CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Denis Augusto Nonato Vespasiano

ANÁLISE E IMPLEMENTAÇÃO DE UM PROCESSO COMPUTACIONAL PARA CRIAÇÃO DE GRAFOS APLICADO AO ROTEAMENTO DE VEÍCULOS

Timóteo

2021

Denis Augusto Nonato Vespasiano

ANÁLISE E IMPLEMENTAÇÃO DE UM PROCESSO COMPUTACIONAL PARA CRIAÇÃO DE GRAFOS APLICADO AO ROTEAMENTO DE VEÍCULOS

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Odilon Corrêa

Timóteo

2021

Denis Augusto Nonato Vespasiano

ANÁLISE E IMPLEMENTAÇÃO DE UM PROCESSO COMPUTACIONAL PARA CRIAÇÃO DE GRAFOS APLICADO AO ROTEAMENTO DE VEÍCULOS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais, campus Timóteo, como requisito parcial para obtenção do título de Engenheiro de Computação.

Trabalho aprovado. Timóteo, 17 de setembro de 2021:

Prof. Me. Odilon Corrêa da Silva
Orientador

Prof. Dr. João Batista Queiroz Zuliani
Professor Convidado

Profa. Esp. Andressa Oliveira Souza
Professora Convidada

Timóteo
2021

Dedico aos meus pais

Agradecimentos

Agradeço aos meus pais, aos amigos que contribuíram direta e indiretamente para que eu obtivesse sucesso neste trabalho e na minha vida, agradeço também a alguns professores da instituição pelos ensinamentos, pela paciência e pela dedicação. Um agradecimento especial ao meu orientador Odilon por ter topado este desafio.

“Você deve aproveitar os pequenos desvios ao máximo. Porque é onde você vai encontrar as coisas mais importantes do que as que você realmente quer”.

Ging Freecs

Resumo

A utilização de base de dados colaborativa para criação de algoritmos com código fonte aberto está se tornando cada vez mais comum e isso é um facilitador para algoritmos com fins acadêmicos. Este trabalho visou escolher uma base de dados colaborativa, tratar os dados e posteriormente criar uma estrutura de grafo. A estrutura de um grafo é comumente utilizada para tratar problemas de roteamento de veículos, uma vez que, com a estrutura montada, fica mais fácil visualizar um mapa. O resultado final gerado pode ser visualizado através de um mapa geográfico na ferramenta Google Earth.

Palavras-chave: Roteamento de veículos, Grafo, Dados Geográficos.

Abstract

The use of collaborative database to create algorithms with open source code is becoming more and more common and this is an enabler for algorithms for academic purposes. This work aimed to choose a collaborative database, process the data and later create a graph structure. The structure of a graph is commonly used to deal with vehicle routing problems, since, with the structure in place, it is easier to visualize a map. The final result generated can be visualized through a geographic map in the Google Earth tool.

Keywords: Vehicle Routing, Graph, Geographic Data

Lista de ilustrações

Lista de tabelas

Índice de algoritmos

Sumário

1 Introdução

No Brasil, cerca de 99,96% das prefeituras possuem serviços de manejamento de resíduos sólidos (IBGE, 2008). Como o serviço é custeado por um órgão público e, este órgão tem recursos limitados, organizar a realização do serviço fundamental para o planejamento dos gastos deste órgão. De acordo com Brasileiro e Lacerda (2008) a limpeza de uma cidade consome entre 7% e 15% do orçamento de um município e 50% destes valores destinados à limpeza são utilizados para a coleta e transportes de resíduos.

Com o aumento do desenvolvimento socioeconômico, cada vez mais os resíduos sólidos são gerados e a necessidade do transporte e descarte dos mesmos aumenta juntamente com o crescimento da população e o desenvolvimento do local (LOURENÇO, 2019).

Brasileiro e Lacerda (2008), constatou que os serviços de coleta de lixo doméstico também são de grande importância para a população, pois eles são responsáveis pelo transporte organizado dos resíduos até seu depósito, onde o lixo deveria ser tratado de forma adequada. Devido a essa importância e ao impacto ambiental gerado por essas operações que são realizadas frequentemente, o planejamento destes processos tende a melhorar o atendimento à população e diminuir os danos causados á natureza pelos resíduos que não são adequadamente tratados.

De acordo com Brasileiro e Lacerda (2008), a coleta e transporte dos resíduos atualmente são feitas utilizando métodos empíricos, que por sua vez, são conhecimentos que podem ser adquiridos através da tentativa e erro, da observação, da experiência ou do senso comum e dispensam a necessidade de comprovação científica. Geralmente, a roteirização da coleta dos resíduos sólidos domésticos é feita com base na experiência anterior do motorista, não havendo uma rota pré-definida.

Ao contrário dos métodos empíricos, os métodos matemáticos utilizam fórmulas e algoritmos para solucionar problemas e estes métodos podem ser trabalhados de forma manual ou computacional. Nos métodos trabalhados computacionalmente, são utilizados softwares chamados roteirizadores para executar os algoritmos. Um software roteirizador define de acordo com um conjunto de variáveis qual a melhor rota a ser escolhida (BRASILEIRO e LACERDA, 2008).

De acordo com Sherafat (2013), uma rota inicia na garagem dos veículos coletores e percorre uma rede (malha viária) utilizando um ou mais veículos com o objetivo de visitar os pontos de coleta. A sua definição e o método de encontrar a solução pode variar de acordo com as restrições que cada uma dessas variáveis impõem ao sistema.

O problema de roteamento de veículos pode ser tratado como um problema mono ou multiobjetivo. Um problema mono-objetivo busca a resolução de um problema em cima de um único objetivo. Em contrapartida os problemas multiobjetivo buscam uma solução com dois ou mais objetivos a serem otimizados (Santos, 2009).

O Problema de Roteamento de Veículos (PRV) é um problema NP-Completo e possui grande importância estratégica, por isso soluções aproximadas vêm sendo estudadas há várias décadas (HEINEN; OSÓRIO, 2006). Estas soluções buscam estipular qual é, dentre várias rotas, a melhor a ser escolhida levando em consideração as restrições impostas.

Não foi possível verificar na literatura, citada neste trabalho, os processos utilizados para criar grafos utilizando uma base de dados colaborativa. Assim, este trabalho propôs um processo computacional que, a partir de uma base de dados, monta um grafo do bairro da cidade, estipula os pontos de coleta e uma vez resolvido o problema de otimização apresenta o resultado de forma gráfica para os usuários.

1.1 Relevância

O desenvolvimento de métodos para a construção de softwares que encontrem rotas para veículos é altamente relevante devido a fatores prático e teórico (SHERAFAT, 2013). A relevância do fator prático se dá em virtude de uma grande parte dos municípios brasileiros realizarem a coleta dos resíduos sólidos domésticos de suas respectivas cidades e os transportarem para o depósito, logo, o planejamento deste tipo de serviço impactaria diretamente nos custos destes órgãos. Já o fator teórico se dá devido a grande parte das soluções exatas em problemas de roteamento serem difíceis de alcançar (SHERAFAT, 2013).

A fim de aplicar algoritmos de otimização em problemas de roteamento é importante ter em mãos uma estrutura de dados que representem adequadamente as variáveis de decisão e permitam avaliar as soluções factíveis em relação à(s) função(ões) objetivo(s) a fim de se obter soluções eficientes. Para problemas reais, criar esta estrutura digitando cada dado pode tornar o trabalho excessivamente custoso.

Geralmente a população não possui a informação formal da rota de coleta, quais os horários estimados para cada região e em muitos lugares também não se tem os pontos de coleta devidamente sinalizados. Sendo assim, o conhecimento do local e do horário da coleta de lixo é baseado nas experiências anteriores da população. Poder apresentar as soluções de problemas de otimização através de mapas ou aplicativos, como os aqui apresentados, com os quais a população está acostumada pode favorecer a eficiência da coleta.

1.2 Objetivos

1.2.1 Objetivo Geral

Este trabalho tem como principal objetivo propor e avaliar um processo computacional para a criação de grafos representativos de regiões de uma cidade a serem aplicados a problemas de roteamento de veículos.

1.2.2 Objetivos Específicos

Este trabalho tem os seguintes objetivos específicos:

1. Apresentar a aplicação de algoritmos de otimização em problemas de roteamento de veículos para a coleta de lixo;
2. Extrair e preparar os dados geográficos de uma base de dados;
3. Criar um grafo do bairro da cidade através dos dados obtidos da base de dados;
4. Registrar os pontos, distâncias e sentido das ruas do grafo;
5. Representar a estrutura do grafo através de um mapa geográfico;
6. Representar as soluções encontradas em problemas de otimização em um mapa geográfico.

1.3 Estrutura da Monografia

Esta monografia está estruturada conforme os itens abaixo e ao final deste trabalho, serão apresentados os resultados obtidos seguidos das conclusões.:

1. O capítulo 2 expõe trabalhos relacionados a este e apresenta os procedimentos para alcançar os resultados inicialmente propostos;
2. No terceiro capítulo estão os conceitos e conhecimentos teóricos necessários para o devido entendimento dos processos que foram utilizados neste trabalho;
3. O capítulo 4 conta com o desenvolvimento do projeto, bem como algoritmos utilizados, fluxogramas dos algoritmos e explicações dos procedimentos metodológicos que foram utilizados;
4. O quinto capíulo contém a avaliação do procedimento.
5. Por último, no capítulo 6 temos a conclusão do trabalho.

