MEC-SETEC
INSTITUTO FEDERAL MINAS GERAIS - *Campus* Formiga
Curso de Ciência da Computação

UTILIZAÇÃO DA HEURÍSTICA NSGA-II NA OTIMIZAÇÃO DAS ROTAS DE COLETA DE LIXO, APLICADA ÀS CIDADES MINEIRAS DE FORMIGA E LAGOA DA PRATA

Lucas Mateus Menezes Silva

Orientador: Everthon Valadão dos Santos

Formiga - MG

2023

LUCAS MATEUS MENEZES SILVA

UTILIZAÇÃO DA HEURÍSTICA NSGA-II NA OTIMIZAÇÃO DAS ROTAS DE COLETA DE LIXO, APLICADA ÀS CIDADES MINEIRAS DE FORMIGA E LAGOA DA PRATA

Monografia do trabalho de conclusão de curso apresentado ao Instituto Federal Minas Gerais - Campus Formiga, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Everthon Valadão dos Santos

Formiga - MG

2023

Silva, Lucas Mateus Menezes
S586u Utilização da Heurística NSGA-II na Otimização das Rotas de Coleta de Lixo, Aplicada às Cidades Mineiras de Formiga e Lagoa da Prata / Lucas Mateus Menezes Silva - Formiga : IFMG, 2023.
70p. : il.

Orientador: Prof. MSc. Everthon Valadão dos Santos
Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Coleta de Lixo ou Resíduos. 2. Otimização de Rotas. 3. Roteamento de Veículos Capacitados (PRVC). 4. Carteiro Chinês com Vento (PCCV). 5. Algoritmo NSGA-II. I. dos Santos, Everthon Valadão. II. Título.

CDD 004

Ficha catalográfica elaborada pela Bibliotecária Msc. Simoni Júlia da Silveira

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
Campus Formiga
Diretoria de Ensino
Docência Área Acadêmica de Computação
Rua São Luiz Gonzaga, s/n - Bairro São Luiz - CEP 35570-000 - Formiga - MG
- www.ifmg.edu.br

LUCAS MATEUS MENEZES SILVA

Utilização da heurística NSGA-II na otimização das rotas de coleta de lixo, aplicada às cidades mineiras de Formiga e Lagoa da Prata

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Minas Gerais - Campus Formiga, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

APROVADO em: 16 de novembro de 2023.

BANCA EXAMINADORA

Prof.º Everthon Valadão dos Santos (orientador, IFMG)

Prof.º Mário Luiz Rodrigues Oliveira (IFMG)

Prof.º Marlon Jesus Lizarazo Urbina (IFMG)

Documento assinado eletronicamente por **Everthon Valadao dos Santos**, **Professor**, em 16/11/2023, às 10:14, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Mario Luiz Rodrigues Oliveira**, **Professor**, em 16/11/2023, às 10:15, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Marlon Jesus Lizarazo Urbina**, **Professor Substituto**, em 20/11/2023, às 10:41, conforme Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade do documento pode ser conferida no site https://sei.ifmg.edu.br/consultadocs informando o código verificador **1729531** e o código CRC **483065A3**.

Agradecimentos

Gostaria de agradecer a todos que estiveram presentes direta ou indiretamente em minha vida durante esse tempo no ensino superior: amigos, professores e familiares que me ajudaram e me incentivaram a realizar o curso.

Agradecer principalmente aos meus pais, que são minhas maiores inspirações na vida e meus maiores incentivadores, a instituição, que pôde me ofertar uma educação pública de qualidade, e que também me proporcionou diversas oportunidades ao longo dessa caminhada, e ao meu orientador Everthon que me guiou com paciência durante a execução deste trabalho.

