacionadas. Combinam-se respostas obtidas em etapas anteriores com novas alternativas possíveis, obtendo-se assim uma nova solução [66].

Na Figura 10, podemos visualizar um exemplo de aplicação do algoritmo de *Floyd-Warshall* e a matriz de distâncias mínimas encontradas. Pode-se observar que conforme mencionado, esse algoritmo calcula o caminho mínimo entre todos os pares de vértices do grafo.

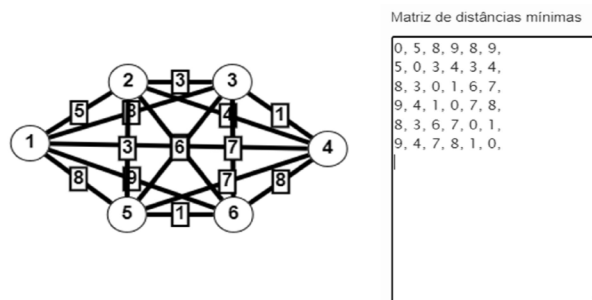


Figura 10: Exemplo de utilização do algoritmo de *Floyd-Warshall* e matriz de distâncias mínimas

A idéia geral do algoritmo de *Floyd-Warshall*, segundo Sampaio (2005), é atualizar a matriz de menores distâncias n vezes (onde n é o número de nós do grafo) buscando na K -ésima interação por melhores distâncias entre pares de nós que passem pelo vértice K [61].

A Figura 11, exemplifica por meio de pseudocódigo o fluxo de funcionamento do algoritmo de *Floyd-Warshall*.

```

ROTINA fw(Inteiro[1..n,1..n] grafo)
# Inicialização
VAR Inteiro[1..n,1..n] dist := grafo
VAR Inteiro[1..n,1..n] pred
PARA i DE 1 A n
  PARA j DE 1 A n
    SE dist[i,j] < Infinito ENTÃO
      pred[i,j] := i
# Laço principal do algoritmo
PARA k DE 1 A n
  PARA i DE 1 A n
    PARA j DE 1 A n
      SE dist[i,j] > dist[i,k] + dist[k,j] ENTÃO
        dist[i,j] = dist[i,k] + dist[k,j]
        pred[i,j] = pred[k,j]
RETORNE dist

```

Figura 11: Pseudocódigo do algoritmo de *Floyd-Warshall* (2021)

⁴Ciclos negativos: aquele em que a soma total do ciclo é negativa, ou seja, a soma dos pesos das arestas envolvidas no ciclo é negativa

2.5.3 Algoritmo de *Bellman-Ford*

O algoritmo de *Bellman-Ford*, criado por *Richard Bellman* e *Lester Ford Jr.* é responsável por calcular, para um dado dígrafo (grafo orientado) G com arestas ponderadas, o menor caminho de um nó de origem s até cada um dos outros nós de G [18].

O algoritmo ficou reconhecido como *Bellman-Ford* ou *Ford-Moore-Bellman*, devido às contribuições dos pesquisadores *Lester Ford*, *Edward Moore* e *Richard Bellman*. O algoritmo de *Bellman-Ford* difere-se do algoritmo de *Dijkstra* que após selecionado um vértice, não calcula novamente a sua distância, por sua vez, o algoritmo de *Bellman-Ford* não determina que a menor distância foi atingida. Possibilitando assim, calcular caminhos mínimos em grafos que tenham arestas com pesos negativos [27].

Demonstramos na Figura 12, um exemplo de utilização do algoritmo de *Bellman-Ford* em um grafo com quatro vértices, com a presença de arestas de pesos negativos, partindo da origem no vértice A até o destino no vértice D.

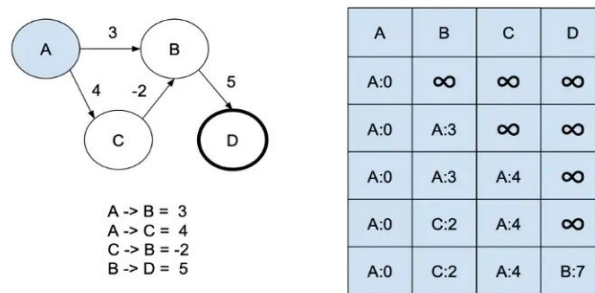


Figura 12: Exemplo execução do algoritmo de *Bellman-Ford* (Stevenard, 2022)[65]

2.5.4 Comparação entre os Algoritmos de *Dijkstra*, *Floyd-Warshall* e *Bellman-Ford*

Com a finalidade de facilitar o entendimento dos algoritmos de *Dijkstra*, *Floyd-Warshall* e *Bellman-Ford*, elaboramos um quadro comparativo dos três algoritmos, contendo a complexidade, as estruturas principais, a aplicabilidade do algoritmo, as limitações e um resumo de como cada um deles funciona.

A Figura 13 apresenta a estrutura do quadro mencionado.

Algoritmo	Complexidade	Estrutura	Resumo	Aplicabilidade	Limitações
<i>Dijkstra</i>	$O((n + m) \log n)$	Filas/Heap/ Relaxamento	Começa no nó definido como origem e analisa o grafo para encontrar o caminho de menor custo entre esse nó e todos os outros nós do grafo. O algoritmo mantém o registro da distância mais curta atualmente conhecida de cada nó até o nó de origem e atualiza esses valores se encontrar um caminho de menor custo. Uma vez que o algoritmo encontrou o caminho de menor custo entre o nó de origem e outro nó, esse nó é marcado como "visitado" e adicionado ao caminho. O processo continua até que todos os nós do grafo tenham sido adicionados ao caminho. Desta forma, temos um caminho que conecta o nó de origem a todos os outros nós seguindo o caminho de menor custo possível para chegar a cada nó.	Cálculo de caminhos mínimos de um vértice para todos os outros.	Não consegue trabalhar com ciclos negativos
<i>Floyd-Warshall</i>	$O(n^3 \log n)$	Matrizes	Calcula a distância de todos os vértices para todos os outros, permitindo passar por K outros vértices. O valor de K é incrementado a cada iteração e uma nova matriz com distâncias com a passagem de K vértices é obtida ao final de cada execução. Os valores começam de infinito. E termina-se quando todos os K vértices que compõem o grafo podem ser utilizados.	Calcula o caminho mínimo de todos os vértices para todos os outros	Funciona somente para matrizes de adjacência. Não consegue trabalhar com arestas negativas.
<i>Bellman-Ford</i>	$O(nm)$	Relaxamento	Assim como o algoritmo de Dijkstra, o algoritmo de Bellman-Ford utiliza a técnica de relaxamento, ou seja, realiza sucessivas aproximações das distâncias até finalmente chegar na solução. A principal diferença entre Dijkstra e Bellman-Ford é que no algoritmo de Dijkstra é utilizada uma fila de prioridades para selecionar os vértices a serem relaxados, enquanto o algoritmo de Bellman-Ford simplesmente relaxa todas as arestas.	Busca de caminho mínimo em um dígrafo ponderado. Calcula o menor caminho de única origem.	Mais lento que o <i>dijkstra</i>

Figura 13: Quadro comparativo dos algoritmos de *Dijkstra*, *Floyd-Warshall* e *Bellman-Ford*

Conforme especificado na Figura 13, os algoritmos de *Dijkstra* e *Bellman-Ford* possuem características semelhantes. No entanto, o fato do algoritmo de *Bellman-Ford* aplicar o relaxamento a todos os vértices o torna mais lento na execução.

2.6 Matriz de distâncias

Uma matriz de distâncias é uma matriz, ou seja, um *array* bidimensional contém as distâncias tomadas em pares de um conjunto de pontos. Esta matriz terá um tamanho de $N \times N$ onde N é o número de pontos, nós ou vértices.

Matrizes de distância estão relacionadas com matrizes de adjacência. Matrizes de adjacências indicam se há ligação entre os vértices e os custos ou distâncias entre esses vértices.

2.7 Open Source Routing Machine (OSRM)

O *Open Source Routing Machine* (OSRM) é um ferramenta moderna para roteamento, desenvolvida em C++, possibilita criar rotas, estatísticas de tempo e caminhos mais curtos em redes rodoviárias.

O OSRM pode ser usado como uma biblioteca via C++ (*libosrm*) ou através da interface HTTP e *osrm-routed*. O uso via *libosrm* permite o ajuste fino do OSRM e com menos sobrecarga. Como objetos importantes da interface OSRM podemos citar: *EngineConfig*, *Status*, *TableParameters*, *BaseParameter*, *Coordinate*, *Parameters for other services* e *JSON* [53].

O *EngineConfig* é utilizado para configurar propriedades e restrições antes de inicia-

lizar uma instância OSRM.

O OSRM apresenta as principais funcionalidades para roteamento, o *Route* e o *Table*. O *Status* é responsável por indicar o estado da solicitação, retornando *Error* para indicar que algum erro aconteceu ou OK para indicar sucesso na requisição.

2.8 OpenStreetMap (OSM)

O *OpenStreetMap* (OSM) é um projeto de construção de um banco de dados geográficos gratuito, com mapeamento colaborativo. Conta com colaboradores voluntários e empresas governamentais, responsáveis por desenvolver a base de dados, manipulá-las e mantê-las. Os dados são obtidos, percorrendo-se áreas e gravando os movimentos com GPS, para criação e atualização de mapas [52].

Por meio do OSM é possível exportar dados de forma simples e prática. Basta pesquisar o local desejado e selecionar a região a ser exportada. Há diferentes formatos de dados disponíveis no *OpenStreetMap*, sendo os principais: o OSM XML e OSM JSON. Na Figura 14, apresentamos a interface do *OpenStreetMap*.

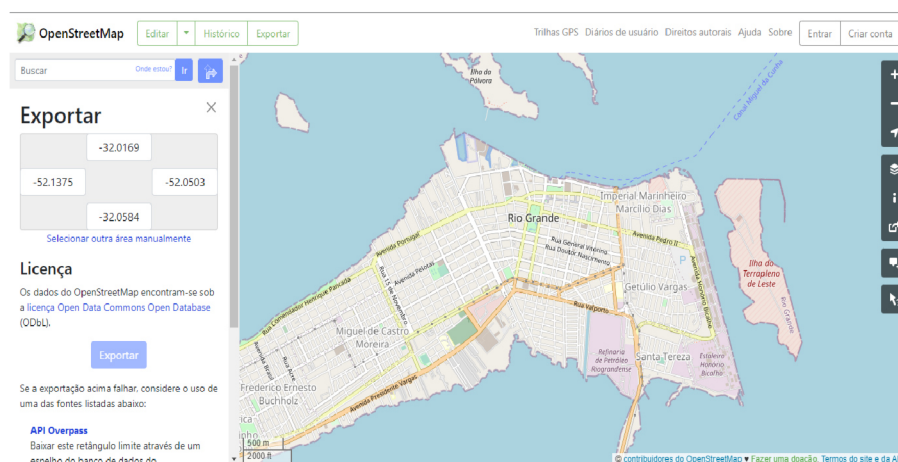


Figura 14: Interface *OpenStreetMap*

2.9 OSMnx

O OSMnx é um pacote *Python* que permite baixar dados geoespaciais do *OpenStreetMap*. Além disso, permite modelar, projetar, visualizar e analisar redes de ruas.

O OSMnx é construído sobre o *GeoPandas*⁵, *NetworkX*⁶ e *Matplotlib*⁷ e interage com APIs *OpenStreetMap* para: baixar e modelar redes de ruas ou outras infraestruturas de rede em qualquer lugar do mundo; baixar quaisquer outras geometrias espaciais; baixar

⁵GeoPandas: Projeto de software livre que facilita o trabalho com dados geoespaciais em python.