Ao final deste trabalho, serão apresentados os resultados obtidos seguidos das conclusões.

2 Trabalhos Relacionados

Ao longo dos anos o Problema do Roteamento de Veículos (PRV) vem sendo estudado, e apresentando diferentes tipos de abordagens. O levantamento bibliográfico se fez necessário para compreender as abordagens já existentes do problema e realizar uma análise das vantagens e desvantagens das metodologias adotadas para o PRV.

2.1 Roteamento de Veículos

Em seu artigo sobre a construção de circuitos e sua aplicação na roteirização de coleta de lixo domiciliar, Sherafat (2013) estudou o problema de roteamento de veículos em forma de grafo, que é uma estrutura formada por nós e arestas. O autor definiu que as vias de mãos únicas seriam representadas por um arco e as vias de mãos duplas por arestas e um cruzamento seria representado por um nó. Algumas das restrições do problema era que poderia haver uma ou mais instalações de abastecimento, a demanda poderia ser suprida por um ou mais veículos, o grafo da malha viária poderia ser orientado ou não orientado e a demanda pode estar em qualquer lugar do grafo e cada demanda poderia ter uma restrição temporal diferente. Com estas restrições, foi detectado que poderia ser obtidas algumas das formulações clássicas que formam problemas de roteamento, tais como: o problema do caixeiro viajante, o problema do carteiro chinês, entre outros. Tanto o problema do caixeiro viajante quanto o problema do carteiro chinês requer um caminho de custo mínimo que percorre todos os nós de um grafo. Após a implementação de um Sistema de Construtor de Circuitos (SCC), o autor comparou dados conforme a Tabela 1 das rotas antes e depois da utilização do sistema e concluiu que o sistema preenche os parâmetros desejados para ser considerada uma boa ferramenta para resolver roteamento de arcos.

Tabela 1 – Comparação de resultados entre percurso utilizado e percurso gerado pelo SCC.

	Percurso Atual - km	Percurso Gerado pelo SCC - km	Economia %
Dist. total percorrida sem penalizações	38,9	33,1	-14,90
Penalização de conversões para a direita	0,973	0,924	-5,03
Penalização de conversões para a esquerda	1,407	1,344	-4,47
Penalização de percurso em marcha-ré	1,100	0,100	-90,90
Penalização total associada ao percurso	3,480	2,368	-31,95
Dist. total percorrida com penalizações	42,4	36,1	-14,80

Fonte: (SHERAFAT, 2013)

Xavier (2010) realizou o estudo sobre a heurística para modelagem e minimização do consumo de combustível para rotas de coleta de lixo. O autor construiu uma rede de arcos a partir de uma base de dados reais e buscou uma forma de diminuir os custos de combustíveis, que segundo ele, é uma parcela importante dos custos. Foi detectado também que o

consumo de combustível possui características não lineares e seu valor depende da ordem em que as ruas são percorridas, implicando em um estudo de circuitos (ao invés de estudo de grafos) envolvendo funções não lineares. O uso de soluções clássicas como a do carteiro chinês são limitados quanto ao consumo de combustível por dois motivos: a capacidade do caminhão não é levada em consideração e sua formulação considera apenas o grafo Euleriano e neste caso o autor detectou que um circuito seria melhor do que um grafo. O CARP (do inglês *Capacitated Arc Routing Problem*) que pode ser enunciado como um veículo de capacidade limitada que deve atender demandas em arcos seguindo as rotas de menor custo total, apresenta uma complexidade muito maior do que o problema do carteiro chinês. O autor desenvolveu um algoritmo para realizar o cálculo das heurísticas e em determinadas partes foram usados algoritmos clássicos como o algoritmo de Dijkstra para encontrar o caminho mínimo para a descarga no aterro. A heurística foi aplicada no distrito de Maracanã em Montes Claros e conforme mostrado na Tabela 2, houve uma reduçao de 8,2% no consumo de combustível somente neste distrito.O autor ainda afirma ter ganhos que podem parecer não perceptíveis ou mensuráveis como a melhoria na poluição ambiental, maior disponibilidade da frota para manutenção e menor desgaste dos veículos.

Tabela 2 – Comparação de distância percorrida e consumo de combustível antes e depois da otimização da heurística .

	sem otimização	com heurística de otimização
distância percorrida (km)	43,2	41,04
consumo de combustível (l)	15,46	14,51

Fonte: (XAVIER et al., 2010)

Simas e Gomez (2004) descreveu o problema de roteamento de veículos como um dos Problemas de Otimização Combinatória (POC) devido a sua relevância prática e dificuldade utilizando dados não reais. O autor mencionou que o tempo de execução para obter uma solução cresce exponencialmente em relação as entradas e, para minimizar este problema, utiliza-se heurísticas para obter soluções em um tempo aceitável, isso faz também com que muitos esforços sejam colocados no desenvolvimento de heurísticas. A Pesquisa (ou Busca) Tabu (XU; KELLY, 1996) foi definida no trabalho como sendo uma técnica para resolver POC que tem uma rotina iterativa para construir vizinhanças com ênfase na proibição do bloqueio num ótimo local. A Pesquisa Tabu procura a melhor solução através de uma exploração agressiva, ou seja, ela escolhe o melhor movimento para a próxima iteração mesmo que isso não ajude a encontrar a solução. O modelo proposto para estudo foi um grafo não direcionado com vértices e arcos onde o vértice zero representa o depósito da frota e os demais vértices representa os clientes e em cada vértice há um peso não negativo que representa a demanda do cliente e este peso é utilizado para que cada cliente seja atendido por um único veículo. O autor fez uma matriz não negativa com a distância entre os vértices, pois o problema foi considerado simétrico, isto é, os custos são iguais em ambas as direções. Após a geração da solução inicial o autor aplicou a pesquisa Tabu e gerou um conjunto de rotas para otimizar a função objetivo do modelo proposto através de sucessivas gerações de vizinhanças. Em sua conclusão, o autor afirma que o refinamento de políticas para geração das vizinhanças, de

acordo com o objetivo modelo, permite que soluções de maior qualidade sejam encontradas.

Em seu estudo, Heinen e Osório (2006) aplicou algoritmos genéticos nos problemas de roteamento de veículos com o objetivo de descrever três heurísticas de aproximação para o PRV, sendo elas: a Heurística de Clarke e Wright (CLARKE; WRIGHT, 1964), a Heurística de Mole e Jameson (MOLE; JAMESON, 1976) e Algoritmos Genéticos utilizando dados não reais. A heurística de Clarke e Wright (CLARKE; WRIGHT, 1964) consiste em mapear todos os clientes até o depósito final e calcular (de acordo com a fórmula matemática da heurística) a distância de todos os clientes até o depósito. Uma desvantagem deste método é que ele não considera os nós internos de uma estrutura de grafo, logo, apenas os clientes das pontas são considerados.Já a heurística de Mole e Jameson (MOLE; JAMESON, 1976) é mais sofisticada e tenta reduzir a fragilidade da heurística de Clarke e Wright (CLARKE; WRIGHT, 1964), tornando os nós internos candidatos á economia, o autor utilizou o critério de escolher o nó mais próximo e assim montar uma rota. Após isso, foi utilizado um algoritmo genético para calcular as heurísticas. Conforme a Tabela 3, o autor concluiu que o algoritmo genético é mais lento em relação aos demais com os mesmos números de entradas, mas com poucas entradas ele encontrou um caminho com custo menor do que as demais heurísticas.

Tabela 3 – Comparação de resultados entre as heurísticas de Clark e Wright, MOle e Jameson e um algoritmo genético.

	Clarck e Wright		Mole e Jameson		Genético	
Tamanho	Tempo	Custo	Tempo	Custo	Tempo	Custo
50	0,00s	652,66	0,01s	521,40	16,91s	265,18
75	0,00s	1038,34	0,06s	587,46	74,56s	314,30
100	0,00s	1168,36	0,15s	704,19	128,25s	449,51
150	0,01s	1409,57	0,60s	849,73	215,36s	764,19
250	0,05s	1810,79	3,49s	1133,57	566,02s	1607,05
500	0,31s	2520,76	39,73s	1734,83	3168,57s	3744,18
1000	2,07s	3715,19	445,51s	2807,60	34913,63s	16988,64

Fonte: (HEINEN; OSÓRIO, 2006)

Mendes, Miranda e Wanner (2016) tratou o problema de roteamento de veículos com com transporte reativo a demanda utilizando dados não reais e, segundo os autores, trata do transporte de passageiros sob demanda buscando atender as diferentes rotas e horários dos passageiros e tem como objetivo a reduçao de inconveniências para os usuários. Na descrição do problema, os autores destacaram que os veículos começam e terminam suas rotas no depósito e a qualquer momento e de qualquer origem podem receber solicitações de transporte. As principais características do Problema do Roteamento de Veículos com Transporte Reativo a Demanda são por exemplo: um único depósito para iniciar e terminar a rota, os usuários especificam a janela de embarque e desembarque, em todos os pontos de paradas podem acontecer embarque e desembarque, entre outros. O problema foi modelado em forma de grafo com um conjunto de vértices e arestas em que o depósito foi colocado no vértice 0, e os demais pontos de embarque e desembarque foram colocados nos vértices. O custo de cada viagem foi colocado no arco que liga um vértice ao outro. O conjunto de veículos foi con-

siderado homogêneo e um veículo só pode retornar a um vértice já visitado para atender uma solicitação e a visíta ao vértice nao pode ser consecutiva. Se todas as demandas forem solicitadas antes do início do processo do roteamento, o problema se torna estático, caso contrário o problema se torna dinâmico. O algoritmo utilizado pelos autores foi o NSGA-II (*Non Sorting Genetic algorithm*) (DEB et al., 2002), que difere dos demais algoritmos genéticos na estrutura do operador de seleção. Além disso, ele trabalha com duas populações ao mesmo tempo em vez de uma. Foram utilizadas cinco funções objetivo e para verificar o conflito e a redução dos objetivos propostos, os autores utilizaram a ferramenta chamada Árvore de Agregação. A conclusão chegada pelos autores foi de que a função relacionada a empresa foi agregada em um único objetivo e a perspectiva dos passageiros e do motorista foi agregada em outro objetivo. Após a redução dos objetivos, foi aplicado o algoritmo NSGA-II (DEB et al., 2002) ao problema bi-objetivo e os resultados apresentaram estabilização e foi capaz de resolver o problema de forma eficiente.