"Não basta saber ler que 'Eva viu a uva'. É preciso compreender qual a posição que Eva ocupa no seu contexto social, quem trabalha para produzir a uva e quem lucra com esse trabalho." (Paulo Freire)

Resumo

O Brasil é um dos grandes produtores de lixo do mundo. De acordo com um estudo feito pelo WWF (World Wide Fund for Nature), o Brasil se encontra na 4ª posição como o maior produtor de lixo plástico em todo o mundo (WWF, 2019). Com tanto lixo sendo produzido no país, torna-se difícil para as cidades coletarem todo o lixo e ainda o façam de uma forma eficiente. Neste trabalho é proposto um método para otimizar rotas de veículos coletores de lixo, através da meta-heurística NSGA-II, tendo sido desenvolvido um protótipo de *software* que gera rotas de coleta otimizadas, para uma determinada cidade ou região. Nos experimentos, foram utilizados como cenários as cidades de Formiga/MG e Lagoa da Prata/MG, as soluções geradas conseguiram eficazmente gerar rotas que passam por todos os pontos de coleta definidos (esquinas entre as ruas) bem como coletar todo o montante de lixo simulado respeitando as restrições na quantidade de caminhões de lixo. Por utilizar uma heurística multiobjetivo, a aplicação gera um conjunto de soluções possíveis, de forma que o responsável pela coleta de lixo daquela localidade escolha a solução que melhor atenda suas demandas, ex.: limitação na quantidade de caminhões, periodicidade desejada da coleta, maior economia de combustível, dentre outras. Com o problema devidamente modelado para a meta-heurística NSGA-II, foi possível otimizar todas as funções-objetivo definidas com um tempo de processamento em torno de quatro horas. Para tal, foram experimentadas diferentes estratégias de mutação e de *crossover*, devidamente documentadas. Dentre as principais contribuições realizadas por esse trabalho estão: confecção de uma aplicação que gera configurações de rotas de coleta de lixo para uma cidade ou região; modelagem para simulação do lixo diariamente gerado em cada rua de uma cidade, em média; modelagem da limitação na capacidade de cada caminhão transportar lixo (Problema de Roteamento de Veículos Capacitados), demandando mais de uma viagem; modelagem de função-objetivo para que as rotas geradas para os caminhões evitem subir e descer morros desnecessariamente (Problema do Carteiro Chinês Com Vento); proposta de uso de memorização da clusterização para agilizar o tempo de execução do algoritmo. O software elaborado exporta um arquivo KML contendo as diferentes rotas geradas, de maneira que possam ser carregadas e visualizadas no Google Earth. Todo o código gerado foi disponibilizado para acesso público em um repositório no GitHub.

Palavras-chave: coleta de lixo ou resíduos, otimização de rotas, Roteamento de Veículos Capacitados (PRVC) Carteiro Chinês Com Vento (PCCV), algoritmo NSGA-II.

Abstract

Brazil is one of the largest waste producers in the world. According to a study carried out by WWF (World Wide Fund for Nature), Brazil is in 4th position as the largest producer of plastic waste in the world (WWF, 2019). With so much trash being produced in the country, it becomes difficult for cities to collect all the waste and still do so efficiently. In this work, a method is proposed to optimize routes for waste collection vehicles, using the NSGA-II meta-heuristic, and a software prototype that generates optimized collection routes for a given city or region. In the experiments, the cities of Formiga/MG and Lagoa da Prata/MG were used as scenarios, the solutions generated were able to effectively generate routes that pass through all defined collection points (street corners) as well as collect the entire amount of simulated waste respecting the restrictions on the number of garbage trucks. By using a multi-objective heuristic, the application generates a set of possible solutions, so that the person responsible for waste collection in that location chooses the solution that best meets their demands, e.g.: limitation on the number of trucks, desired garbage collection frequency, greater fuel economy, among others. With the problem properly modeled for the NSGA-II meta-heuristic, it was possible to optimize all defined objective functions with a processing time of around four hours. To this end, different mutation and crossover strategies were tried, duly documented. Among the main contributions made by this work are: creation of an application that generates waste collection route configurations for a city or region; modeling to simulate the daily garbage generated on each street in a city, on average; modeling the limitation on the capacity of each truck to transport waste (Capacitated Vehicle Routing Problem), requiring more than one trip; objective function modeling so that the routes generated for trucks avoid going up and down hills unnecessarily (Windy Postman Problem); proposed use of clustering memoization to speed up the algorithm's execution time. The developed software exports a KML file containing the different routes generated, so that they can be loaded and viewed on Google Earth. All generated code was made available for public access in a repository on GitHub.