⁶NetworkX: Pacote python para criação, manipulação e estudo da estrutura, dinâmica e funções de redes complexas.

⁷Matplotlib: Biblioteca para criar visualizações estáticas, animadas e interativas em Python.

por nome da cidade, polígono, caixa delimitadora ou ponto/endereço e distância da rede; fazer o *download* de redes dirigíveis, caminháveis, de bicicleta ou todas as redes de rua; fazer o *download* das elevações dos nós e calcular os graus das arestas (inclinações); atribuir velocidades ausentes e calcular os tempos de deslocamento da borda do gráfico; simplificar e corrigir a topologia da rede para limpar os nós e consolidar as interseções; salvar redes em disco como *shapefiles*, *GeoPackages* e *GraphML*; salvar/carregar a rede viária de/para um arquivo XML .osm local e entre outras [7].

3 TRABALHOS RELACIONADOS

Para apresentar os trabalhos relacionados com essa dissertação de mestrado, uma Revisão Sistemática da Literatura (RSL) foi realizada.

3.1 Revisão Sistemática da Literatura

Revisões Sistemáticas apresentam uma questão explícita, definida claramente, uma pesquisa abrangente e sistemática de estudos, uma estratégia reprodutível explícita para o rastreamento e inclusão de estudos, uma extração de dados reprodutível explícita, análise apropriada e apresentação dos resultados, interpretações apoiadas por dados, e implicações para futuras pesquisas [57].

O processo sistemático respaldou-se no método proposto por *Denyer e Tranfield* em 2009 [19], que dividem a RSL em 5 etapas: (i) Formulação da Questão de Pesquisa, (ii) Localização dos estudos, (iii) Avaliação e seleção dos Estudos, (iv) Análise e Síntese, e (v) Relato e Uso de Resultados. A Figura 15, apresenta as fases da RSL, que compõe o protocolo de revisão sistemática desse trabalho, discutido em detalhes nas próximas seções.

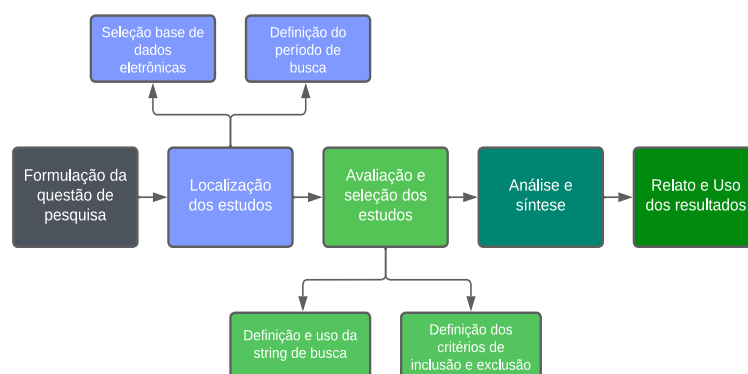


Figura 15: Fases da RSL, conforme modelo proposto por *Denyer e Tranfield*

3.1.1 Formulação da questão de pesquisa

Nesta fase, propusemos uma questão de pesquisa e algumas subquestões, responsáveis por nortear o estudo. Para tal, foi elaborada a seguinte questão de pesquisa: “Quais os métodos e soluções são utilizados para otimizar o problema de roteamento de veículos nas entregas na última milha e quais os fatores limitadores dos métodos propostos?”. As subquestões são definidas por:

Questão 1: Quais os métodos propostos na literatura para resolver o problema de roteamento de veículos?

Questão 2: Quais os principais métodos para solução do problema de roteamento de veículos nas entregas de última milha?

Questão 3: Quais os principais limitadores dos métodos propostos?

Na sequência, selecionamos dentre os repositórios de pesquisas científicas eletrônicas, aqueles mais relevantes e com relação com o tema pesquisado.

3.1.2 Localização dos estudos

As bases de dados escolhidas, consistem em repositórios nacionais e internacionais de Artigos de Periódicos, Artigos de Conferências, Teses de Doutorado e Dissertações de Mestrado, as bases de dados selecionadas constam na Tabela 01.

Tabela 1: Repositórios utilizados na pesquisa

Repositórios Internacionais	Link de acesso
ACM Digital Library	https://dl.acm.org/
IEEE Xplorer	https://ieeexplore.ieee.org/Xplore/home.jsp
Science Direct	https://www.sciencedirect.com/
Springer Link	https://link.springer.com/
Wiley Online Library	https://onlinelibrary.wiley.com/
Repositórios Nacionais	Link de acesso
BDTD	https://bdtd.ibict.br/vufind/

Para realizar a busca automática nesses repositórios, usamos as palavras-chave: (*solution*, *vehicle routing problem*, *last mile*), e os sinônimos: (*application*, *optimization*, *vehicle routes problem*, *vehicle itinerary problem*), bem como, para otimizar a busca, utilizamos os conectores lógicos (*AND* e *OR*). As *strings* simplificadas e a *string* resultante são:

1. (“*solution*” *OR* “*optimization*” *OR* “*application*”) *AND* (“*vehicle routing problem*” *OR* “*vehicle routes problem*” *OR* “*vehicle itinerary problem*”);
2. (“*vehicle routing problem*” *OR* “*vehicle routes problem*” *OR* “*vehicle itinerary problem*”) *AND* “*last mile*”;

String Resultante: (((“*solution*” OR “*optimization*” OR “*application*”) AND (“*vehicle routing problem*” OR “*vehicle routes problem*” OR “*vehicle itinerary problem*”)) AND “*last mile*”).

3.1.3 Avaliação e seleção

A busca contemplou trabalhos em Língua Portuguesa e Inglesa, no período compreendido entre os anos de 2016 a 2021. Somente foram incluídos nessa pesquisa, Artigos de Periódicos, Artigos de Conferências, Teses de Doutorado e Dissertações de Mestrado.

Aplicou-se a *string* de busca em cada um dos repositórios de dados selecionados, levando-se em conta os metadados: título, resumo, palavras-chave, e os critérios mencionados anteriormente. Obtivemos os resultados e suas constantes, conforme a Tabela 02.

Tabela 2: Resultado inicial da pesquisa nos repositórios de dados.

Repositórios Internacionais	Total de estudos
<i>Science Direct</i>	593
<i>Springer Link</i>	588
<i>Wiley Online Library</i>	65
<i>IEEE Xplorer</i>	39
<i>ACM Digital Library</i>	17
Repositórios Nacionais	Total de estudos
BDTD	85
Total	1387

Ademais, salienta-se que no repositório brasileiro BDTD, realizamos buscas com *strings* simplificadas para expandir os resultados.

As pesquisas iniciais, sem a aplicação de filtros e critérios, nos retornaram um total de 1387 estudos, visando garantir a qualidade da pesquisa, aplicamos alguns critérios de inclusão e exclusão, apresentados a seguir:

- Inclusão de artigos escritos em Língua Portuguesa e Inglesa;
- Exclusão de artigos sem autoria;
- Exclusão de artigos introdutórios para questões especiais;
- Exclusão de trabalhos em livros, capítulos de livros e *Workshops*;
- Exclusão de artigos duplicados em diferentes fontes de dados; e
- Exclusão de artigos que não possuem relação com o tema da pesquisa.

Após, o refinamento da busca foi utilizado os critérios de inclusão e exclusão, contamos com um total de 103 estudos selecionados. A Base de Dados *Science Direct*, apresenta maior contribuição para a pesquisa com um total de 36 trabalhos. A relação de trabalhos pertencentes aos demais repositórios, pode ser observada na Tabela 03.

Tabela 3: Resultado da pesquisa nos repositórios pesquisados após os critérios.

Repositórios Internacionais	Total de estudos
<i>Science Direct</i>	36
<i>Springer Link</i>	24
<i>IEEE Xplorer</i>	14
<i>ACM Digital Library</i>	07
<i>Wiley Online Library</i>	00
Repositórios Nacionais	Total de estudos
BDTD	22
Total	103

3.1.4 Análise e síntese

Os resultados obtidos por meio da *string* de busca nos repositórios de dados científicos e da posterior seleção de estudos foram exportados para a ferramenta *Zotero*¹, responsável por auxiliar no processo de organização, padronização dos dados, citação e compartilhamento da pesquisa científica.

Na análise dos artigos, apresentamos o percentual de publicações por repositório, a quantidade de publicações por ano, os tipos das fontes de publicação, as palavras-chave, o quantitativo de autores, as instituições de vínculo e os países de origem.

Sintetizamos as publicações, dividindo as produções em categorias, conforme as respostas para as questões problema, o que permitiu a sistematização das soluções abordadas nos trabalhos.

3.1.5 Relatos dos resultados

As buscas realizadas nas bases de dados científicas, inicialmente, apresentaram um total de 1387 trabalhos. Sendo que destes, a maior parte pertence a Base de Dados *Science Direct*, representam 43% do total, conforme disposto na Figura 16.

¹Zotero: Assistente de pesquisa gratuito que auxilia na coleta, organização, citação e compartilhamento, disponível em: <https://www.zotero.org/>.

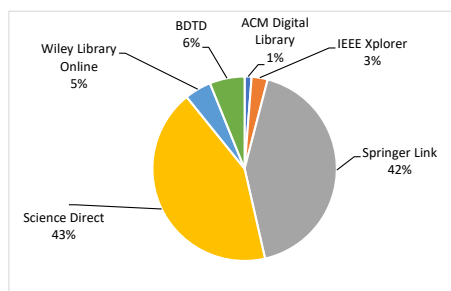


Figura 16: Resultado da *string* de busca nos repositórios

Após os critérios de inclusão e exclusão, contamos com um total de 103 trabalhos selecionados. Sendo, 35% destes da Base de Dados *Science Direct* e 23% do *Springer Link*. Os demais repositórios, somam 42%, como pode ser observado na Figura 17.

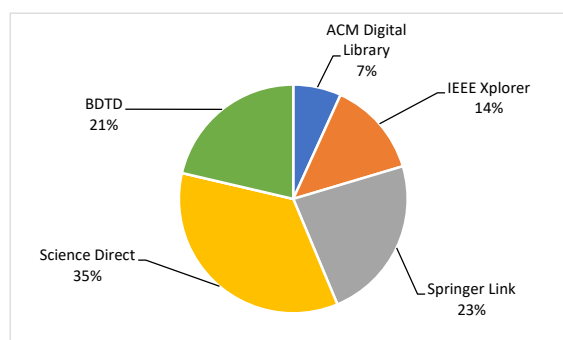


Figura 17: Resultado após critérios de inclusão e exclusão

Classificamos os estudos, de acordo com o tipo de produção científica e publicação, em: Artigos de Periódico, Artigos de Conferências, Teses de Doutorado e Dissertações de Mestrado. Seguindo essa classificação, contamos com 55 Artigos de Periódicos Científicos, 26 Artigos de Conferências, 16 Dissertações de Mestrado e 06 Teses de Doutorado, conforme Figura 18.

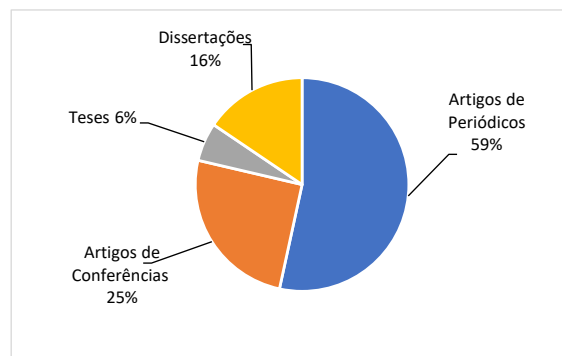


Figura 18: Tipos de produções científicas

Esses trabalhos foram publicados por 309 autores, vinculados a 138 instituições de ensino, distribuídas por 28 países. Os países com maior contribuição no período compre-

endido são a China e o Brasil, totalizam 44 publicações. Na Figura 19, podemos observar, uma nuvem de palavras contendo os países envolvidos.