Os algoritmos utilizados nos trabalhos citados anteriormente necessitam de uma estrutura de dados para fazer de forma eficiente o que é proposto. A maioria dos autores que trabalharam com uma malha viária utilizaram um grafo. A estrutura de um mapa se assemelha ao modelo de um grafo, onde as ruas podem ser representadas pelas arestas de um grafo e as esquinas são os vértices, facilitando assim a conversão dos dados nesta estrutura de dados. Vale destacar que nenhum dos trabalhos que foram citados nesta seção apresenta um procedimento sobre a extração e preparo da base de dados.

3 Referencial Teórico

Nas próximas seções serão apresentados alguns conceitos necessários para o desenvolvimento deste trabalho.

3.1 OpenStreetMap

O OpenStreetMap é uma plataforma *open source*, isto é, uma plataforma de dados abertos que podem ser utilizados por qualquer pessoa e, esta plataforma, contém dados geográficos de todo o mundo. A plataforma é atualizada constantemente pelos usuários da mesma, ou seja, cada usuário tem permissão para inserir e atualizar os dados caso seja necessário.

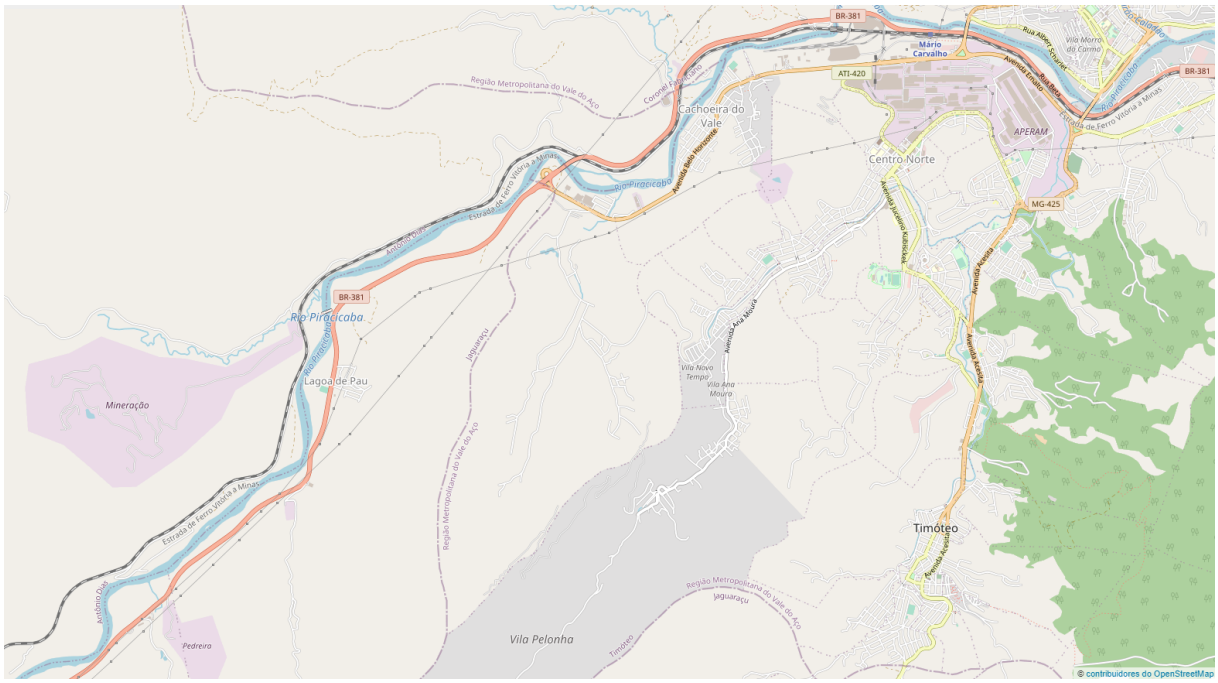
A plataforma *web* possui dados como a latitude e a longitude, nome de ruas e pontos turísticos e nela os dados são representados de forma gráfica através de um mapa da Terra. Na Figura 1 é possível ver o exemplo de algumas marcações no mapa da cidade de Timóteo como a área de mineiração que fica próximo á cidade, o rio Piracicaba que passa pelo bairro Cachoeira do Vale, áreas verdes e demais marcações e pontos turísticos da cidade de Timóteo. Além de apresentar os dados de forma gráfica, o OpenStreetMap ainda permite realizar o *download* dos dados de determinada região, assim, após escolher uma área do mapa, é possível obter o arquivo do tipo OSM (OpenStreetMap), que é semelhante ao arquivo do tipo KML, porém é um arquivo próprio do OpenStreetMap.

3.2 KML

De acordo com a documentação fornecida pela Google através da plataforma Google Developers, o KML (*Keyhole Markup Language*) é um formato de arquivos usado para exibir informações geográficas em um navegador da Terra, como Google Maps, Google Earth e o Waze. Ele utiliza um sistema de *tags* aninhadas para organizar e estruturar seus arquivos. Existe uma documentação apresentada e mantida pela Opengis que lista todas as *tags* que são opcionais e obrigatórias em um arquivo KML. Na Figura 11 podemos ver um exemplo de um arquivo que foi utilizado para realizar a marcação de um ponto. A *tag* Point informa que será marcado um ponto e a Coordinates representa as coordenadas em que este ponto será marcado. O primeiro valor refere-se a longitude do ponto e o segundo a latitude, ambos são atributos obrigatórios. Já o terceiro ponto é a altutide do ponto e este é um atributo opcional, ou seja, caso não seja informada a altitude, por padrão o ponto será marcado na altitude 0.

Softwares como o Google Earth e o Google Maps, utilizados para realizar marcações nos mapas geográfico, exportam e importam essas marcações através desta estrutura de arquivos.