Keywords: waste collection, route optimization, Capacitated Vehicle Routing Problem (CVRP), Windy Postman Problem (WPP), NSGA-II algorithm.

Lista de ilustrações

Figura 1 – Ilustração do problema de roteamento de veículos (PRV) 19
Figura 2 – Dominância de pontos e fronteira de Pareto 24
Figura 3 – Exemplificação do *Crowding Distance* 25
Figura 4 – Funcionamento básico do algoritmo NSGA-II 25
Figura 5 – Exemplificação do hypervolume 26
Figura 6 – Captura inicial dos dados da cidade de Formiga/MG 36
Figura 7 – Verificação dos dados filtrados e mapeados 38
Figura 8 – Detalhe dos dados mapeados 38
Figura 9 – Primeiro grafo montado pelo algoritmo 39
Figura 10 – Grafo simplificado da cidade 40
Figura 11 – Comparação entre os grafos gerados 40
Figura 12 – Exemplo de clusterização 42
Figura 13 – Representação de um indivíduo 43
Figura 14 – Exemplo de cruzamento entre dois indivíduos 44
Figura 15 – Exemplo de mutação de um indivíduo 45
Figura 16 – Exemplo de veículo em rua íngreme 47
Figura 17 – Evolução do algoritmo ao longo de 500 gerações 52
Figura 18 – Rotas com todas as métricas minimizadas - Formiga-MG 55
Figura 19 – Detalhamento de rota - Formiga-MG 56
Figura 20 – Rotas com foco na minimização no número de caminhões - Formiga-MG 57
Figura 21 – Rotas com foco na minimização no tempo de coleta - Formiga-MG . . 58
Figura 22 – Evolução das fronteiras de Paretto - Formiga-MG 59
Figura 23 – Rotas com todas as métricas minimizadas - Lagoa da Prata-MG 61
Figura 24 – Rotas com foco na minimização do nº de caminhões - Lagoa da Prata . 62
Figura 25 – Rotas com foco na minimização no tempo de coleta - Lagoa da Prata-MG 63
Figura 26 – Evolução das fronteiras de Paretto - Lagoa da Prata-MG 63

Lista de tabelas

Tabela 1 – Níveis dos fatores no primeiro projeto fatorial 50
Tabela 2 – Níveis dos fatores no segundo projeto fatorial 51
Tabela 3 – Níveis dos fatores no terceiro projeto fatorial 51

Lista de quadros

Quadro 1 – Consulta realizada na ferramenta Overpass-turbo 35
Quadro 2 – Solução minimizando as métricas de forma balanceada 55
Quadro 3 – Solução minimizando a métrica do número de caminhões 56
Quadro 4 – Solução minimizando a métrica do tempo de coleta. 58
Quadro 5 – Solução minimizando todas as métricas balanceadamente 60
Quadro 6 – Solução minimizando a métrica do número de caminhões 61
Quadro 7 – Solução minimizando a métrica do tempo de coleta 62

Lista de abreviaturas e siglas

API	*Aplication Programming Interface*
CVRP	*Capacitated Vehicle Routing Problem*
KML	*Keyhole Markup Language*
PCC	Problema do Carteiro Chinês
PCCV	Problema do Carteiro Chinês com Vento
PCCD	Problema do Carteiro Chinês Direcionado
NSGA-II	*Non-dominated Sorting Genetic Algorithm*
OSM	*OpenStreetMap*
PRV	Problema do Roteamento de Veículos
PRVC	Problema de Roteamento de Veículos Capacitado
RAM	*Random Access Memory*
VRP	*Vehicle Routing Problem*
WPP	*Windy Postman Problem*
XML	*Extensible Markup Language*