Figura 19: Nuvem de palavras relacionadas com os países dos estudos selecionados

A seguir, é apresentado o número de instituições de ensino distribuídas pelos países, presentes nas publicações.

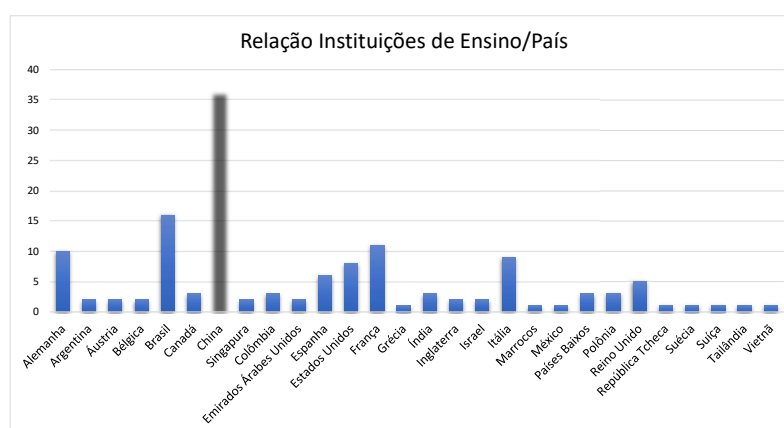


Figura 20: Número de instituições de ensino por país

Conforme o gráfico presente na Figura 20, a China é o país com o maior número de instituições que investem em pesquisas científicas relacionadas ao problemas de logística.

A Figura 21, apresenta o quantitativo de trabalhos publicados por fontes de publicações Internacionais. A *Transportation Research Part E: Logistics and Transportation Review*² conta com o com maior número de trabalhos.

²<https://www.journals.elsevier.com/transportation-research-part-e-logistics-and-transportation-review>

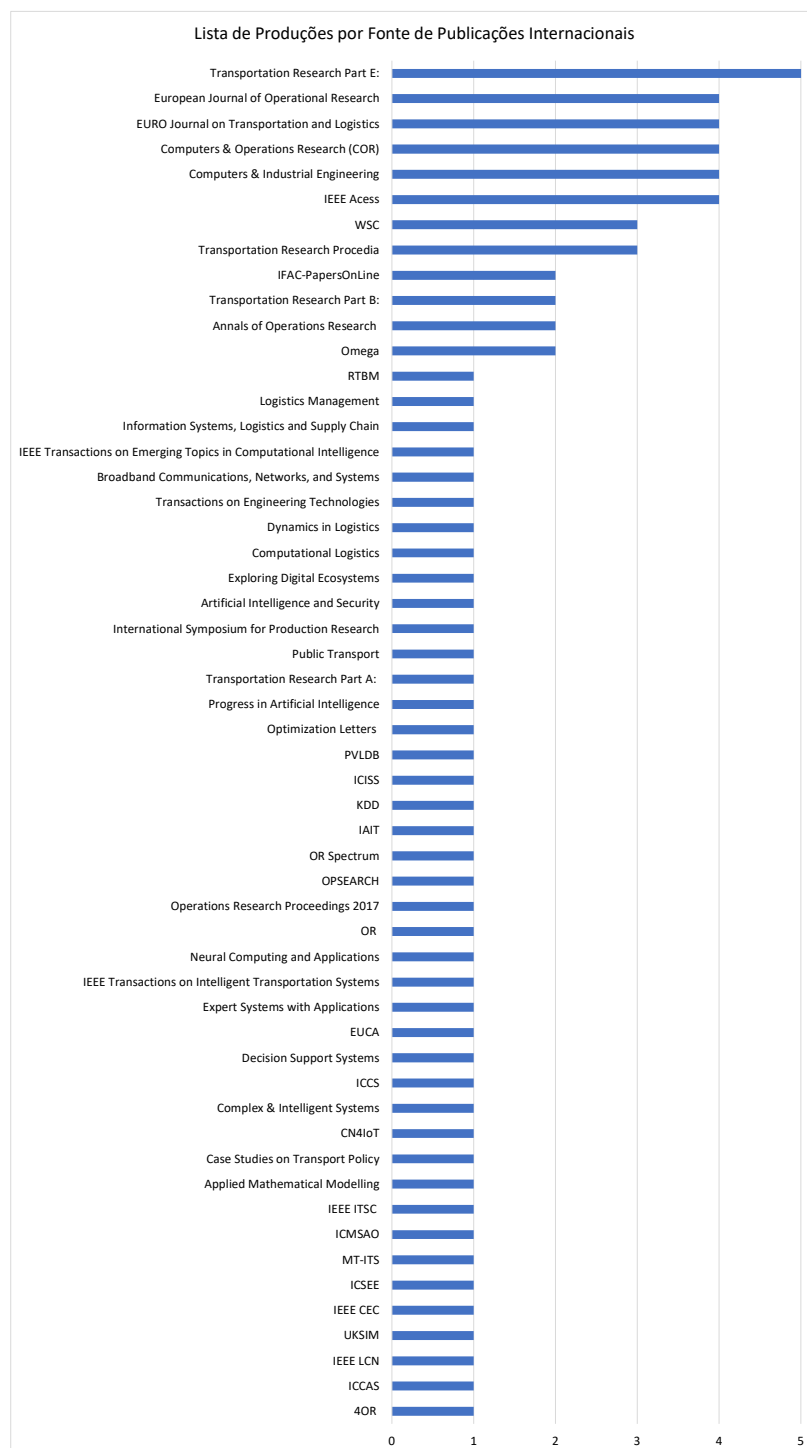


Figura 21: Lista de produções por fontes internacionais de publicações

Na Figura 22, destacamos as instituições de ensino brasileiras presentes nas publicações e o quantitativo de trabalhos de cada instituição. A UFSCar³ apresenta o maior número de publicações.

³UFSCar - Universidade Federal de São Carlos

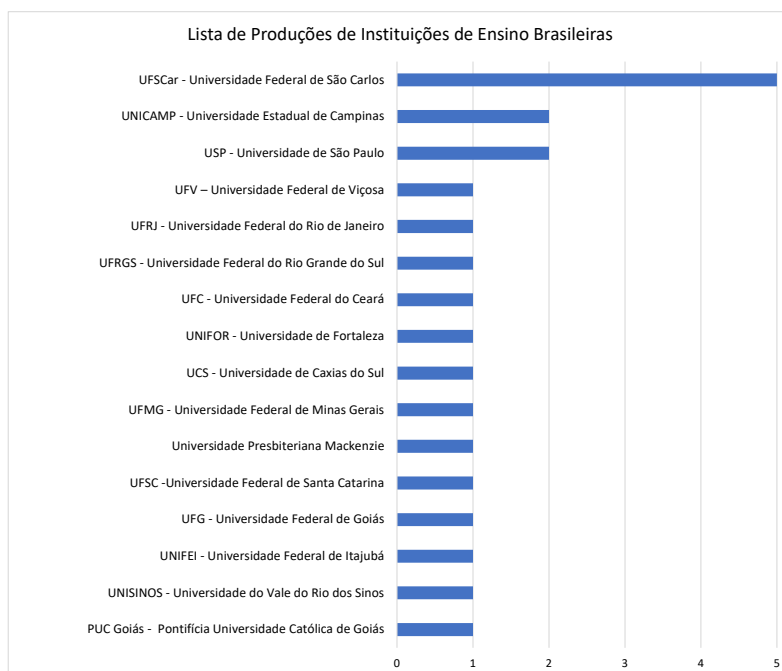


Figura 22: Lista de produções de instituições de ensino brasileiras

Os estudos selecionados foram publicados no período de 2016 a 2021. Sendo, os três últimos anos, os mais significativos em número de trabalhos publicados. A Figura 23, apresenta a relação de trabalhos publicados por ano.



Figura 23: Número de publicações por ano

As produções analisadas utilizaram 522 palavras-chave, destaca-se as seguintes: *Routing* com 61 ocorrências, *Problem* com 46, *Vehicle* com 35 ocorrências, *Delivery* com 33, *Logistics* com 33 e *Algorithm* com 19, presentes na nuvem de palavras na Figura 24.

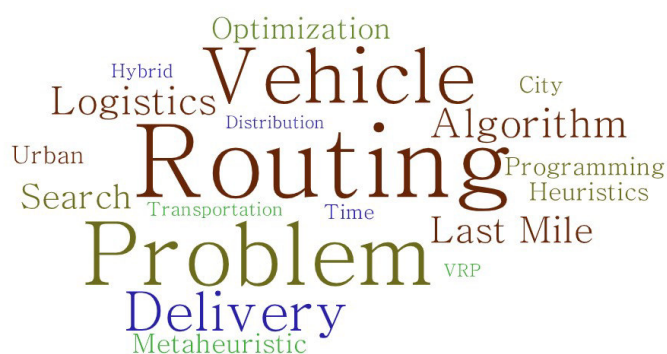


Figura 24: Nuvem de palavras relacionadas com as palavras-chave nos estudos selecionados

Todos os trabalhos selecionados, apresentavam resposta para a questão 1: **quais os métodos propostos na literatura no período de 2016 à 2021, para resolver o problema de roteamento de veículos?** Os métodos para soluções VRP apresentados nos estudos foram sintetizados em 5 categorias, sendo as seguintes: Métodos Exatos, Heurísticas, Metaheurísticas, Inteligência Artificial e Métodos Híbridos.

Dos 103 estudos selecionados, aproximadamente 67% das soluções propostas utilizam Metaheurísticas e Métodos Híbridos, 17% aplicam métodos exatos, apenas 6% valem-se de alguma técnica de inteligência artificial e 10% fazem uso de Heurísticas. Esses resultados podem ser observados na Figura 25.

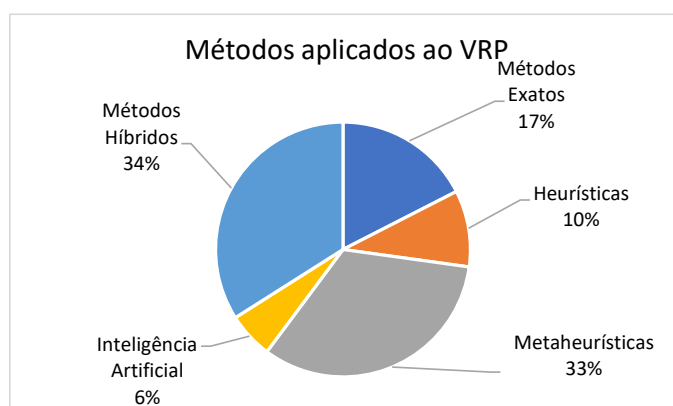


Figura 25: Categorização de soluções VRP

Apenas 51 estudos contemplam a questão 2: **quais os principais métodos para solução do problema de roteamento de veículos aplicados nas entregas de última milha?** Destes, os métodos mais utilizados são Metaheurísticas e Métodos Híbridos, presentes em 60% dos trabalhos, conforme observa-se no gráfico da Figura 26.