Figura 1 – Mapa da cidade de Timóteo



Fonte: OpenStreetMap

Figura 2 – Exemplo de arquivo KML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
    <Placemark>
        <name>
            Simple placemark
        </name>
        <description>
            Attached to the ground. Intelligently places itself at the height of the underlying terrain.
        </description>
        <Point>
            <coordinates>
                -122.0822035425683,37.42228990140251,0
            </coordinates>
        </Point>
    </Placemark>
</kml>
```

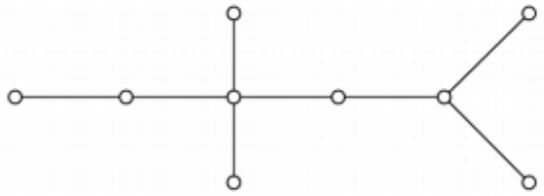
Fonte: Google Developers

3.3 Grafo

Um grafo é uma estrutura matemática abstrata que permite que as relações entre os elementos sejam mostradas de forma gráfica e é formado por um conjunto de vértices e um conjunto de arestas (PECLY; MELLO, 2013). Os vértices guardam os dados que relacionamos enquanto as arestas representam as relações entre estes dados. A Figura 3 representa a estrutura explicada anteriormente.

Esta estrutura pode ser orientada ou não orientada. O grafo orientado possui setas representando o sentido do fluxo de dados e, este sentido, deve ser respeitado. Já na estrutura

Figura 3 – Representação de um grafo não direcionado.



Fonte:(FRITSCHER, 2011)

não orientada o fluxo de dados pode ocorrer em ambos os sentidos. A figura 3 representa o grafo não direcionado e ao adicionar uma direção nos vértices, ele se torna um grafo direcionado..

Os valores que acompanham as arestas são denominados pesos, estes valores representam o custo de sair de um vértice e ir até o outro. Os vértices podem conter simplesmente um nome, caso o interesse no grafo seja apenas o caminho percorrido por exemplo, mas também podem conter informações adicionais como a capacidade de armazenamento do local, lotação e etc. Alguns problemas clássicos da computação como o problema do caixeiro viajante (onde um viajante tenta determinar a menor rota para visitar uma série de cidades e retornar a cidade de origem ao visitar todas) e o problema das pontes de *Königsberg* (que consiste no problema de uma cidade com sete pontes e discutia-se a possibilidade de atravessar todas as pontes sem repetir nenhuma) são alguns dos clássicos problemas da computação que são trabalhados utilizando a estrutura do grafo.

3.4 Algoritmos para calcular rotas

Alguns dos algoritmos utilizados para determinar o caminho mais curto entre dois vértices de um grafo, segundo Atzingen et al. (2012), são, por exemplo, o Algoritmo de Dijkstra (SALAS, 2008), o A-Estrela (HART; NILSSON; RAPHAEL, 1968) e o Radix Heap (AHUJA et al., 1990). Após um estudo comparativo, Atzingen et al. (2012) concluiu que o algoritmo de Dijkstra obteve os melhores resultados se comparados com os demais citados. Porém, problemas reais podem trazer condições e restrições práticas que devem ser consideras na definição do melhor caminho, que pode não ser necessariamente o mais curto. Em casos como esses, a abordagem que leva em consideração apenas a distância total percorrida pode ser insuficiente ou inadequada para a obtenção de resultados úteis (SANTOS, 2017).

O NSGA-II é um algoritmo baseado em classificação por não dominância para calcular o valor da aptidão dos elementos da população genética (OLAZAR, 2007). A classificação por não dominância é feita comparando cada indivíduo aos demais da população para determinar aqueles não dominados. O primeiro elemento é separado da população e o procedimento é repetido para encontrar os demais indivíduos não dominados e os classificar de acordo com a

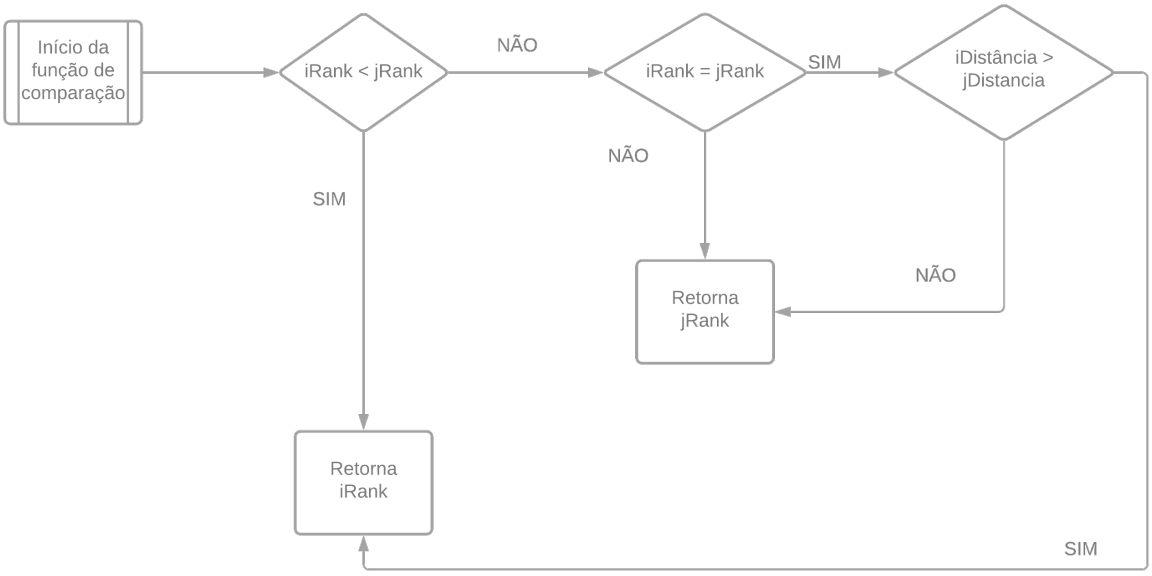
ordem de dominância.

Este algoritmo também possui uma função denominada *crowded-comparison*, que guia o processo de seleção em vários estágios. De acordo com Deb et al. (2002), a função guia uniformemente em direção a uma frente ótima e assume que cada indivíduo i possui dois atributos:

- *Rank* ou Hierarquia de não dominância ($iRank$)
- *Crowding distance* ou distância de agrupamento

A comparação fica conforme o algoritmo da Figura 4

Figura 4 – Comparação feita pela função de *Crowded*



Fonte: Adaptado de Deb Et. Al. 2002

Em seu procedimento geral, o NSGA-II inicializa a população P_0 de forma aleatória e com tamanho N. A população é organizada de acordo com o critério de não-dominação. Para cada solução é atribuído um valor de $iRank$, onde, 1 é melhor do que 2 e assim por diante. Posteriormente, é aplicado um método de seleção, recombinação e mutação por torneio para, assim, criar uma população descendente Q_0 também de tamanho N. Aplica-se o elitismo ao comparar a população atual à melhor população não dominada encontrada previamente. O procedimento é diferente após primeira geração (DEB et al., 2002). Obtêm-se uma população de $R_t = P_t \cup Q_t$ de tamanho $2N$ que é ordenada de acordo com os critérios de não dominação. Após a ordenação, as primeiras N melhores soluções irão integrar $F1$ (conjunto solução 1) e as demais farão parte de $F2$. Se o tamanho de $F1$ é menor do que N todos os seus elementos formam P_{t+1}. Caso haja espaço restante nas populações é preenchido pelos melhores indivíduos não dominados. O procedimento se repete até que nenhum conjunto possa ser mais acomodado (OLAZAR, 2007). Após isso, aplica-se os operadores de seleção, cruzamento e

mutação á nova população P_{t+1} para formar a população Q_{t+1} de tamanho N. Assim, após terminar os processos, ele retorna a melhor solução encontrada. O pseudocódigo do algoritmo NSGA-II fica conforme o Algoritmo 1.

Inicializar (P_0);
$t_0 = 0$;
Avaliar (P_0);
Quando (não critério de parada) fazer
  $R_t = P_t \cup Q_t$; combina pais e filhos da população;
  F = **ordenamento_não-dominado** (R_t),
  $F = (F_1, F_2, ...)$, todos os frentes não dominados de R_t;
  $P_{t+1} = 0; i = 1$; Nova população vazia;
  Quando ($|P_{t+1}| + |F_i| \leq N$); fazer preencher a população pai.;
  Calcular a *crowding-distance* (F_i);
  $P_{t+1} = P_{t+1} \cup F_i$; inclui o *i-ésimo* frente não-dominado na população pai;
  $i++$;
  ordenar(F_i,$\succ_n$); função que ordena em forma descendente usando $\succ_n$;
  $P_{t+1} = P_{t+1} \cup F_i[(1 : (N - |P_{t+1}|)]$,
  procura os primeiros ($N - |P_{t+1}|$) elementos de F_i ;
  Q_{t+1} = reprodução (P_{t+1}),
  aplicação de operadores genéticos para criar a nova população Q_{t+1} ;
  $t = t + 1$; incremento do contador de geração;
  P_t = Nova população ;
  Retorna a melhor solução encontrada;

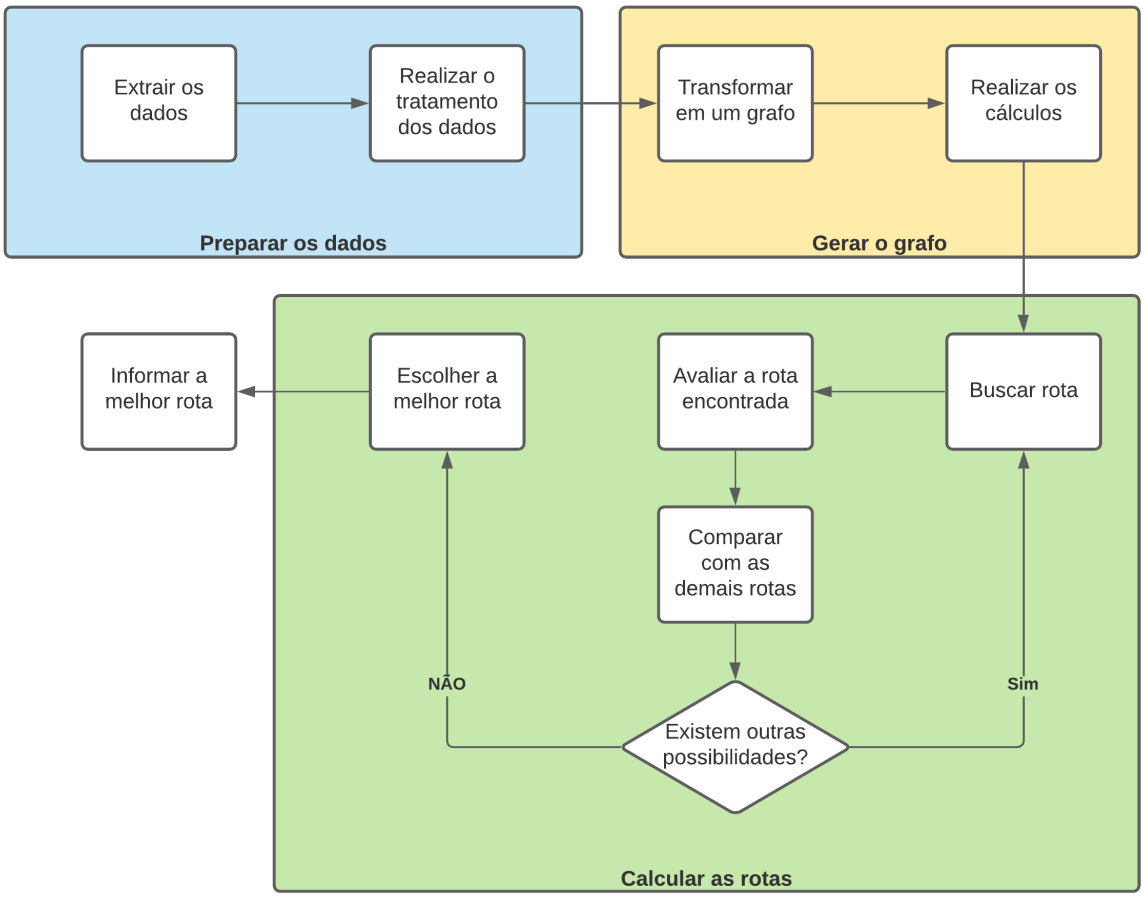
Algoritmo 1: Algoritmo NSGA-II (Fonte: Olazar,(2007))

Qualquer que seja o algoritmo aplicado, é necessário poder representar as possíveis rotas e valorar cada trecho de acordo com as variáveis de decisão a serem utilizadas.

4 Desenvolvimento

Para a realização dos testes do processo computacional se fez necessário a extração de dados reais de uma plataforma que os disponibilizasse de forma gratuita. Após algumas pesquisas, o OpenStreetMap se mostrou ser a plataforma mais viável para a extração dos dados que foram utilizados nos testes dos algoritmos devido ao fato de não limitar os dados extraídos. Os algoritmos seguiram a o passo a passo conforme a Figura 5:

Figura 5 – Fluxograma do funcionamento do processo computacional



Fonte: O Autor

Todos os códigos e arquivos desenvolvidos e utilizados no trabalho estão disponíveis no repositório: <https://github.com/dvespasiano/tcc>

4.1 Preparar os dados do OpenStreetMap

4.1.1 Extrair os dados

Para realizar o *download* dos dados do OpenStreetMap basta se dirigir até o *site* da plataforma e usar a opção de exportar e selecionar a área que deseja exportar. Caso a área a ser exportada seja grande, a própria plataforma indica as plataformas alternativas para conseguir exportar os dados solicitados que são: a API Overpass, a Planet OSM, a Geofabrik que são citadas na página do OpenStreetMap.

Os dados disponibilizados pelo OpenStreetMap são exportadas em um arquivo no formato *OSM*, mas apesar desta extensão o arquivo possui a sua estrutura semelhante a um arquivo *KML*. Na Figura 6, podemos ver como o arquivo é estruturado pela plataforma.

Figura 6 – Exemplo de um arquivo OSM

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.8.3 (1953531 spike-07.openstreetmap.org)" copyright="OpenStreetMap and contributors" attribution="http://www.openstreetmap.org/copyright"
license="http://opendatacommons.org/licenses/odbl/1-0/">
 <bounds minlat="-19.5920000" minlon="-42.6541000" maxlat="-19.5751000" maxlon="-42.6336000"/>
 <node id="246671293" visible="true" version="8" changeset="51949262" timestamp="2017-09-11T17:35:34Z" user="nyuriks" uid="339581" lat="-19.5771395" lon="-42.6436108">
  <tag k="name" v="Timóteo"/>
  <tag k="place" v="town"/>
  <tag k="population" v="81119"/>
  <tag k="source" v="geominas"/>
  <tag k="wikidata" v="Q737237"/>
  <tag k="wikipedia" v="pt:Timóteo"/>
 </node>
 <node id="353291797" visible="true" version="3" changeset="30216635" timestamp="2015-04-14T16:21:13Z" user="zeninguem" uid="2082896" lat="-19.5831751" lon="-42.6468036">
  <tag k="source" v="IBGE"/>
 </node>
 <node id="353291799" visible="true" version="5" changeset="63162813" timestamp="2018-10-03T14:37:36Z" user="zeninguem" uid="2082896" lat="-19.5801592" lon="-42.6448582">
  <tag k="source" v="IBGE"/>
 </node>
 <node id="353291802" visible="true" version="5" changeset="94699642" timestamp="2020-11-24T10:42:47Z" user="QuidiVidi" uid="10811281" lat="-19.5872583" lon="-42.6478154">
  <tag k="source" v="IBGE"/>
 </node>
```