Sumário

1 INTRODUÇÃO 16
1.1 Justificativa 16
1.2 Objetivo Geral 17
1.3 Objetivos Específicos 17

2 FUNDAMENTAÇÃO TEÓRICA 18
2.1 Coleta de Lixo e Resíduos em Áreas Urbanas 18
2.2 Problema de Roteamento de Veículos 19
2.2.1 Problema de Roteamento de Veículos Capacitados 19
2.3 Problema do Carteiro Chinês 19
2.3.1 Problema do Carteiro Chinês Com Vento 20
2.3.2 Problema do Carteiro Chinês Direcionado 20
2.3.3 Grafos eulerianos 21
2.4 Clusterização (agrupamento) de dados 21
2.5 Algoritmos Genéticos 21
2.5.1 NSGA-II: uma heurística multi-objetivo 22
2.5.2 *Hipervolume* 26
2.6 Trabalhos relacionados 27

3 MATERIAIS E MÉTODO 28
3.1 *Publish or Perish* 28
3.2 OpenStreetMap 28
3.3 Overpass-turbo 29
3.4 Open Topo Data 29
3.5 Google Earth 29
3.6 Python 30
3.6.1 Bibliotecas Python 30
3.7 Git e GitHub 31
3.8 Hardware 31
3.9 Método 31
3.10 Estudo e obtenção dos dados do mapa 31
3.11 Estudo e adaptação da meta-heurística NSGA-II 32
3.11.1 Representação de um indivíduo (solução) 32
3.12 Implementação do cruzamento entre indivíduos 32
3.12.1 Implementação da mutação entre indivíduos 33
3.12.2 Implementação das funções objetivo 33

3.13 Calibragem dos parâmetros via Projeto Fatorial 2^k . . . 33
3.14 Execução dos experimentos . . . 33

4 DESENVOLVIMENTO . . . 35
4.1 Obtenção do mapa da cidade . . . 35
4.1.1 Estrutura do arquivo XML . . . 36
4.1.2 Manipulação do arquivo XML . . . 37
4.1.3 Visualização do mapa obtido para a cidade . . . 37
4.2 Grafo da cidade . . . 38
4.2.1 Modelagem da quantidade de lixo gerado e simplificação do grafo . . . 39
4.2.2 Altitude dos pontos de coleta . . . 41
4.2.3 Clusterização dos pontos do grafo utilizando o algoritmo *k-means* . . . 41
4.2.4 Montagem do *cache* de agrupamentos do grafo . . . 42
4.3 Implementação do NSGA-II . . . 43
4.3.1 Abstração do indivíduo . . . 43
4.3.2 Cruzamento entre indivíduos . . . 44
4.3.3 Mutação . . . 44
4.3.4 Quantidade de agrupamentos (*clusters*) para a cidade . . . 45
4.3.5 Função objetivo: Minimização do tempo de trabalho dos caminhões . . . 46
4.3.6 Função objetivo: Minimização da quantidade de caminhões . . . 46
4.3.7 Função objetivo: Minimização da variação de altitude . . . 47
4.3.8 Avaliação do indivíduo . . . 48
4.3.9 Normalização das métricas do indivíduo . . . 48
4.4 Testes preliminares e ajustes para validação do algoritmo . . . 49
4.5 Calibração dos parâmetros por meio do Projeto Fatorial 2^k . . . 49
4.5.1 Primeiro experimento do projeto fatorial 2^k . . . 50
4.5.2 Segundo experimento do projeto fatorial 2^k . . . 50
4.5.3 Terceiro experimento do projeto fatorial 2^k . . . 51
4.5.4 Definição dos parâmetros para o NSGA-II . . . 52

5 RESULTADOS E ANÁLISE . . . 54
5.1 Resultados com o mapa de Formiga-MG . . . 54
5.1.1 Solução minimizando todas as métricas de forma balanceada . . . 54
5.1.2 Solução minimizando a métrica do número de caminhões . . . 56
5.1.3 Solução minimizando a métrica do tempo de coleta . . . 57
5.1.4 Evolução das fronteiras de Paretto . . . 59
5.2 Resultados com o mapa de Lagoa da Prata-MG . . . 59
5.2.1 Solução minimizando todas as métricas de forma balanceada . . . 60
5.2.2 Solução minimizando a métrica do número de caminhões . . . 61
5.2.3 Solução minimizando a métrica do tempo de coleta . . . 62