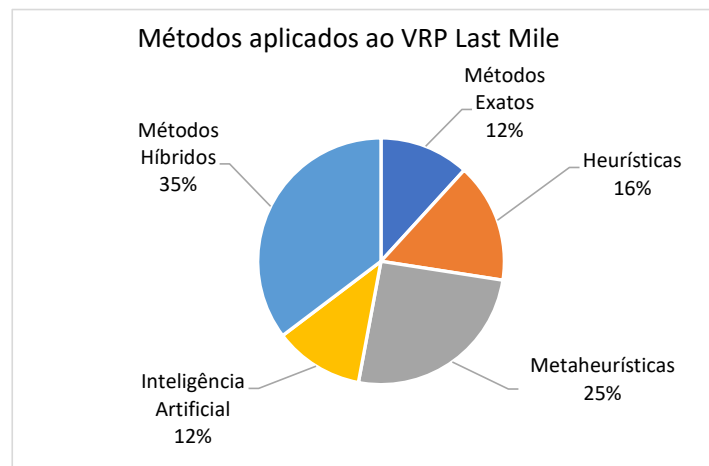


Figura 26: Categorização de soluções VRP *Last Mile*

Quanto à questão 3: **quais os principais limitadores dos métodos propostos?** A grande maioria dos trabalhos utilizados nessa pesquisa, enfrentaram problemas ao trabalhar com grandes bases de dados. Os testes foram realizados com poucas instâncias, obtendo nesses casos, bons índices de precisão e redução de custos. No entanto, os trabalhos em que foram utilizados bases de dados maiores, os índices dos resultados foram menos significativos.

Compreende-se por meio da análise e síntese dos resultados da pesquisa, que os métodos mais utilizados para solucionar problemas de roteamento de veículos clássicos, bem como, na última milha, consistem em metaheurísticas e hibridização.

Salienta-se que a totalidade da amostra com soluções baseadas em IA, apareceram somente em soluções do VRP *Last Mile*, não estavam presentes no problema clássico.

Pode-se concluir que a maioria das soluções existentes que alcançam melhores resultados e otimalidade utilizam instâncias e conjunto de dados pequenos para os testes.

4 PROCEDIMENTOS METODOLÓGICOS

A presente pesquisa científica teve seu início baseado em uma Revisão Sistemática da Literatura, utilizou abordagem metodológica Quali-Quantitativa para identificar o estado da arte do tema, reuniu conclusões e colaborações de outras pesquisas científicas.

Posteriormente, iniciamos o estudo da estrutura e organização do *Dataset Benchmark Loggi BUD*¹, que forneceu dados para os testes iniciais.

Com a finalidade de selecionar os algoritmos que seriam utilizados na aplicação, estudamos alguns dos principais algoritmos para cálculo de caminhos mínimos, sendo estudados e testados os algoritmos de *Dijkstra*, *Bellman-Ford* e *Floyd-Warshall*.

As ferramentas *OpenStreetMap* (OSM), *Open Source Routing Machine* (OSRM) e *OSMnx* foram utilizadas para mapeamento, processamento, modelagem, projeto, manipulação, visualização e análise de dados geoespaciais.

A aplicação foi desenvolvida por meio da linguagem de programação *Python*, com auxílio da linguagem de marcação de hipertexto (HTML) para desenvolvimento das páginas *web*, mecanismo CSS para adicionar estilos às páginas e o *Framework Flask*².

Os testes e a validação foram realizados por meio de diferentes experimentos, com diferentes bases de dados. Testes iniciais com o *Dataset Benchmark Loggi BUD* e os demais com conjuntos de dados XML extraídos do *OpenStreetMap*.

4.1 Classificação da pesquisa

O desenvolvimento do trabalho baseou-se em uma abordagem de pesquisa experimental.

A Pesquisa Experimental consiste em determinar um objeto de estudo, selecionar as variáveis que seriam capazes de influenciá-lo, definir as formas de controle e de observação dos efeitos que a variável produz no objeto [25].

Segundo Fonseca (2002), as duas modalidades de pesquisa experimental mais comuns são:

¹Benchmark Loggi BUD: <https://github.com/loggi/loggibud>

²Flask: Framework web escrito em Python utilizado para construir sites, aplicativos web, API e entre outros.

- Pesquisas Experimentais apenas com dois grupos homogêneos, denominados experimental e de controle, aplicado um estímulo ao grupo experimental, no final comparam-se os dois grupos para avaliar as alterações.
- Pesquisas Experimentais antes-depois com um único grupo, definido previamente em função de suas características e geralmente reduzido [23].

4.2 Proposta da aplicação

A proposta deste trabalho, consistiu na implementação de uma aplicação para o problema de roteamento de veículos na última milha, apoiada no algoritmo de *Dijkstra* e em dados reais obtidos no *Benchmark Loggi BUD (Benchmark Urban Deliveries)* e no *OpenStreetMap*, que possibilitou fornecer as rotas mínimas.

O conjunto de dados *Benchmark Loggi BUD* disponibilizado pela empresa logística *Loggi*[40], serviu como base de dados para os testes iniciais e o *OpenStreetMap* foi utilizado nos testes seguintes fornecendo um conjunto de dados em XML.

4.3 Modelo de arquitetura proposta

Para desenvolvimento do trabalho, estruturamos os processos em uma arquitetura formada por: um conjunto de dados *Benchmark Loggi Bud*, arquivos Json ou XML com dados das entregas utilizados inicialmente para a plotagem de mapas no *Openstreetmap*, um algoritmo de roteirização, por meio do qual são realizadas requisições HTTP ao servidor *OSRM*, cujo os dados processados, são estruturados em uma matriz de distâncias que serve como base de dados para o algoritmo de *Dijkstra* que forneceu os caminhos mínimos com base nos dados fornecidos. Essa arquitetura pode ser visualizada na Figura 27.

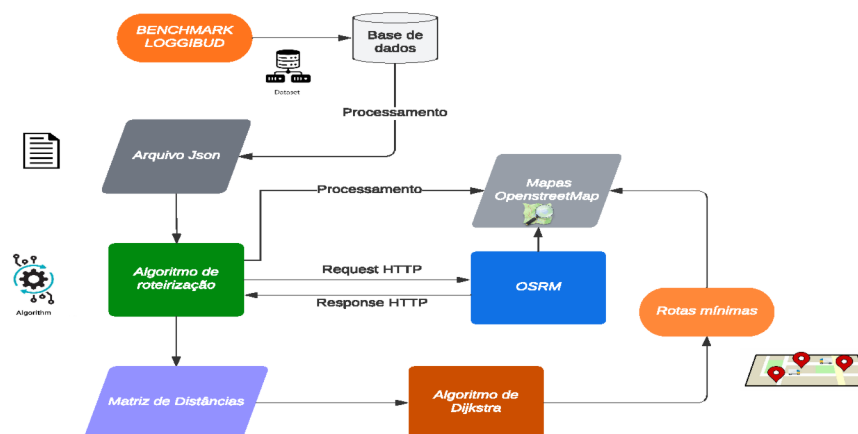


Figura 27: Arquitetura do modelo

4.4 Dataset Benchmark Loggi BUD

O *dataset* utilizado no trabalho trata-se do *Benchmark Loggi BUD*, que contém conjuntos de dados para problemas de grande escala, que simulam o desafio de entregas na última etapa da cadeia logística. O *Loggi BUD* em sua versão inicial fornece 90 instâncias de treino e 30 instâncias de avaliação para as cidades de Rio de Janeiro (RJ), Brasília (DF) e Belém (PA) [40].

Os dados desse *dataset* estão estruturados em dicionários de dados *JSON*³ organizados conforme as cidades citadas. Cada *JSON*, contém um nome, a região atendida, as coordenadas do ponto de origem (latitude e longitude), a capacidade dos veículos e os pontos de entrega, constituídos de um identificador e coordenadas do endereço. Na Figura 28, demonstramos a estrutura de um arquivo *JSON* presente no *dataset*.



Figura 28: Estrutura dos Dados *JSON*

Para utilização do *dataset*, foi necessário instalar um *docker*⁴ para executar o servidor *OSRM*. Após a instalação do *docker*, foi realizado o *download* dos arquivos pré-compilados disponibilizados pela *Loggi* e extraímos o conjunto de dados em um diretório *OSRM*. Na sequência, executamos um container de *back-end OSRM*. A Figura 29, demonstra algumas configurações do *container*.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
62863bc7c221	osrm/osrm-backend:v5.24.0	"osrm-routed --algor..."	2 weeks ago	Up 2 weeks	0.0.0.0:5000->5000/tcp, :::5000->5000/tcp	osrm

Figura 29: Docker *OSRM*

A manipulação e análise dos dados foi realizada por meio da Biblioteca *Pandas*⁵ e com o auxílio do *OpenStreetMaps*, convertimos as coordenadas em endereços completos e plotamos em mapas, buscando identificar as regiões atendidas.

³JSON: acrônimo de *JavaScript Object Notation* utilizado para troca de dados entre sistemas, possui formato leve e simples.

⁴Docker: plataforma *open source* que facilita a criação e administração de ambientes isolados. Ele possibilita o empacotamento de uma aplicação ou ambiente dentro de um container, se tornando portátil para qualquer outro host que contenha o *Docker* instalado.

⁵Pandas: Biblioteca de software criada para manipulação e análise de dados no Python.

Na Figura 30 apresentamos um dos mapas iniciais criados, contendo alguns pontos de entrega, resultado do processamento dos dados do *Benchmark*.