Fonte: O Autor

4.1.2 Tratar os dados

Antes de realizar o tratamento dos dados, se faz necessário entender como é a sua estrutura. Cada *tag* representa um tipo de objeto no arquivo. Entende-se por *tag* a palavra que é antecedida por um símbolo de menor $(<)$, que demarca a abertura e o símbolo de maior $(>)$ que indica o fechamento da respectiva *tag*. Na lista abaixo estarão as *tags* mais comuns e utilizadas no arquivo.

1. A *tag xml* indica o formato do arquivo e seus atributos indicam tanto a versão do *xml* quanto o tipo do alfabeto utilizado pelo arquivo;
2. As linhas iniciadas por *osm* indicam o repositório, licenças e versão do sistema utilizado quando foi feita a extração dos dados.
3. Cada *node* pode possuir uma série de atributos não obrigatórios e também possui alguns atributos que são importantes para a utilização dos dados geoespaciais e são eles:
   - O *id* que representa a identificação única daquele ponto e é representada por números;
   - *Visible* que é um atributo do tipo boolean e indica se o ponto está visível (caso seu valor seja *true*) ou não (caso seu valor seja *false*);

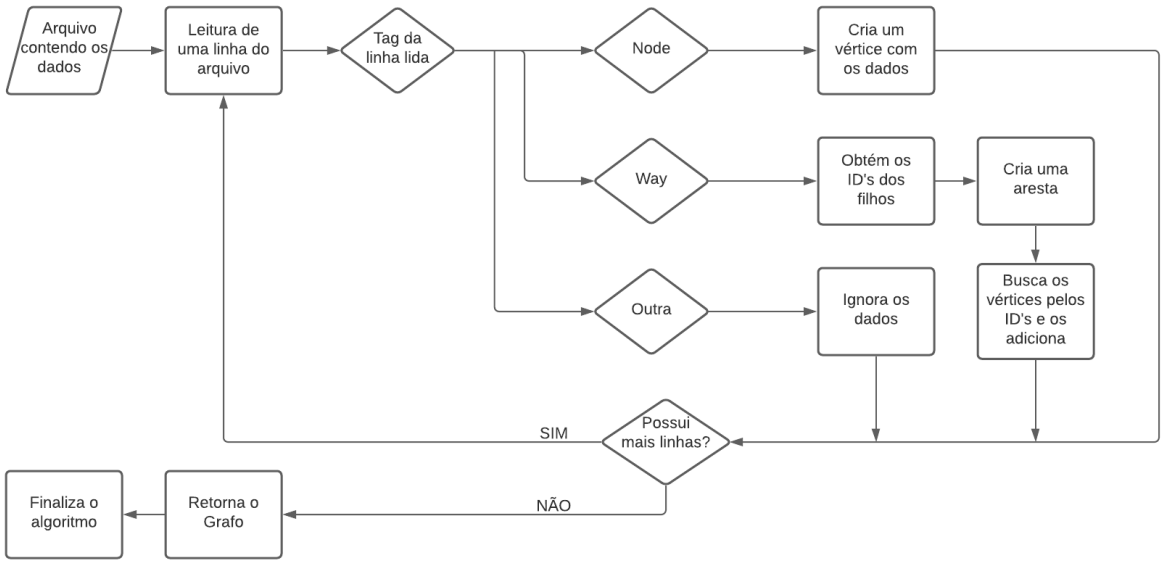
- O atributo *lat* representa a latitude geográfica daquele ponto e é dada em números;
- O atributo *lon* representa a longitude do ponto e é dada em números;
- Já a *tag* representa diversos outros tipos de dados e possui 2 campos sendo um deles o nome e o outro o valor do atributo.

4. As *ways* indicam as ruas e essas possuem os mesmos atributos básicos dos nós, mas possuem um array de filhos:
   - Os filhos das *ways* possuem a *tag nd*, que tem o atributo *ref* . Esse atributo é o número do *id* de um node.

Sendo assim, é possível entender que uma *way* é uma rua e que essa rua é formada pela união de vários pontos que estão presentes nela.

Utilizando as definições informadas anteriormente, foi confeccionado um algoritmo para realizar o tratamento dos dados. Para trabalhar estes dados de forma mais fácil, o arquivo *OSM* foi transformado em um grafo e a linguagem utilizada foi o JavaScript, uma vez que a grande maioria dos dados trocados pela internet estão em formato JSON (*JavaScript Object Notation*). Seguindo as definições do que cada *tag* significa, o pseudocódigo deste tratamento de dados para retornar um grafo em formato JSON a partir deste KML ficou conforme a Figura 7.

Figura 7 – Fluxograma do pseudocódigo do tratamento dos dados

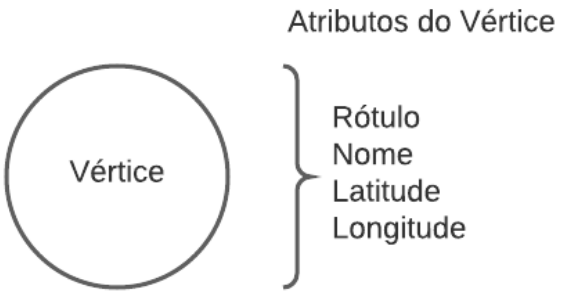


Fonte: O Autor

4.2 Gerar o grafo

Apesar de ter sua estrutura simples, as partes que compõem o grafo possuem atributos necessários para que os dados obtidos sejam armazenados e guardados de forma correta e de fácil acesso.