5.2.4 Evolução das fronteiras de Paretto 63

6 CONSIDERAÇÕES FINAIS 64
6.1 Principais contribuições 65
6.2 Trabalhos futuros 66

REFERÊNCIAS 68

1 Introdução

O lixo vem, com o passar do tempo, se tornando um problema alarmante. Com o crescimento da população mundial, cresceu também a produção de lixo. Como exemplo, a população mundial entre 1991 e 2000 cresceu 15,6%, enquanto a produção de lixo no mesmo período cresceu 49% (WALDEMAN, 2011). Devido a essa enorme produção de lixo, surgem diversos problemas causados por esses resíduos, um destes é: como coletar tanto lixo de maneira eficiente? A coleta se feita de forma incorreta e ineficiente pode causar acúmulo de lixo em determinados locais e, por consequência, alagamentos em caso de chuvas e/ou proliferação de doenças e pragas como baratas e ratos, que comprometem a saúde pública (SANTO, 2013).

Além destes problemas, o recolhimento do lixo pode demandar muitos recursos, principalmente em grandes cidades, que possuem uma maior produção de lixo e uma área maior a ser coberta pelos veículos, fazendo com que os mesmos consumam mais combustível e tempo para a finalização da coleta. Um exemplo da quantidade de gastos que a coleta pode demandar é visto na cidade de São Paulo. A capital paulista gastou, em 2012, cerca de 56 milhões de reais mensais com a coleta de resíduos (G1, 2012). Caso uma otimização nas rotas dos veículos coletores fosse realizada, a cidade poderia economizar uma parte destes custos, que poderiam ser investidos em outras áreas diversas para benefício da população.

1.1 Justificativa

Com as dimensões do problema em mente, torna-se necessário buscar meios para a resolução deste problema, ou seja, investigar maneiras de reduzir o custo e tempo para o lixo ser recolhido. Este trabalho tem como foco essa redução de custos, por meio da otimização da coleta, para os veículos conseguirem efetuar de forma mais eficiente o recolhimento do lixo de uma determinada cidade ou região.

Como citado anteriormente, a melhoria das rotas causaria diversos benefícios, como: economia de recursos públicos na manutenção e abastecimento dos veículos, diminuição do tempo que o lixo ficaria exposto no local de coleta, diminuição da chance de proliferação de pragas e doenças e menores ocorrências de alagamentos provocados por lixo obstruindo bueiros e canais (SANTO, 2013).

1.2 Objetivo Geral

O objetivo geral deste trabalho é desenvolver uma solução automatizada para gerar rotas de coleta de lixo para uma cidade desejada. Os trajetos gerados visarão otimizar os custos relacionados (Ex.: quantidade de caminhões necessários, distância percorrida, etc.).

1.3 Objetivos Específicos

- Mapear os pontos de coleta e possíveis rotas para os veículos coletores (*dataset*);
- Obter o *dataset* contendo o mapa viário e topográfico de algumas cidades como, por exemplo, Formiga/MG e Lagoa da Prata/MG;
- Modelar a quantidade média de lixo diariamente gerado na cidade utilizada como cenário, considerando a quantidade média de lixo produzida por família, o comprimento de cada rua e a quantidade de casas por rua;
- Abstrair uma forma de representação dos pontos de coleta, das rotas e do problema, para a solução poder ser implementada;
- Implementar e validar a heurística adotada, modelando o problema para encaixar-se na metáfora utilizada por ela utilizada;
- Aplicar a heurística no *dataset* e (re)calibrá-la conforme a qualidade dos resultados obtidos (otimização obtida) e o tempo de processamento necessário para tal;
- Analisar os resultados dos experimentos, destacando as melhorias obtidas e identificando possíveis trabalhos futuros.

2 Fundamentação Teórica

Ao realizar a revisão bibliográfica dos principais assuntos deste trabalho, foram encontradas diversas pesquisas sobre otimização de rotas e montagem de trajetos. Neste capítulo são apresentados os principais conceitos e os trabalhos relacionados.