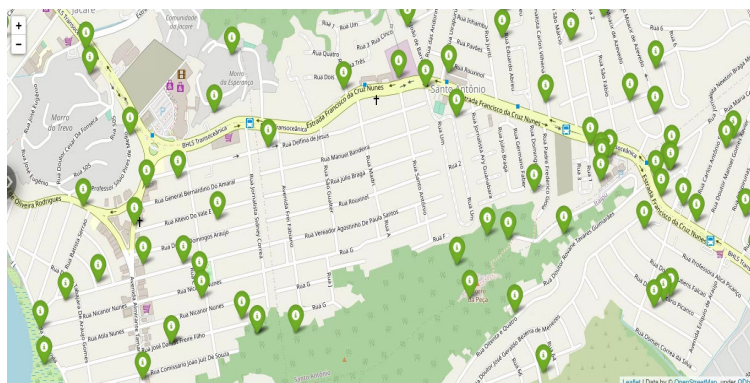


Figura 30: Mapa pontos de entrega

O próximo objetivo era obter as distâncias de um ponto ao outro para utilizarmos no grafo do Algoritmo de *Dijkstra*. As distâncias foram obtidas através de requisição HTTPS ao servidor *OSRM*, enviando como parâmetros as coordenadas de origem e destino e o modo de transporte (*car*, *foot*, *bike*, etc). Como resposta a requisição recebemos diversas informações, como a distância entre os pontos, o tempo de duração da viagem, o nome dos endereços e as coordenadas dos pontos. Alguns desses dados, resultado das requisições podem ser observados na Figura 31.

```
Terminal: Local +
(venv) (base) patricia@lab15-System-Product-Name:~/logibud-master$ python3.8 GrafoDeDistancias.py
[0.29, 1.8658, 1.1533, 1.1937, 1.309, 1.8402, 4.8842, 1.5366, 1.5631, 1.7832999999999999, 0.7152000000000001, 0.9552999999999999, 1.2682, 0.6144, 0.5809, 2.2276, 1.41
45, 2.3469, 0.6742, 1.4467999999999999, 1.7395999999999998, 0.2328, 4.8471, 2.1784, 1.5957000000000001, 1.5969, 1.458, 1.3417999999999999, 0.9117000000000001, 0.49819
99999999999, 1.3139, 0.8695, 2.8569, 1.8522, 2.8565, 1.1694, 2.8917, 1.9835, 1.3851, 2.5288000000000004, 0.8811, 0.7622000000000001, 3.6755999999999998, 2.4267, 1.20
42, 3.842, 1.3977, 0.9198999999999999, 1.2899, 1.2735999999999998, 1.6445999999999998, 1.9287999999999998, 2.3969, 3.9562, 0.573, 0.9315, 0.6512, 2.8185, 2.1129000000
000002, 0.6166, 1.228, 0.7442000000000001, 1.2282, 2.7423, 4.8142, 4.882, 2.2424, 1.1135, 1.4338999999999998, 1.7962, 0.7774, 1.8487, 1.2982, 0.7663, 2.6955, 2.1826,
0.4674, 1.887, 1.4029, 1.7695, 1.5957000000000001, 3.5616999999999996, 0.4153, 1.7635999999999998, 1.9383, 2.3507, 3.842, 1.5405, 1.2492999999999999, 2.8998, 0.384, 0
.337, 1.272, 2.4445, 1.6892, 1.5965, 2.6343, 1.875, 1.981, 0.5473, 1.5683, 2.9781, 2.9249, 0.9105, 1.4897, 2.8813, 0.9416, 2.8163, 1.3722999999999999, 1.458, 2.660199
9999999997, 1.7816, 1.5595, 1.4323, 1.6262999999999999, 1.6396, 0.3521, 3.2506999999999997, 1.8962999999999999, 3.5744000000000002, 1.7372, 1.1832, 1.3137, 2.41930000
00000002, 0.25680000000000003, 0.757, 2.1433, 1.9549, 1.7552999999999999, 1.3839, 1.981, 2.4153000000000002, 1.3865, 31.8996, 31.8548, 50.2483, 29.6638, 32.526, 29.68
86, 31.4185, 38.8189, 29.0455, 32.9085, 33.232699999999994, 29.2347, 29.485400000000002, 34.7615, 34.805800000000005, 41.5824, 29.9728, 30.188, 39.1595, 30.4285, 33.0
547, 47.3281, 50.2683, 33.1698, 29.6312, 33.4601, 31.5456, 32.533, 31.3953, 34.6337, 50.5216, 34.3651, 32.8513, 31.923299999999998, 34.6874, 32.2899, 34.5547, 29.394,
29.2347, 33.035599999999995, 29.345599999999997, 30.726800000000004, 32.9966, 33.3611, 33.414300000000004, 29.246, 33.8116, 41.52, 32.6212, 30.7839, 33.8544, 32.4338
```

Figura 31: Resultado Requisição HTTP para distâncias

4.5 Validação do modelo

A última etapa do trabalho, consistiu na validação do modelo proposto baseada na execução de três testes principais.

No primeiro teste, a aplicação recebeu instâncias de dados reais do *Loggi BUD* para processamento dos caminhos mínimos e o Algoritmo de *Dijkstra* calculou e retornou a menor distância compreendida entre a origem e os demais pontos da amostra.

No segundo teste, utilizamos dados dinâmicos por meio de conjunto de dados em XML gerados no *OpenStreetMap*, como resultado obtivemos as rotas com caminhos mínimos geradas pelos Algoritmos de *Dijkstra e Bellman-Ford*.

Como último teste, calculamos rotas mínimas para transporte escolar do Instituto Federal de Crateús no Ceará. A escolha desse conjunto de dados se justifica pelo fato de estar disponível na internet para acesso público e por se tratar de dados de rotas reais, além disso, os demais conjuntos de dados encontrados forneciam somente o nome do proprietário da localidade, não sendo possível encontrar a real localização do ponto de parada do transporte, por exemplo, ponto 1 residência do Joãozinho.

A distância da rota gerada foi comparada com a distância obtida no *Google Maps*, uma das ferramentas de pesquisa, visualização de mapas e roteamento mais conhecidas e consolidadas, sendo possível constatar que a aplicação retornou o caminho mínimo, validando a solução.

5 ANÁLISE E DISCUSSÃO DOS DADOS

O presente capítulo está dividido em três seções que apresentam os resultados e a análise dos testes realizados no decorrer da pesquisa. A primeira seção dispõe sobre os testes iniciais da aplicação realizados com dados do *Benchmark Loggi BUD*. Na segunda seção, são discutidos os testes realizados com dados dinâmicos, mais especificamente com arquivos de roteirização gerados no *OpenStreetMap* e na terceira seção temos os dados referentes aos testes realizados com um conjunto de dados de transporte escolar.

Todos os testes com dados dinâmicos e transporte escolar foram executados em um *notebook* com as seguintes configurações de *hardware*: 16 GB de memória RAM e processador AMD *Ryzen 5 5600U* com 6 núcleos de CPU, 12 *threads*, *clock* básico de 2.3 GHz e *clock* de *Max Boost* até 4.2 GHz.

5.1 Testes com dados do *Benchmark Loggi BUD*

Essa seção apresenta os resultados dos testes realizados com o algoritmo de *Dijkstra* e os dados do *Benchmark Loggi BUD*.

Inicialmente, lemos um arquivo de dados Json referente a região classificada como região zero do Rio de Janeiro, no qual temos 3728 pontos de entrega. Os dados desse arquivo foram enviados para um mapa em HTML baseado em *OpenStreetMap* que é responsável por salvar e apresentar todos os pontos de entrega da região em questão. Os referidos dados são apresentados nas Figuras 32 e 33 a seguir.

```

Terminal: Local
95823), size=2), Delivery(id='7265c3b32545fe4b737c5d96f9fe44d', point=Point(Lng=-43.040102984608936, lat=-22.82417593912677), size=9), Delivery(id='e7f40ae3a5addf827
5f47c04f8c12c37', point=Point(Lng=-43.83267288819325, lat=-22.813844438495677), size=7), Delivery(id='98a92ead8c5b39877159fb3c2217ccbb', point=Point(Lng=-43.844388979
850015, lat=-22.8075235064866), size=1), Delivery(id='d598b70ca01f96ec234a68f52c69afc8', point=Point(Lng=-43.02565504613167, lat=-22.81752990750083), size=7), DeLive
ry(id='ff00f52de1d6ef03acf04915e218c031', point=Point(Lng=-43.040263954015886, lat=-22.815338402638133), size=3), Delivery(id='2b08a910922202ede1ef0cc2359f88d', poi
nt=Point(Lng=-43.84228583883524, lat=-22.81378795978488), size=6), Delivery(id='da7dfce51885fe97ab786f1dfaf2ddb', point=Point(Lng=-43.83879287352534, lat=-22.82383249
8841616), size=4), Delivery(id='199b05077f7395c44ae5dc501ec81ee4', point=Point(Lng=-43.0241239557979, lat=-22.814030476466947), size=5), Delivery(id='0d1acbf72a526bc
409ba8dd840f0b05', point=Point(Lng=-43.836167130781564, lat=-22.820403798080658), size=10), Delivery(id='e7095990ce6658dd8bd9d57981fec15a5', point=Point(Lng=-43.837634
57395887, lat=-22.818742811815483), size=1), Delivery(id='594943fec2f10e57af02a2b985d54c6a', point=Point(Lng=-43.8389210952452, lat=-22.815576138919415), size=1), Del
ivery(id='c1d352e7033168d5676acba820bac7c5', point=Point(Lng=-43.032960459399696, lat=-22.811644096639647), size=6), Delivery(id='c2077ee9d7e709f7c496e7eeaac227f', p
oint=Point(Lng=-43.83848804885548, lat=-22.813931672433688), size=10), Delivery(id='d9cf7c04e919a75c982e0808eb2b2543', point=Point(Lng=-43.82653658123728, lat=-22.886
51889873486), size=8), Delivery(id='67a81b823297832cc47adb8a8802c42c', point=Point(Lng=-43.82617119952517, lat=-22.81738859199573), size=7), Delivery(id='9a21a59a599
a4719f09200508a8e0ba07', point=Point(Lng=-43.04084492888289, lat=-22.807882656513743), size=6), Delivery(id='23648c94a029ecf717acdac0ac81915e', point=Point(Lng=-43.040
66132851633, lat=-22.883687892699272), size=9), Delivery(id='682f2d1f64f55dad54386d1d03dd2af9', point=Point(Lng=-43.83201355856611, lat=-22.88735678925421), size=10),
Delivery(id='94b4e8fc584e5168956688a4f94d93e3', point=Point(Lng=-43.8283077229614, lat=-22.818412888289424), size=5), Delivery(id='aa748837798caabc2abec47908818e2',
point=Point(Lng=-43.8315911497832, lat=-22.81708436429074), size=9)])

Total de pontos de entrega: 3728
Gerando mapa dos pontos de entrega e salvando no arquivo MapaEntregas.html
Mapa pontos com os 200 pontos de amostra, salvo no arquivo MapaEntregasAmostra.html

```

Figura 32: Dados obtidos com o processamento dos dados da Região 0 do RJ

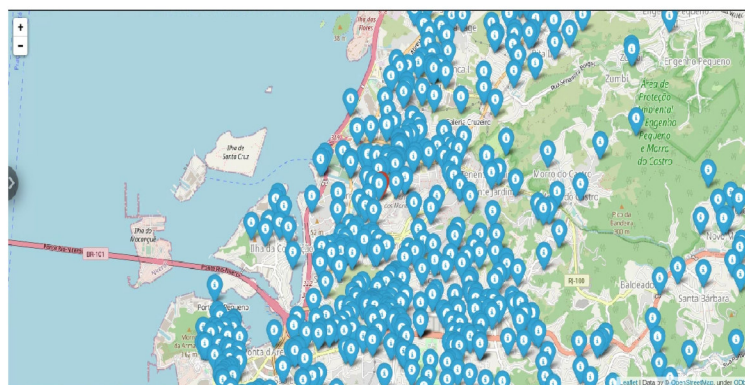


Figura 33: Mapa dos pontos de entrega classificados no *dataset* como Região 0 do RJ

Após a exploração dos dados, selecionamos uma amostra com 200 pontos de entrega, que passaram pelo processamento anterior e foram enviados e salvos no mapa denominado MapaEntregasAmostra.html. O mapa mencionado é demonstrado na Figura 34.

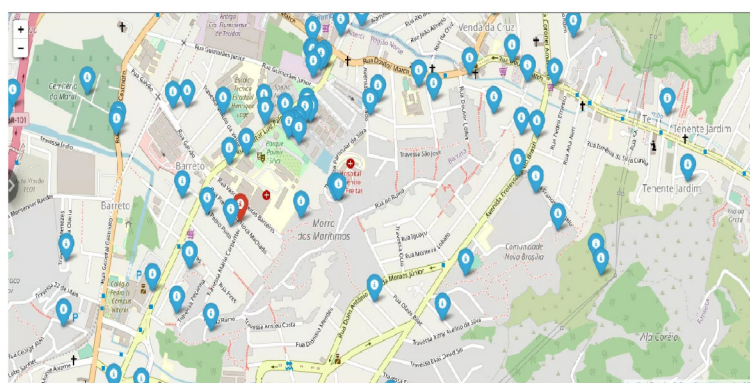


Figura 34: Mapa pontos de entrega da amostra selecionada

Os balões azuis representam os locais de entrega, identificados pelas coordenadas geográficas, latitude e longitude. O balão na cor vermelha, representa o centro de distribuição, origem do trajeto.

Com os dados da amostra selecionados, por meio do algoritmo desenvolvido, realizamos requisições HTTP ao servidor OSRM, enviamos o ponto de origem e os pontos de entrega e requisitamos a distância entre os pontos. Com o resultado da solicitação criamos uma matriz de distâncias.