Figura 8 – Estrutura do vértice montado pelo algoritmo



Fonte: O Autor

Cada vértice do grafo é preenchido por um nó, e este nó representa um ponto no grafo. Para que os cálculos da distância entre dois nós seja feita, cada um deles precisa guardar a sua posição geográfia que é dada por sua latitude e sua longitude. Para que cada vértice seja único e possa ser encontrado de forma direta, cada um possui um identificador único, que foi chamado de "nome". O nome é um ID único que é informado pelo banco de dados ao requisitar as informações e este ID é composto apenas por números. Os vértices possuem também um *nickname*, que foi chamado de rótulo e tem como objetivo identificar se o ponto se trata de uma rua, de um comércio, de um uma igreja, entre outros. Assim, a estrutura do vértice fica conforme a Figura 8.

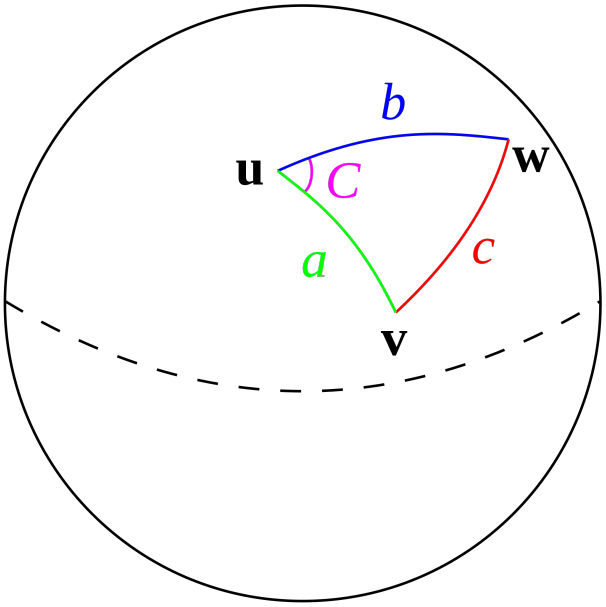
A aresta tem como objetivo ligar os vértices, ou seja, a aresta representa no mapa o caminho a ser percorrido de um ponto ao outro. Cada vértice precisa guardar o cálculo da distância entre os pontos que estão ligados por ela, logo, toda aresta possui dois vértices e, com esses dois vértices, é possível realizar o cálculo da distância entre eles.

A distância entre dois pontos geográficos foi calculada por Prasetya et al.,(2020) utilizando a Fórmula de Haversine (TUSSA'DDIA; HARLINDA; MUDE, 2021), assim, para realizar o cálculo da distância entre dois vértices esta fórmula também foi utilizada neste trabalho. A representação da fórmula pode ser vista na Figura 9. Na Equação 4.4 está a equação utilizada que é uma fórmula simples para cálculo da distância em uma superfície esférica. A constante R possui o valor de 6.372,795477598 km, valor este que é a média quádrica da terra, isto é, a cada distância R devemos considerar a curvatura da terra para realizar o cálculo da distância. Ainda na Equação 4.4 , a função $atan^2(x, y)$ com duas variáveis retorna o ângulo θ entre os dois pontos. A curvatura da Terra é levada em consideração pois os cálculos da distância entre os pontos estão sendo feitos através da latitude e da longitude. Após realizar o cálculo da distância, este valor é guardado na variável "peso" do vértice.

$$distanciaLatitude = (LatitudeB - LatitudeA) * \frac{\pi}{180} \tag{4.1}$$

$$distanciaLongitude = (LongitudeB - LongitudeA) * \frac{\pi}{180} \tag{4.2}$$

Figura 9 – Representação gráfica da Fórmula de Haversine



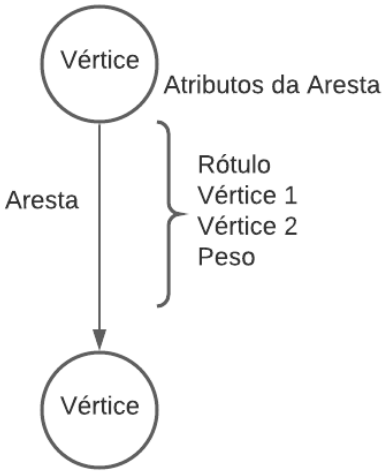
Fonte: (PRASETYA et al., 2020)

$$a = sin(\frac{distanciaLatitude}{2})^2 + cos(LatitudeA * \frac{\pi}{180}) * cos(LatitudeB * \frac{\pi}{180}) * sin(\frac{distanciaLongitude}{2})^2 \tag{4.3}$$

$$distancia = R * 2 * atan^2(\sqrt{a}, \sqrt{(1-a)}); \tag{4.4}$$

A estrutura da aresta ficou conforme a Figura 10:

Figura 10 – Estrutura da aresta montado pelo algoritmo



Fonte: O Autor

Assim, com os vértices montados e as arestas os ligando, forma-se o grafo através dos dados geoespaciais fornecidos pela plataforma openstreetmap.

4.3 Rotas

Após o grafo montado e com os valores de distâncias preenchidos nas suas devidas arestas,

Num problema monobjetivo a melhor rota encontrada será retornada. Num problema multiobjetivo podem ser retornadas todas as soluções pareto não-dominadas ou aplicar um método de tomada de decisão que escolha a sollução dentre o conjunto de soluções retornadas. Em qualquer dos casos é importante representar esta solução de modo que os decisores ou o público compreenda facilmente.

4.4 Exportar o resultado

Com a rota retornada como solução do problema de otimização, pode ser montado um arquivo de extensão KML. Um arquivo KML é um arquivo que é usado pelos sistemas de mapa afim de preservar informações como rotas, locais, fluxogramas, entre outros em mapas, e por apresentar um padrão, pode ser importado em diversos *softwares*. Um arquivo KML apresenta uma estrutura bem semelhante a um arquivo XML, assim, a codificação de sua estrutura é padronizada, logo, o sistema feito neste trabalho gerou um arquivo KML como resultado afim de apresentar de forma simples sua resposta. Na figura 11 temos um exemplo de um arquivo kml gerado pelo sistema onde as respectivas *tags* apresentam os significados listados abaixo:

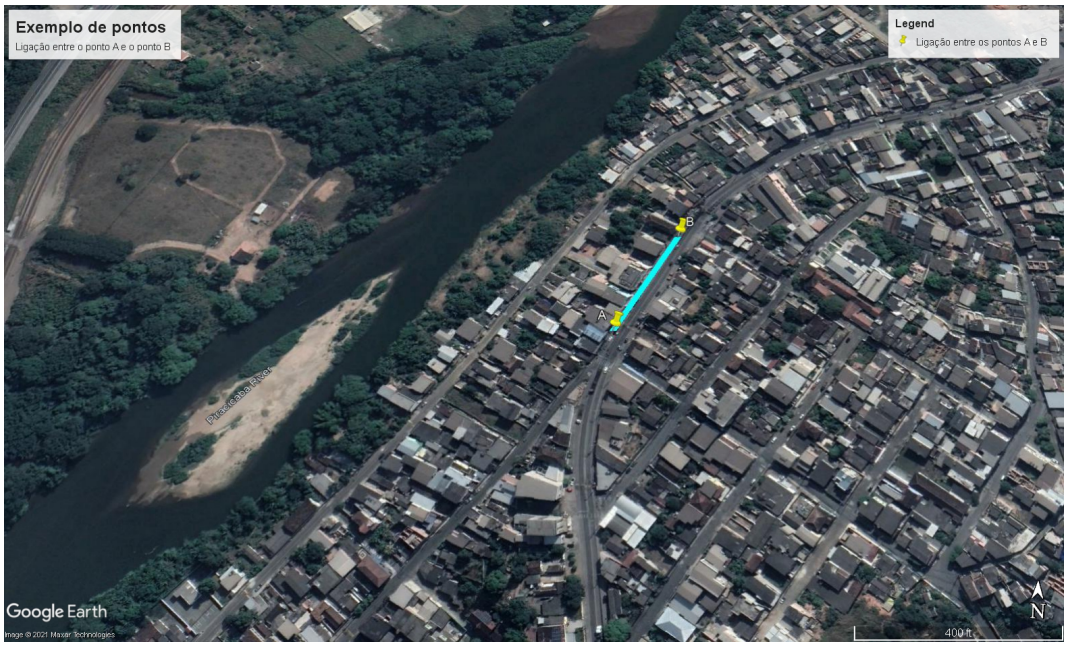
- A *tag name* é o nome a ser exibido;
- *Placemark* significa que será marcado um ponto no mapa;
- *Visibility* é a camada em que o ponto será exibido, para este trabalho utilizamos apenas uma camada;
- *Point* é o tipo de ponto marcado pelo mapa, neste caso foi escolhido o ponto simples;
- Dentro da tag *point* temos a tag *coordinates* que ficam as coordenadas geográficas (latitude e longitude) de onde o ponto será marcado no mapa;