2.1 Coleta de Lixo e Resíduos em Áreas Urbanas

A coleta de lixo e resíduos em áreas urbanas é um processo fundamental na promoção da saúde pública, na preservação do meio ambiente e na manutenção da qualidade de vida nas cidades. Segundo Detofeno e Steiner (2010):

> Com o crescimento populacional e o consequente aumento do consumo de bens alimentícios per capita, e de acordo com o Censo Demográfico do Instituto Brasileiro de Geografia e Estatística (IBGE, 2000), 81% da população brasileira está concentrada em áreas urbanas causando um volume crescente de resíduos produzidos, demonstrando assim a importância da gestão de resíduos nas áreas urbanas.

A coleta de lixo envolve a coleta, transporte e disposição adequada dos resíduos sólidos gerados pelas atividades cotidianas das comunidades urbanas. Segundo Detofeno e Steiner (2010):

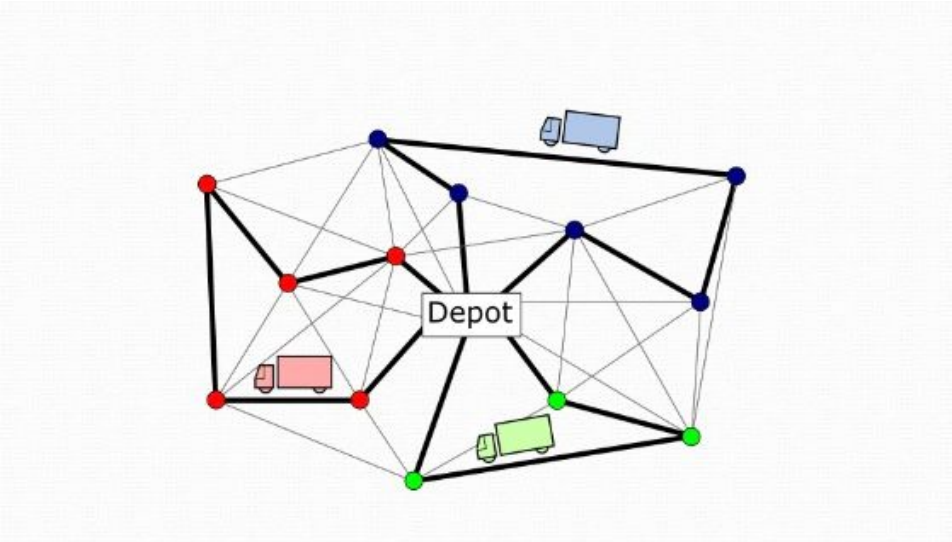
> Em geral, numa cidade existem três tipos principais de recolha de resíduos: a dos resíduos urbanos, a dos resíduos hospitalares e a recolha seletiva. A coleta de lixo urbano [...] consiste na coleta e transporte de resíduos domiciliares e urbanos produzidos em residências, condomínios, instituições públicas e estabelecimentos comerciais, industriais e de serviços.

A coleta de lixo e resíduos é geralmente realizada por equipes de trabalhadores que percorrem rotas pré-determinadas em caminhões de coleta, coletando sacos de lixo deixados pelos residentes ao longo dessas rotas. Os resíduos coletados incluem materiais como resíduos domésticos, resíduos de construção e demolição, resíduos comerciais e industriais, entre outros. Após a coleta, os resíduos são transportados para instalações de tratamento, como aterros sanitários, usinas de reciclagem, ou incineradores, dependendo das práticas locais de gestão de resíduos.

2.2 Problema de Roteamento de Veículos

O problema de roteamento de veículos, (PRV) ou também *Vehicle Routing Problem* (VRP), é um problema primeiramente descrito por Dantzig e Ramser (1959). Segundo Laporte (1992), O PRV se descreve como: "(...) o problema de projetar uma entrega ideal ou rotas de coleta de um ou vários depósitos para uma série de cidades ou clientes geograficamente dispersos, sujeito a restrições laterais". A Figura 1 ilustra o problema.