Os dados da matriz foram utilizados para o desenvolvimento do grafo do algoritmo de *Dijkstra*. Sendo cada ponto de entrega atribuído ao vértice do grafo e as distâncias obtidas atribuídas às arestas, convertemos as distâncias para quilômetros, visto que, estavam classificadas em metros.

Com o grafo criado, e o ponto de origem atribuído, calculamos por meio do algoritmo de *Dijkstra* a menor distância compreendida entre a origem e os demais pontos da amostra. Conforme pode ser observado na Figura 35.

```

Terminal: Local x +
Distancia da origem ate o vertice
0 total 0.0 Km
1 total 30.738100000000003 Km
2 total 4.8661 Km
3 total 0.5814 Km
4 total 5.2696000000000005 Km
5 total 4.0252 Km
6 total 3.6307 Km
7 total 2.161 Km
8 total 3.5783 Km
9 total 3.9483 Km
10 total 3.8977 Km
11 total 2.1625 Km
12 total 5.3366000000000001 Km
13 total 2.6809000000000003 Km
14 total 2.6029 Km
15 total 3.1176999999999997 Km
16 total 5.7091 Km
17 total 0.4804 Km
  
```

Figura 35: Distâncias obtidas com o Algoritmo de *Dijkstra*

5.2 Testes com dados dinâmicos

Na sequência dos testes iniciais realizados com dados do *dataset Loggi BUD*, iniciamos um segundo teste, desta vez, com dados de mapeamento oriundos de arquivos XML.osm gerados na plataforma *OpenStreetMap*, disponível em: <https://www.openstreetmap.org/#map=14/-30.0238/-51.1466>, conforme interface apresentada a seguir.

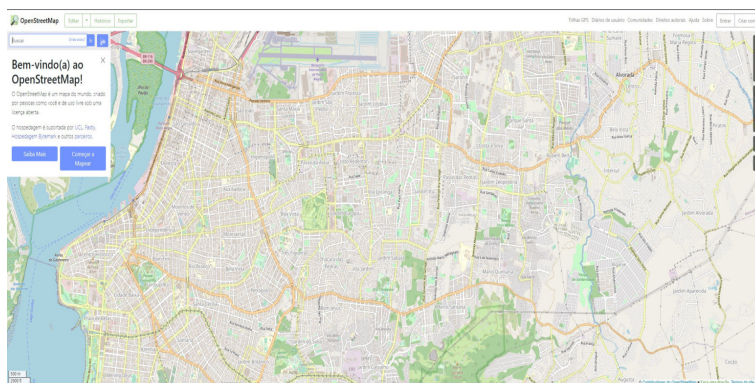


Figura 36: Tela inicial do *OpenStreetMap*

Na interface apresentada basta inserir a cidade que deseja mapear, clicar em “Ir” e “Exportar”, selecionar a área a ser mapeada e clicar novamente em exportar, com isso será gerado um arquivo com estrutura XML denominado por padrão *map.osm*.

De posse do arquivo gerado, fizemos a manipulação dos dados, definimos um ponto de origem e diversos destinos, locais que obrigatoriamente devem compor a rota. Com o arquivo de dados pronto para ser utilizado, executamos a aplicação que possui a interface inicial conforme Figura 37 apresentada na sequência.



Figura 37: Tela inicial da aplicação

Como podemos observar acima, é solicitado para o usuário a seleção do estado e cidade a serem mapeados, bem como, o *upload* do arquivo *osm* com os nós definidos para traçar a menor rota, após preencher todos os campos solicitados, basta clicar em Gerar rota.

Os estados e as respectivas cidades são fornecidos utilizando a API Localidades UFs do IBGE¹, responsável por retornar o conjunto de Unidades da Federação do Brasil.

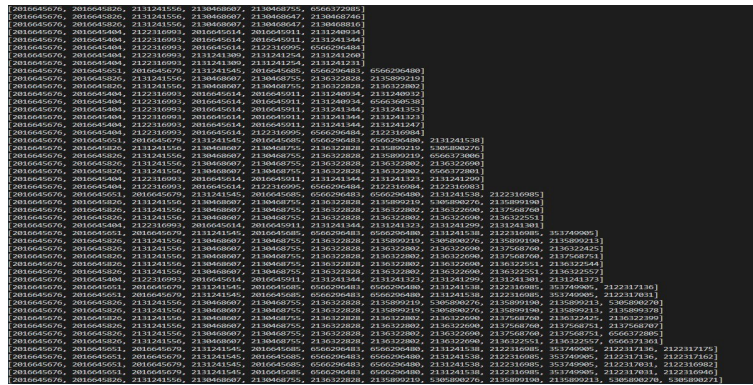
Ao solicitar a geração de rotas, a aplicação realiza o processamento da requisição. Inicialmente, identifica a cidade e estado informados e realiza a leitura do arquivo *osm*,

¹ API Localidades IBGE: <https://servicodados.ibge.gov.br/api/docs/localidades>

captura todas as informações referentes aos nós da rota, como id do *node*, a latitude e longitude.

Os dados capturados do arquivo *osm* são adicionados a um grafo que possui todos os nós e arestas da área mapeada e é configurado com parâmetro *network_type* definido como do tipo *drive*, ou seja, o tipo de malha de rua é definido para dirigir. Além dessa opção, a biblioteca do OSMnx, possibilita configurar redes de rua do tipo *bike*², *drive_service*³, *all*⁴, *all_private*⁵ ou *walk*⁶.

Posteriormente, com o auxílio da biblioteca *folium*⁷, plotamos o grafo no mapa *OpenStreetMap*, utilizado no algoritmo de *Dijkstra* e no algoritmo de *Bellman-Ford* para cálculo do menor caminho entre a origem e todos os demais pontos. A Figura 38 apresenta parte do resultado da execução do algoritmo de *Dijkstra*, comparando o nó definido como origem com os demais nós compreendidos na rota, apresenta a ordem de passagem pelos nós conforme vai calculando o menor caminho.



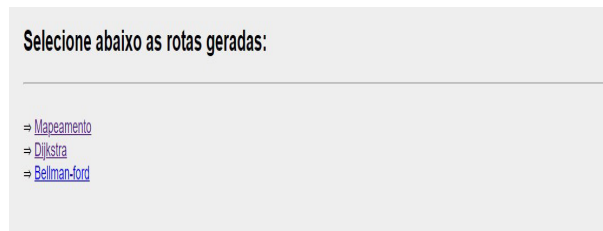


Figura 39: Página web contendo menu de opções

Ao selecionar a opção Mapeamento, temos como resultado o exemplo presente nas Figuras 40 e 41, que apresentam o mapeamento da cidade de Mata no Rio Grande do Sul.

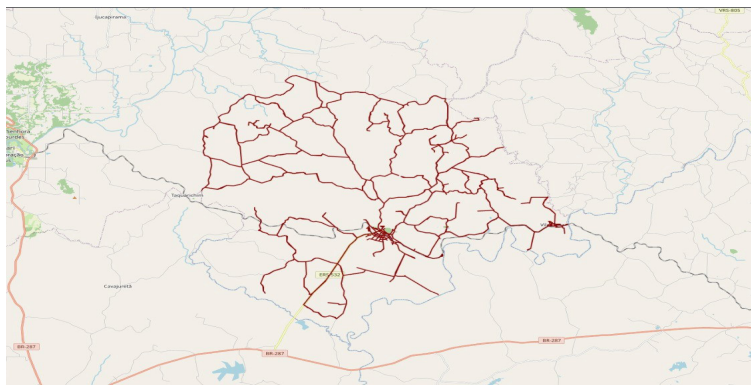


Figura 40: Mapeamento da cidade de Mata



Figura 41: Grafo gerado da cidade de Mata

Em relação às rotas geradas, nesse exemplo da cidade de Mata, os dois algoritmos retornaram rotas iguais com um total de 79,97 Km percorridos em cada uma. No entanto, o algoritmo de *Dijkstra* teve menor tempo de execução com aproximadamente 1,3 segundos de diferença, conforme pode ser observado na Figura 42.

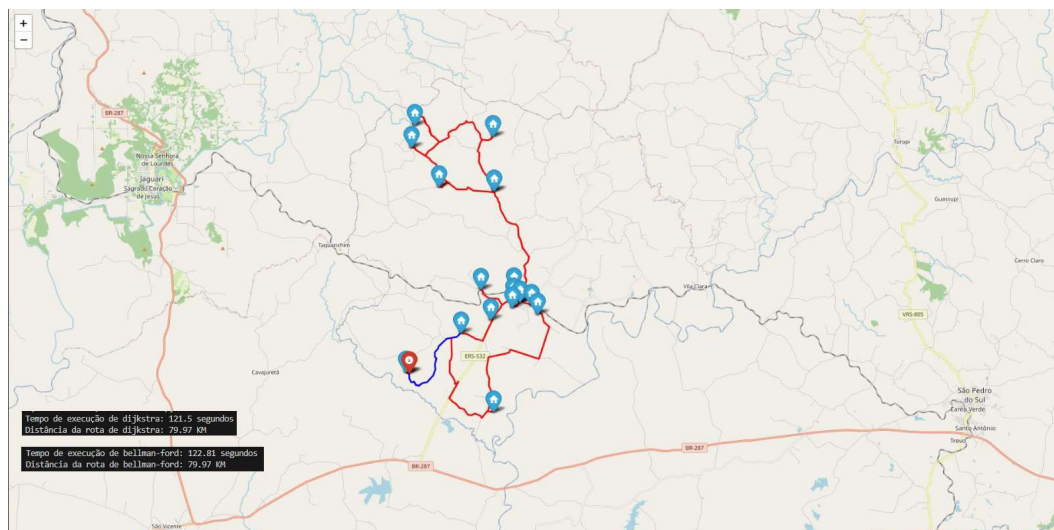


Figura 42: Rota gerada pelo algoritmo de *Dijkstra e Bellman-Ford*

5.3 Teste com dados de transporte escolar

Submetemos a aplicação a testes com um conjunto de dados de transporte escolar do Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE, localizado na cidade Crateús no Ceará.

Os dados foram obtidos de uma base de dados na página do Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE⁸, no entanto, estavam estruturados em uma tabela contendo somente o número do ponto e os nomes dos locais de parada do veículo. Sendo necessário estruturar esses dados em um conjunto de dados XML com o auxílio do *OpenStreetMap*, gerando um arquivo com o identificador dos *nodes*, latitude, longitude e o ponto de partida do transporte.

A Figura 43 e a Figura 44 apresentam a cidade de Crateús mapeada, representando o resultado do grafo gerado.