Figura 11 – Exemplo de arquivo kml

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2" xmlns:gx="http://www.google.com/kml/ext/2.2" xmlns:kml="http://www.opengis.net/kml/2.2" xmlns:atom="http://www.w3.org/2005/Atom">  "
<Folder>
    <name>alfaville</name>
    <visibility>0</visibility>
    <Placemark>
        <name>-19.5559522,-42.5875635</name>
        <visibility>0</visibility>
        <Point>
            <coordinates>-42.5875635,-19.5559522,0</coordinates>
        </Point>
    </Placemark>
    <Placemark>
        <name>-19.558855,-42.5882609</name>
        <visibility>0</visibility>
        <Point>
            <coordinates>-42.5882609,-19.558855,0</coordinates>
        </Point>
    </Placemark>
    <Placemark>
        <name>-19.5572387,-42.5879417</name>
        <visibility>0</visibility>
        <Point>
            <coordinates>-42.5879417,-19.5572387,0</coordinates>
        </Point>
    </Placemark>
```

Fonte: O Autor

Figura 12 – Exemplo de marcação de pontos ligados



Fonte: O Autor

4.5 Visualizar o resultado

Assim, a resposta final do algoritmo é um arquivo, e basta importá-lo para visualizar as informações no mapa conforme o exemplo da Figura 12, temos os pontos A e B e uma representação ilustrada no mapa da ligação do grafo presente no arquivo de resultados do algoritmo.

5 Avaliação do procedimento

Nas próximas seções estão divididos os estudos de caso feitos na cidade de Timóteo. Estes estudos de casos foram divididos em bairros e em cada bairro foram feitos os procedimentos descritos na seção anterior para a extração dos dados, montagem do grafo e posteriormente a montagem das rotas para realizar a avaliação dos procedimentos feitos.

Timóteo é uma cidade com uma população estimada em 90.568 habitantes e possui densidade demográfica de 562,70 hab/km^2. A cidade pertence á microrregião de Ipatinga e a mesorregião do Vale do Rio Doce (IBGE, 2020). Os dois bairros para os testes foram escolhidos de forma arbitrária.

Para o tratamento dos dados foram utilizados algoritmos nas linguagens de programação Java e Javascript, linguagens essas que são semelhantes e possuem alguns métodos já implementados de estrutura de dados que são explicados juntamente com os algoritmos em que serão utilizados.

Durante o desenvolvimento das técnicas que serão explicadas nos capítulos posteriores se fez necessário o uso de um computador. Abaixo as configurações da máquina utilizada:

- Processador Intel Core i7-9750H 2.60 Ghz;
- Memória RAM 16GB;
- Placa de vídeo geforce GTX 1650;
- SSD *NVMe* 240GB;
- HD 1TB;
- Sistema Operacional *Windows* 10 x64;

5.1 Caso 1: Rota do bairro Alphaville de baixo

O primeiro bairro escolhido foi o Alphaville de baixo. Este bairro possui um caminho para entrada e saída, assim, é necessário passar pelo ponto de entrada do bairro ao final da rota.

O grafo do bairro foi construído como mostra a Figura 13. A rota do bairro foi apresentada graficamente como mostra a Figura 14. Uma característica do algoritmo que é possível ver na Figura 13 é que os pontos de coleta são enumerados de acordo com a sua aparição na base de dados, ou seja, a rua que aparece primeiro no arquivo extraído do OpenStreetMap recebe o primeiro ponto e a sua sequência continua até o fim daquela rua e o processo se repete até finalizar os pontos.

Figura 13 – Grafo do Bairro Alphaville de baixo

Fonte: O Autor

Após a montagem do grafo, foi simulada uma rota que passasse por mais pontos e cem ciclos. O resultado pode ser visto na Figura 14 onde o ponto inicial e o ponto final estão destacados na imagem com cores diferentes.

Uma limitação encontrada na base de dados colaborativa foi que algumas ruas não apresentam seu sentido, assim, as ruas deste bairro estão todas com sentido indefinido no OpenStreetMap.

Uma observação que se faz necessária neste estudo de caso é que a preferência é sempre pelo menor caminho final com maior número de pontos de coleta e não pela menor distância do trecho, isto é, mesmo que um trecho seja menor, se ele possuir menos pontos de coleta e impactar mais para que o caminho final seja maior ou possuir ciclos, ele não será escolhido.

Figura 14 – Rota do bairro Aphaville de Baixo

Fonte: O Autor

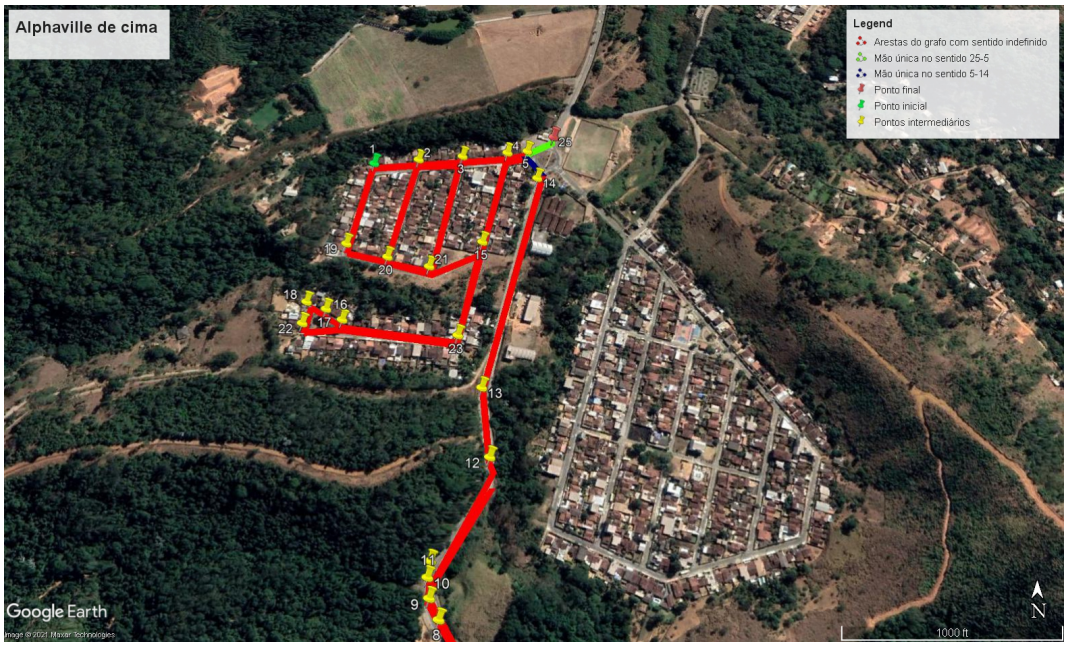
5.2 Caso 2: Rota do bairro Alphaville de cima

O segundo bairro escolhido foi o Alphaville de cima. Este bairro também possui apenas uma entrada.

Na Figura 15 podemos ver a representação do bairro em forma de grafo. A sua rota foi calculada como representado na Figura 16. Ainda na Figura 15 podemos observar que as ruas da parte do meio do bairro só podem ser acessadas através das extremidades, logo, para se passar por elas durante a rota, seria necessário repetir algum ciclo. Neste caso de teste foi possível observar que dois trechos das ruas apresentavam seus sentidos na base de dados, sendo assim, os mesmos foram representados com cores diferentes no grafo e seus sentidos foram mostrados na legenda.

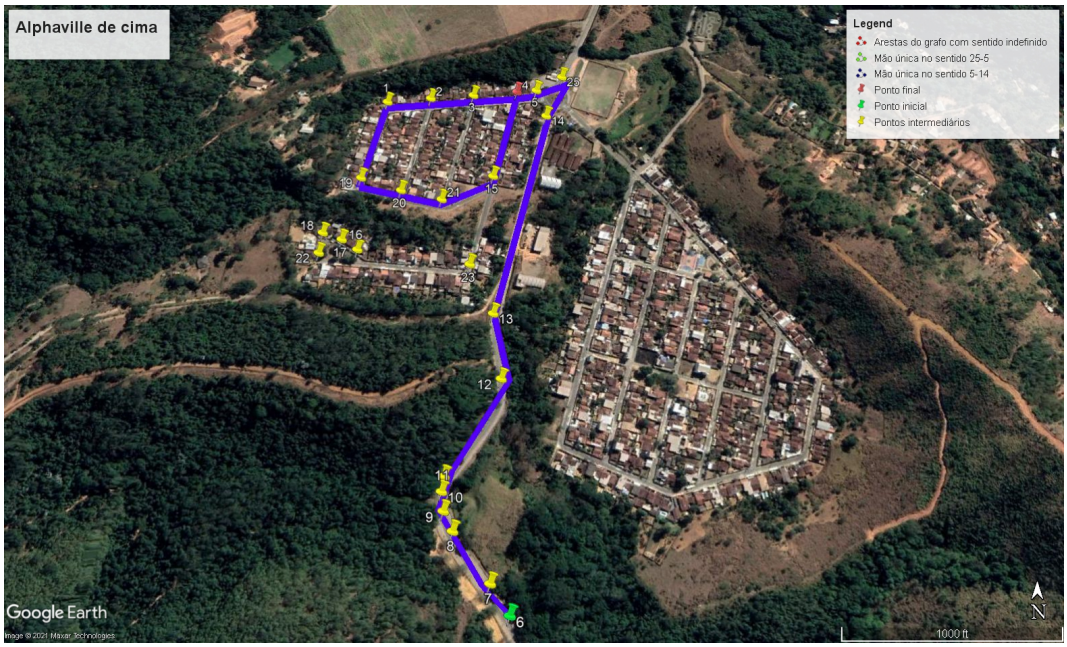
Uma limitação que pôde ser observada, foi que o algoritmo marcou pontos de coleta onde não existem residências. Regiões como na parte inferior da Figura 16 e destacada na Figura 17, cujo caminho é único, pode ser retirada do problema a fim de diminuir a dimensão do problema e depois acrescentada à rota final.

Figura 15 – Grafo do Bairro Alphaville de cima



Fonte: O Autor

Figura 16 – Rota do bairro Alphaville de cima



Fonte: O Autor

5.3 Avaliação dos resultados

Os resultados obtidos nos casos de teste pelo processo computacional e com as informações obtidas através da plataforma OpenStreetMap puderam ser vistos e análisados em um mapa geográfico, uma vez que o processo conseguiu gerar o grafo das ruas de um determinado bairro e posteriormente conseguiu exportá-los da forma esperada.

Uma das características importantes que foram observadas é que a rota não repetiu os

Figura 17 – Destaque de parte do bairro Alphaville de cima que ficou ausente da rota

Fonte: O Autor

pontos, visto que essa foi uma das condições passadas inicialmente no algoritmo. Os pontos foram devidamente marcados seguindo uma sequência numérica gerada a partir da ordem em que as ruas estavam apresentadas na base de dados do OpenStreetMap. Os lugares em que os pontos iriam repetir foi utilizado um ponto já existente. Vale ressaltar que como a fórmula de Haversine calcula a distância entre dois pontos através de suas coordenadas geográficas (latitude e longitude) em linha reta, caso a via apresente uma mudança em seu sentido, ela será dividida neste ponto para que os cálculos sejam mais próximos da realidade.

Uma limitação que foi observada nos casos anteriores foi a presença de pontos de coleta onde não temos residências marcadas no mapa. A base de dados do OpenStreetMap não disponibiliza esta informação, logo, o algoritmo irá marcar pontos de coleta em locais desnecessários. Na Figura 16 temos a parte desabitada do bairro com alguns pontos de coleta. Outra limitação observada foi que, caso um ponto não esteja exatamente na mesma coordenada de outro ele não será excluído, assim, na Figura 13 os pontos 1 e 10 estão no mesmo local, mas um deles não foi excluído pois não apresentavam a mesma latitude e longitude.