Figura 1 – Ilustração do problema de roteamento de veículos (PRV)



Fonte: (ROMAIS, 2012)

2.2.1 Problema de Roteamento de Veículos Capacitados

O PRV possui diversas variantes, e uma delas é uma versão capacitada, em que os veículos possuem limitações quanto a sua capacidade de armazenamento. O Problema de Roteamento de Veículos Capacitados (PRVC), é definido por Laporte (1992) como uma das possíveis condições laterais do problema, em que um peso não negativo é atribuído em cada ponto de coleta e a soma desses pesos na rota do veículo não deve ultrapassar a sua própria capacidade de carga.

2.3 Problema do Carteiro Chinês

No problema clássico de otimização em teoria dos grafos conhecido como Carteiro Chinês (PCC), o objetivo é encontrar o caminho mais curto (ou circuito fechado) que visita cada aresta de um grafo pelo menos uma vez, enquanto minimiza a distância total percorrida pelo carteiro (EISELT; GENDREAU; LAPORTE, 1995).

2.3.1 Problema do Carteiro Chinês Com Vento

O Problema do Carteiro Chinês Com Vento (PCCV ou *Windy Postman*) é uma variante do PCC, cuja peculiaridade é que algumas arestas têm um custo adicional devido ao vento, tornando o caminho mais longo em relação às outras arestas ou, então, tornando o caminho que utilize uma aresta em um sentido mais custoso que o caminho que utilize a aresta correspondente no sentido inverso (grafo direcionado).

No PCCV o objetivo é encontrar o caminho que visita todas as arestas, incluindo as afetadas pelo vento, de modo a minimizar a distância total percorrida, considerando esses custos adicionais de vento. Segundo Eiselt, Gendreau e Laporte (1995):

> Um problema intimamente relacionado ao CPP misto é o Problema do Carteiro Chinês Com Vento (PCCV) introduzido pela primeira vez por Minieka (1979). Aqui, G é um grafo não direcionado, embora o custo de atravessar uma aresta dependa da direção do deslocamento. O PCCV consiste em determinar uma travessia de menor custo de todas as arestas de G. Brucker (1981) e Guan (1984b) mostraram que o PCCV é NP-difícil, mas o problema é solucionável em tempo polinomial se G for Euleriano (Win 1989).

Neste trabalho, consideramos que um caminhão de lixo subir uma rua em aclive teria um custo maior do que o caminhão descer a rua em declive. Assim, as soluções geradas indiretamente irão reduzir o consumo de combustível, bem como o tempo gasto ao percorrer as rotas.

2.3.2 Problema do Carteiro Chinês Direcionado

Há ainda uma variante do Problema do Carteiro Chinês chamada Problema do Carteiro Chinês Direcionado (PCCD). Eiselt, Gendreau e Laporte (1995) observam que ao contrário do caso não direcionado (PCC), o problema PCCD pode não ter solução mesmo se o grafo G for conexo. Para existir uma solução, o grafo deve ser fortemente conectado, ou seja, deve haver um caminho direcionado entre cada par de nós.

Neste trabalho não consideramos situações em que as ruas sejam de mão única, tendo sido consideradas todas as ruas como sendo de mão dupla. O motivo para tal é que, até o momento, não conseguimos obter dados atualizados e confiáveis sobre os sentidos permitidos e proibidos (mão e contramão) da malha viária das cidades utilizadas como cenário dos experimentos.

2.3.3 Grafos eulerianos

Um grafo G(V, A) é uma estrutura definida por dois conjuntos: o conjunto V que define os vértices que o grafo contém, e o A que define as arestas, que realizam a ligação entre os vértices. De acordo com Ore e Wilson (1990), um grafo euleriano é aquele no qual o seu conjunto de vértices e arestas, permite que se forme um circuito euleriano no mesmo. Um circuito euleriano, por sua vez, pode ser definido como um circuito que se passa por todas as arestas de um grafo sem que nenhuma delas seja repetida durante o trajeto. O circuito euleriano foi primeiramente definido por Leonhard Euler, enquanto estudava sobre o problema das pontes de Königsberg (atual Kaliningrado).