⁸Página do Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE disponível em: <https://ifce.edu.br/>

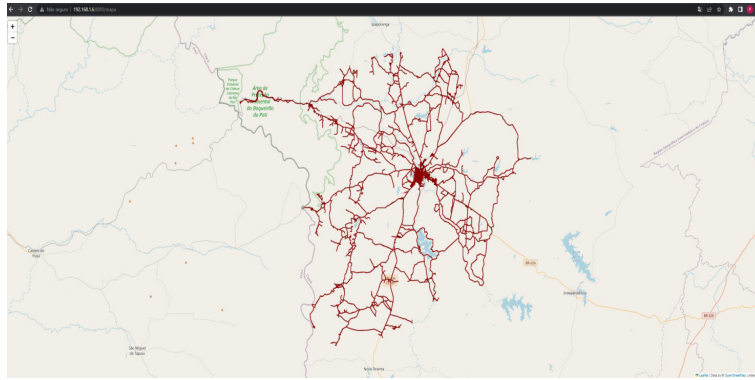


Figura 43: Mapeamento da cidade de Crateús

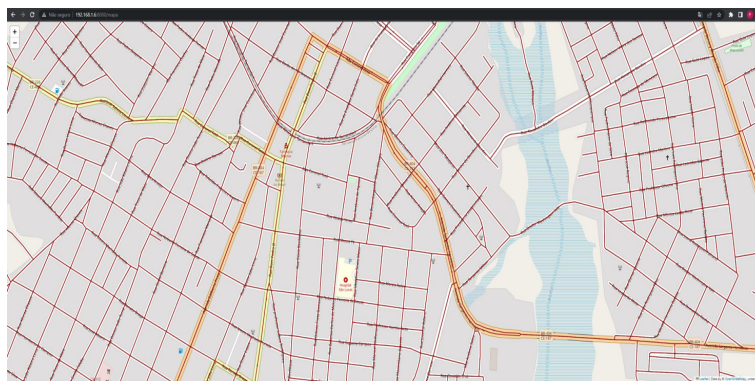


Figura 44: Grafo gerado da cidade de Crateús

O teste realizado utilizou somente dez pontos de parada, para que pudessemos comparar com a rota gerada no *Google Maps*, que na versão gratuita está limitado a dez pontos.

As distâncias percorridas nas rotas obtidas com *Dijkstra* e *Bellman-ford* na aplicação são bem semelhantes com o resultado obtido no *Maps*. No entanto, a rota gerada na aplicação mostrou uma economia de aproximadamente 0,2 Km percorridos.

As rotas geradas na aplicação pelos dois algoritmos foram iguais, cada uma retornou 11,55 Km percorridos. Em relação ao tempo de execução, houve uma diferença de aproximadamente dois segundos entre os dois algoritmos. Os resultados podem ser observados na Figura 45.

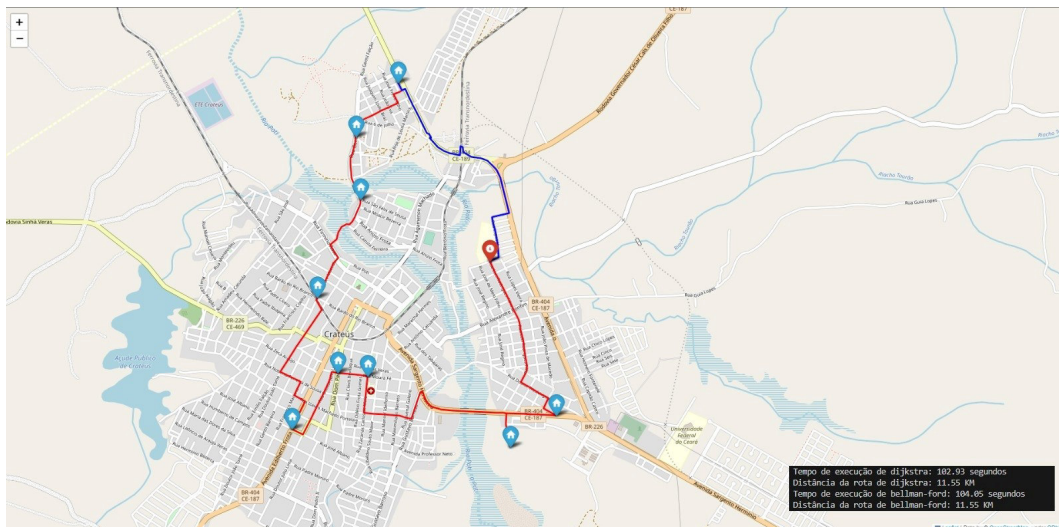


Figura 45: Rota gerada com *Dijkstra e Bellman-Ford*

Por sua vez, em relação a distância percorrida, o *Maps* retornou uma rota diferente com um total de 11,8 Km de distância percorrida. A rota gerada pode ser visualizada na Figura 46.

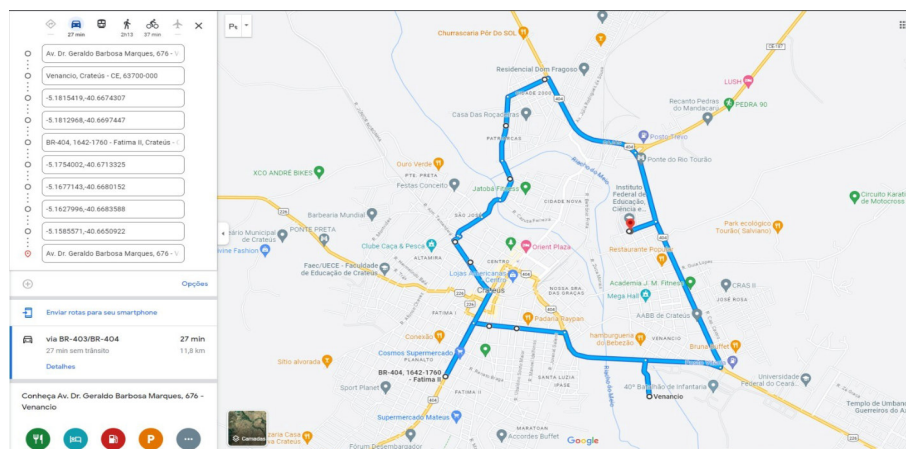


Figura 46: Rota gerada com o *Maps*

Na sequência, testamos a aplicação com uma rota menor compreendendo apenas quatro nós. Os resultados gerados por *Dijkstra e Bellman-Ford* podem ser observados na Figura 47.



Figura 47: Rota gerada com *Dijkstra* e *Bellman-Ford*

As rotas geradas com os algoritmos tradicionais são iguais em relação a distância e caminhos percorridos, ambas indicam 6,7 Km de rota.

Realizamos o mesmo teste no *Maps* que totalizou 8,6 Km percorridos, conforme pode se observar na Figura 48. Esse resultado demonstra que as rotas geradas na aplicação reduzem as distâncias percorridas em aproximadamente 1,9 Km.

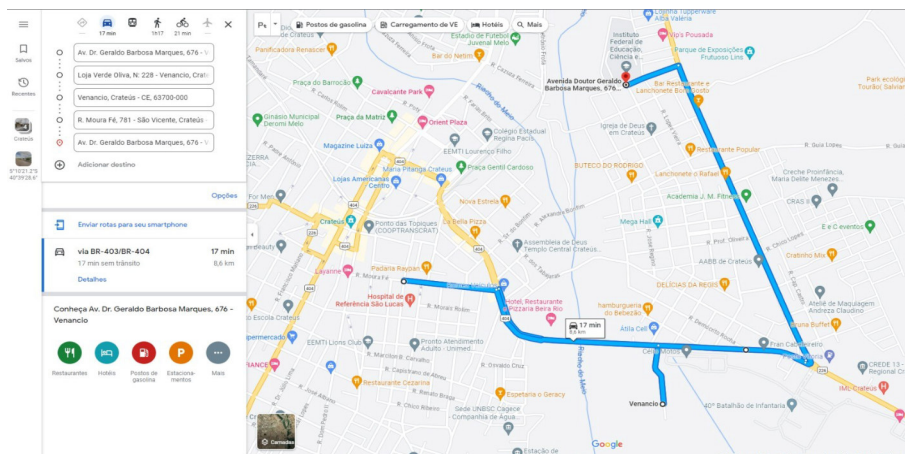


Figura 48: Rota gerada com o *Maps*

Ao observar os dados dos testes realizados foi possível perceber que o algoritmo de *Dijkstra* apresenta um melhor desempenho em relação ao tempo de processamento do *Bellman-Ford*, com diferenças de tempo de execução entre 2 e 4 segundos em média.

Além disso, ao comparar as distâncias das rotas calculadas pela aplicação com as distâncias apresentadas no *Maps* constatou-se que se assemelham em relação as distâncias percorridas.

6 CONSIDERAÇÕES FINAIS

Este trabalho teve como principal objetivo o desenvolvimento de uma aplicação para resolver o problema de roteamento de veículos na última milha, baseado em algoritmos tradicionais e na utilização de conjuntos de dados reais disponíveis no *Benchmark Loggi BUD* e no *OpenStreetMap*. Alinhado a esse objetivo, foi proposto uma nova abordagem para cálculo e geração de rotas com distâncias mínimas por meio de arquivos de dados *osm*.

Durante o desenvolvimento da pesquisa, foram realizados testes e comparativos de desempenho com os algoritmos de *Dijkstra* e *Bellman-Ford*. Os dois algoritmos retornaram as mesmas rotas e as mesmas distâncias percorridas, no entanto, o algoritmo de *Dijkstra* teve melhor desempenho no processamento das requisições, apresentando em todos os testes realizados tempos de execução menores que o algoritmo de *Bellman-Ford*.

Como forma de validar os resultados apresentados pela aplicação, fizemos alguns testes com a ferramenta *Maps* da empresa *Google*. Ao final de todos os testes, foi possível constatar que a aplicação propôs rotas consistentes e com distâncias semelhantes às distâncias geradas no *Google Maps*, podendo auxiliar no processo de tomada de decisão de rotas para redução de custos operacionais com transporte escolar.

Como trabalhos futuros, espera-se focar no desenvolvimento da aplicação *mobile*, com interface amigável e intuitiva, e que além disso, seja capaz de considerar outros fatores, tais como condições da rodovia, tempo de duração da viagem, trânsito, segurança e entre outros. Além disso, otimizar a aplicação para que possa trabalhar com um volume maior de dados com menor tempo de processamento.

REFERÊNCIAS

- [1] Altabeeb, A. M. e. a. (2021). **Solving capacitated vehicle routing problem using cooperative firefly algorithm.** *Applied Soft Computing*, 108.
- [2] Bai, R., Chen, X., et al. (2021). **Analytics and Machine Learning in Vehicle Routing Research.** *Cornell University, arXiv:2102.10012v1 [cs.LG] 19 Feb 2021.*
- [3] Baldacci, R., M. A. and Roberti, R. (2012). **Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints.** *European Journal of Operational Research*, 218:1–6.
- [4] Barros, E. A. R., Pamboukian, S. V. D., et al. (2007). **Algoritmo de Dijkstra: Apoio didático e multidisciplinar na implementação, simulação e utilização computacional.** *ICECE - International Conference on Engineering and Computer Education, SP, Brasil.*
- [5] Barroso, M. M. d. A. (2014). **Aplicação de grafos em um problema de rede.** *Abakós, Belo Horizonte*, 2(2):48–78.
- [6] Benevides, P. F. (2011). **Aplicação de Heurísticas e Metaheurísticas para o Problema do Caixeiro Viajante em um problema real de Roteirização de Veículos.** *Universidade Federal do Paraná, UFPR.*
- [7] Boeing, G. (2017). **OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks.** *Computers, Environment and Urban Systems* 65, pages 126–139.
- [8] Bogue, E. T. (2020). **Algoritmos exatos e heurísticas para variações de problemas de roteamento, empacotamento e corte guilhotinado.** *Universidade Federal de Minas Gerais - UFMG.*
- [9] Borowski, M., Gora, P., et al. (2020). **New Hybrid Quantum Annealing Algorithms for Solving Vehicle Routing Problem.** *ICCS International Conference on Computational Science*, 12142(1).