Os resultados apresentados mostraram que o processo computacional consegue extrair os dados da plataforma colaborativa, tratá-los e montar um grafo do local que posteriormente pode ser utilizado para realizar o cálculo de uma rota de acordo com os parâmetros a serem passados a um algoritmo como o NSGA-II. Além disso, também foi possível exportar os resultados de forma a serem vistos em um mapa geográfico, porém, os mesmos apresentaram limitações devido a falta de informações da base de dados. Após a visualização dos resultados foi possível observar que pontos importantes como a densidade demográfica do local poderiam fazer com que os resultados fossem diferentes e evitasse a marcação de pontos onde não existem residências.

6 Conclusão

Neste trabalho, buscou-se criar um grafo de um determinado local através de uma base de dados colaborativa e posteriormente utilizar o grafo. Para isso, foi necessário extrair e tratar os dados do OpenStreetMap, criar um grafo através dos dados obtidos, marcar pontos e criar um modelo matemático em que uma das restrições é não repetir os pontos marcados durante a rota. Este modelo matemático parece adequado para que possa ser utilizado por algoritmos de otimização que necessita do mapeamento dos bairros em forma de grafo. Foram encontrados alguns problemas em não repetir os lugares já visitados, uma vez que alguns locais possuem apenas uma via para entrada e saída, como é o caso da região do bairro Alphaville de cima presente na Figura 17. Casos como este se repetem em diversos locais e esta é uma limitação atual do processo computacional.

Um entrave enfrentado na pesquisa foi padronizar todos os dados das diferentes plataformas para que eles se comunicassem. Foi utilizado o OpenStreetMap e o Google Earth que forneceram a base de dados e a visualização dos resultados, respectiamente. Apesar de usarem o mesmo tipo de arquivo base eles possuem diferenças que foram necessárias serem tratadas para que as duas plataformas fossem integradas ao modelo matemático proposto. Um ponto de melhoria que foi detectado em um dos casos de testes é que a densidade demográfica do local é um dado importante e evitaria que locais em que não possuem residências fossem levados em consideração, podendo fazer com que tanto os fatores computacionais como fatores de qualidade dos resultados fossem afetados positivamente.

Conclui-se, então, que o processo computacional descrito neste trabalho conseguiu realizar os objetivos propostos e foi capaz de gerar o grafo do mapeamento de bairros e representação de rotas através. Os processos para gerar os grafos utilizaram a base de dados colaborativa do OpenStreetMap e eles também foram capazes exportar as estruturas de forma que podem ser exibidos graficamente em plataformas como o Google Earth. Os resultados foram satisfatórios, uma vez que os casos de teste apresentaram poucas limitações que foram exploradas na seção anterior, mas conseguiram mostrar que os processos foram feitos e puderam ser testados de forma eficiente.

Referências

AHUJA, R. K. et al. Faster algorithms for the shortest path problem. *Journal of the ACM (JACM)*, ACM New York, NY, USA, v. 37, n. 2, p. 213–223, 1990. Citado na página 20.

ATZINGEN, J. V. et al. Análise comparativa de algoritmos eficientes para o problema de caminho mínimo. *Universidade de São Paulo (USP). São Paulo. Escola Politécnica*, 2012. Citado na página 20.

BRASILEIRO, L. A.; LACERDA, M. G. Análise do uso de sig no roteamento dos veículos de coleta de resíduos sólidos domiciliares. *Engenharia Sanitária e Ambiental*, SciELO Brasil, v. 13, n. 4, p. 356–360, 2008. Citado na página 11.

CIDADES e Estados. 2020. <https://www.ibge.gov.br/cidades-e-estados/mg/timoteo.html>. Acessado em 20/11/2020. Citado na página 30.

CLARKE, G.; WRIGHT, J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, Informs, v. 12, n. 4, p. 568–581, 1964. Citado na página 16.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, IEEE, v. 6, n. 2, p. 182–197, 2002. Citado nas páginas 17 e 21.

FRITSCHER, E. Propriedades espectrais de um grafo. 2011. Citado na página 20.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, IEEE, v. 4, n. 2, p. 100–107, 1968. Citado na página 20.

HEINEN, M. R.; OSÓRIO, F. S. Algoritmos genéticos aplicados ao problema de roteamento de veículos. *HÍFEN*, v. 30, n. 58, 2006. Citado nas páginas 12 e 16.

LOURENÇO, J. C. *Gestão Dos Resíduos Sólidos Urbanos*. [S.l.]: Clube de Autores, 2019. Citado na página 11.

MENDES, R. S.; MIRANDA, D. S.; WANNER, E. F. Abordagem multiobjetivo para o problema de roteamento de veículos com transporte reativo a demanda. *dissertação de mestrado, Centro Federal de Educação Tecnológica de Minas Gerais*, 2016. Citado na página 16.

MOLE, R.; JAMESON, S. A sequential route-building algorithm employing a generalised savings criterion. *Journal of the Operational Research Society*, Taylor & Francis, v. 27, n. 2, p. 503–511, 1976. Citado na página 16.

OLAZAR, M. R. R. *Algoritmos Evolucionários Multiobjetivo para alinhamento múltiplo de sequências biológicas*. 2007. Tese (Doutorado) — Dissertação de mestrado, COPPE/UFRJ, 2007. Citado nas páginas 9, 20, 21 e 22.

PECLY, P. H. D.; MELLO, J. A teoria dos grafos na análise do fluxograma do curso de engenharia de produção da uff. *Relatórios de Pesquisa em Engenharia de Produção*, v. 13, p. 20–33, 2013. Citado na página 19.

PESQUISA Nacional de Saneamento Básico. 2008. <https://cidades.ibge.gov.br/brasil/pesquisa/30/84366?ano=2008/>. Acessado em 18/09/2020. Citado na página 11.

PRASETYA, D. A. et al. Resolving the shortest path problem using the haversine algorithm. *J. Crit. Rev*, v. 7, n. 1, p. 62–64, 2020. Citado nas páginas 26 e 27.

SALAS, A. Acerca del algoritmo de dijkstra. *arXiv preprint arXiv:0810.0075*, 2008. Citado na página 20.

SANTOS, F. A. Otimização multi-objetivo aplicada à alocação dinâmica de rotas em redes de telecomunicações. Universidade Federal de Minas Gerais, 2009. Citado na página 11.

SANTOS, S. S. Uma abordagem multiobjetivo para o problema de roteamento de transporte público. Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG, 2017. Citado na página 20.

SHERAFAT, H. Sistema construtor de circuitos e sua aplicação na roteirização de coleta de lixo domiciliar. *Revista GEINTEC-Gestão, Inovação e Tecnologias*, v. 3, n. 5, p. 329–347, 2013. Citado nas páginas 11, 12 e 14.

SIMAS, E. P.; GÓMEZ, A. T. Uma solução para o problema de roteamento de veículos através da pesquisa tabu. *Monografia de Graduação-Universidade do Vale do Rio dos Sinos. São Leopoldo, RS*, 2004. Citado na página 15.

TUSSA'DDIA, N.; HARLINDA, H.; MUDE, M. A. Aplikasi pencarian barbershop menggunakan metode harversine formula untuk menentukan jarak terdekat. *Buletin Sistem Informasi dan Teknologi Islam*, v. 2, n. 1, p. 56–63, 2021. Citado na página 26.

XAVIER, R. S. et al. Heuristica para modelagem e minimização do consumo de combustível para rotas de coleta de lixo. *Bento Gonçalves*, p. 12, 2010. Citado nas páginas 14 e 15.

XU, J.; KELLY, J. P. A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation science*, INFORMS, v. 30, n. 4, p. 379–393, 1996. Citado na página 15.