- [10] Boysen, N., Fedtke, S., et al. (2020). **Last-mile delivery concepts: a survey from an operational research perspective.** *OR Spectrum*, 43:1–58.
- [11] Brito, A. P. d. (2014). **Grafos, a Fórmula de Euler e os Poliedros Regulares.** *Universidade Federal Rural de Pernambuco - Mestrado Profissional em Matemática em Rede Nacional PROFMAT.*
- [12] Cai, Y. e. a. (2022). **A hybrid evolutionary multitask algorithm for the multiobjective vehicle routing problem with time windows.** *Information Sciences*, 612:168–187.
- [13] Cai, Z., Zhang, Z., et al. (2018). **Solving the last mile delivery problem using iterated local search approach.** *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC).*
- [14] Carić, T. and Gold, H. (2008). **Vehicle Routing Problem.** IntechOpen.
- [15] Casademunt, M. S. M. (2017). **Uma nova abordagem de utilização de grafos a partir de camadas de iso-eficiência utilizando o DEA (Data Envelopment Analysis).** *Universidade Estadual do Ceará, Centro de Ciências e Tecnologia.*
- [16] Chu, H., Zhang, W., et al. (2021). **Data-driven optimization for last-mile delivery.** *Complex & Intelligent Systems.*
- [17] Cormen, T. H. e. a. (2012). **Algoritmos Teoria e Prática.** 3. ed. Rio de Janeiro: Elsevier, 2012.
- [18] Daudt, C. S. and de Miranda, C. L. P. (2010). **Programação Dinâmica: Algoritmo de Bellman-Ford.** *Universidade Federal do Rio Grande do Sul.*
- [19] Denyer, D. and Tranfield, D. (2009). **Producing a systematic review.** *Buchanan, D. (ed.). The Sage Handbook of Organizational Research Methods. London: Sage, pages 671–689.*
- [20] Falci, M. L. F. and Jabour, F. C. (2015). **Um algoritmo de otimização de compras com suporte a rotas.** *Revista Seminários de Trabalho de Conclusão de Curso do Bacharelado em Sistemas da Informação, Instituto Federal do Sudeste de Minas Gerais, MG, Brasil.*
- [21] Floyd, R. W. (2010). **Algoritmo 97: Caminho mais curto.** *Communications of the ACM*, 5(6).
- [22] FNDE (2022). **Painel de Investimentos em Educação Básica.** *Disponível em: <https://www.gov.br/inep/pt-br/assuntos/noticias/institucional/mec-lancou-o-painel-de-investimentos-em-educacao-basica>. Acesso em: 30/04/2023.*

- [23] Fonseca, J. J. S. (2002). **Metodologia da Pesquisa Científica**. Fortaleza: UEC.
- [24] Gevaers, R., Voorde, E. V., et al. (2009). **Characteristics of Innovations in last mile Logistics - using best practices, case studies and making the link with green and sustainable logistics**. Association for European Transport and contributors 2009.
- [25] Gil, A. C. (2007). **Como elaborar projetos de pesquisa**. 4. ed. São Paulo:Atlas.
- [26] Glover, F. and Kochenberger, G. (2003). **Handbook of metaheuristics**. 1. ed. Nova Iorque: Kluwer Academic Publishers.
- [27] Goldbarg, M. and Goldbarg, E. (2017). **Grafos: Conceitos, Algoritmos e Aplicações**. Elsevier.
- [28] Gomes, D. (2010). **Inteligência Artificial: conceitos e aplicações**. Revista Olhar Científico – Faculdades Associadas de Ariquemes, 1(2).
- [29] Gupta, A. and Saini, S. (2017). **An Enhanced Ant Colony Optimization Algorithm for Vehicle Routing Problem with Time Windows**. 2017 Ninth International Conference on Advanced Computing (ICoAC).
- [30] Góes, A. R. T. (2005). **Otimização na distribuição de carga horária de professores - Método exato, método heurístico, método misto e interface**. Universidade Federal do Paraná, UFPR, Curitiba, PR.
- [31] Haykin, S. (2001). **Redes Neurais: Princípios e prática**. Bookman.
- [32] Hernandez, F., Berton, L., et al. (2009). **O Problema de Caminho Mínimo com incertezas e restrições de tempo**. Revista Pesquisa Operacional, 29:471–488.
- [33] Hougardy, S. (2010). **The Floyd–Warshall algorithm on graphs with negative cycles**. Information Processing Letters, 110(8-9):279–281.
- [34] Ilhe, G. (2018). **Last-Mile: o que é e o quanto isso impacta em sua operação?** E-commerce Brasil.
- [35] Jiang, H. e. a. (2022). **An evolutionary algorithm for solving Capacitated Vehicle Routing Problems by using local information**. Applied Soft Computing, 117.
- [36] Jiang, L., Zang, X., et al. (2021). **A covering traveling salesman problem with profit in the last mile delivery**. Optimization Letters.
- [37] Kim, G., Ong, Y., et al. (2015). **City Vehicle Routing Problem (City VRP): A Review**. IEEE Transactions on Intelligent Transportation Systems, 16.

- [38] Largo, S., S. O. and El Akkaoui, Z. (2020). **Green Vehicle Routing Problem: A Short Survey.** *2020 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD).*
- [39] Lim, S. F. W. T., Zhang, A. N., et al. (2016). **Three-Dimensional Vehicle Routing Problem for Urban Last Mile Logistics: Problem Formulation and Computational Analysis.** *2016 UKSim-AMSS 18th International Conference on Computer Modelling and Simulation (UKSim),* pages 252–257.
- [40] Loggi (2021). **LoggiBUD: Loggi Benchmark for Urban Deliveries.** *GitHub repository. Disponível em: <https://github.com/loggi/loggibud>. Acesso em: 01/08/2021.*
- [41] Lopes, P. H. M. (2011). **Uma solução para o problema de roteamento de embarcações de apoio "offshore" através da Metaheurística RTR.** *Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia - COPPE - UFRJ.*
- [42] Luo, J. and Chen, M. (2014). **Multi-phase modified shuffled frog leaping algorithm with extremal optimization for the MDVRP and the MDVRPTW.** *Computers Industrial Engineering, 72:*84–97.
- [43] Luo, M., Hou, X., and Yang, J. (2020). **Surface Optimal Path Planning Using an Extended Dijkstra Algorithm.** *IEEE Access, 8.*
- [44] MCCNET (2020). **MCC-NET - Indicador de Consumo do Mercado Online.** *Câmara Brasileira de Comércio Eletrônico (camara-e.net).*
- [45] Melo, A. G. L. R. T. (2015). **A aplicação de heurísticas para melhoramento de rotas de recolha de RSU no Barreiro.** *Instituto Superior Técnico da Universidade de Lisboa.*
- [46] Menezes, S. M. M. d. (2019). **Otimização de roteamento de veículos para coleta de lixo de descarte irregular em área pública.** *Universidade do Estado do Rio Grande do Norte - UERN.*
- [47] Moonsri, K., S. K. and Worasan, K. (2022). **A Novel Enhanced Differential Evolution Algorithm for Outbound Logistics of the Poultry Industry in Thailand.** *Journal of Open Innovation: Technology, Market, and Complexity, 8.*
- [48] Morais, V. W. C. d. (2012). **Algoritmos para o problema de roteamento de veículos com cross-docking.** *Instituto de Ciências Exatas da Universidade Federal de Minas Gerais - UFMG.*
- [49] Muzakir, A. and Hutrianto (2020). **Bellman-Ford Algorithm for Completion of Route Determination: An Experimental Study.** *Journal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI), 6(1).*

- [50] Narasimha, K. V. e. a. (2013). **An ant colony optimization technique for solving min–max Multi-Depot Vehicle Routing Problem.** *Swarm and Evolutionary Computation*, 13:63–73.
- [51] Negri, M. A. S. (2017). **Caminhos em um grafo e o Algoritmo de Dijkstra.** *Universidade Federal de Santa Catarina, UFSC, SC, Brasil.*
- [52] OpenStreetMap (2023). **O OpenStreetMap fornece dados de mapa para milhares de sites, aplicativos móveis e dispositivos de hardware.** *Disponível em: <https://www.openstreetmap.org/about>. Acesso em: 28/04/2023.*
- [53] OSRM (2023). **OSRM API Documentation.** *Disponível em: <https://project-osrm.org/docs/v5.24.0/api/>. Acesso em: 01/04/2023.*
- [54] Pantarotto, C. (2018). **Otimização do roteamento e programação de veículo e drone aplicada a last mile delivery.** *Universidade de São Paulo, Escola Politécnica, Departamento de Engenharia da Produção, São Paulo.*
- [55] Pureza, V., M. R. and Reimann, M. (2012). **Vehicle routing with multiple delivermen: Modeling and heuristic approaches for the VRPTW.** *European Journal of Operational Research*, 218:636–647.
- [56] Rangel, S. e. a. (2018). **Elementos de Teoria dos Grafos: Notas de Aula.** *UNESP Instituto de Biociência, Letras e Ciências Exatas - Departamento de Matemática Aplicada.*
- [57] Ravindran, V. and Shankar, S. (2015). **Systematic reviews and meta-analysis demystified.** *Indian J. Rheumatol*, 10:89–94.
- [58] Risald, A. E. M. and Suyoto (2017). **Best routes selection using Dijkstra and Floyd-Warshall algorithm.** *2017 11th International Conference on Information Communication Technology and System (ICTS)*, pages 155–158.
- [59] Sabet, S. and Farooq, B. (2022). **Green Vehicle Routing Problem: State of the Art and Future Directions.** *IEEE Access*, 10:101622–101642.
- [60] Salavati-Khoshghalb, M., Gendrau, M., et al. (2019). **A hybrid recourse policy for the vehicle routing problem with stochastic demands.** *EURO Journal on Transportation and Logistics*, 8(1):269–298.
- [61] Sampaio, R. M. and Yanasse, H. H. (2004). **Estudo e Implementação de Algoritmos de Roteamento sobre Grafos em um Sistema de Informações Geográficas.** *INFOCOMP Journal of Computer Science*, 3(1).

- [62] Schulz, A. A. H., S. C. and de Souza, A. S. (2019). **Uma abordagem meta heurística para o problema do VRPTW.** *IX Congresso Brasileiro de Engenharia de Produção, Ponta Grossa, PR, Brasil.*
- [63] Silva, D. F. and Sanches, A. L. (2009). **Aplicação Conjunta do Método de Dijkstra e Otimização Combinatória para Solução do Problema do Caixeiro Viajante.** *SEGeT – Simpósio de Excelência em Gestão e Tecnologia.*
- [64] Silva, J. A. S. and Mairink, C. (2019). **Inteligência artificial: aliada ou inimiga.** *LIBERTAS: Revista de Ciências Sociais Aplicadas, Belo Horizonte*, 9(2):64–85.
- [65] Stevenard, J. B. (2022). **Bellman-Ford.** <https://levelup.gitconnected.com/>.
- [66] Triana, Y. S. and Syahputri, I. (2018). **Implementation Floyd-Warshall Algorithm for the Shortest Path of Garage.** *International Journal of Innovative Science and Research Technology*, 3(2).
- [67] Varga, L. (2022). **Solutions to the routing problem: towards trustworthy autonomous vehicles.** *Artificial Intelligence Review.*
- [68] Wilhelm, V. E. (2013). **Problema de Roteamento de Veículos.** *Departamento de Engenharia de Produção – UFPR, PR, Brasil.*
- [69] Zanatto, N. E. (2021). **GRAPHEDU: Sistema Web para Visualização de Algoritmos de Grafos.** *Universidade de Caxias do Sul.*
- [70] Zhang, S. Z. and Lee, C. K. M. (2016). **Flexible vehicle scheduling for Urban Last Mile Logistics: The emerging technology of Shared Reception Box.** *2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM).*