2.4 Clusterização (agrupamento) de dados

De acordo com Oliveira (2008), a clusterização consiste em uma forma de organização de dados, retirando-os de grandes conjuntos e separando-os em grupos de acordo com características semelhantes entre esses dados.

Segundo Jain (2008), um *cluster* é composto por vários objetos semelhantes agrupados. E, apesar de milhares de algoritmos de clusterização publicados, o usuário ainda enfrenta um dilema quanto à escolha do algoritmo, métrica de distância, normalização de dados, número de *clusters* e critérios de validação. O popular algoritmo de agrupamento K-means, descrito por Hartigan e Wong (1979) visa dividir M pontos em N dimensões em K *clusters* para que a soma dos quadrados no *cluster* seja minimizada. O algoritmo requer como entrada uma matriz de M pontos em N dimensões e uma matriz de K centros de *cluster* iniciais em N dimensões. O procedimento geral é procurar uma partição K com soma de quadrados localmente ótima no *cluster*, movendo pontos de um *cluster* para outro (HARTIGAN; WONG, 1979).

Neste trabalho, o agrupamento (clusterização) via K-means foi aplicado no grafo que representa os pontos de coleta de lixo em uma cidade, classificando os vértices mais próximos em grupos, para reduzir o espaço de busca e obter uma maior velocidade no processamento de rotas.

2.5 Algoritmos Genéticos

Os algoritmos genéticos são um tipo de algoritmo com inspiração na teoria proposta por Charles Darwin, que explica o processo de evolução das espécies, em que o mais apto possui mais chances de sobreviver e passar seus genes adiante. As características que diferem o algoritmo genético das demais meta-heurísticas são: utilização de uma população de soluções que guia o processo de busca de uma solução; a combinação de dois indivíduos dessa população para ser gerado um novo indivíduo provavelmente melhor e a conservação

da diversidade na população para que a mesma não caia sobre um ótimo local (WHITLEY, 2019).

Segundo Scaburi (2020), um algoritmo genético possui os seguintes componentes básicos: a representação genética utilizada para o problema em questão, população inicial, a função objetivo que avalia os indivíduos, o método de *crossover*, a forma de seleção de gerações, os critérios de parada e as configurações do algoritmo.

No Algoritmo 1 é apresentado um pseudocódigo que define a ideia geral de um algoritmo genético.

Algoritmo 1: Algoritmo Genético

```
início
    Gera uma população inicial P
    Avalia a população gerada
    while Condição de parada do
        Selecionar indivíduos pais
        Realizar Crossover
        if condição de mutação foi satisfeita then
            Realizar mutação do indivíduo
        end
        Avaliar indivíduo resultante do crossover
        Selecionar nova população
    end
fim
```

2.5.1 NSGA-II: uma heurística multi-objetivo

Nondominated Sorting Genetic Algorithm II, mais conhecido como NSGA-II, é um algoritmo evolucionário elitista multi-objetivo descrito por Deb et al. (2002). O NSGA-II se difere do NSGA nos três seguintes pontos: um melhor custo assintótico do algoritmo, pois o NSGA possui uma complexidade computacional de $O(MN^3)$ enquanto a do NSGA-II é de $O(MN^2)$, onde M representa a quantidade de objetivos e N o tamanho da população; uma maior aplicação do elitismo, dado que em seu artigo é apresentado que o elitismo ajuda a manter boas soluções na população do algoritmo, melhorando assim o resultado; remoção do parâmetro de compartilhamento, parâmetro esse utilizado no NSGA para garantir uma maior diversidade na população (DEB et al., 2002).

O pseudocódigo do NSGA-II é apresentado no Algoritmo 2 e seu funcionamento se dá, resumidamente, da seguinte forma: É Inicialmente gerada uma população inicial P_0, que contém um determinado número de indivíduos. A partir de P_0, é gerada outra população Q_0, sendo o resultado da mistura de genes entre os melhores indivíduos de P_0.

A cada iteração t da quantidade de gerações g, a junção entre os indivíduos que

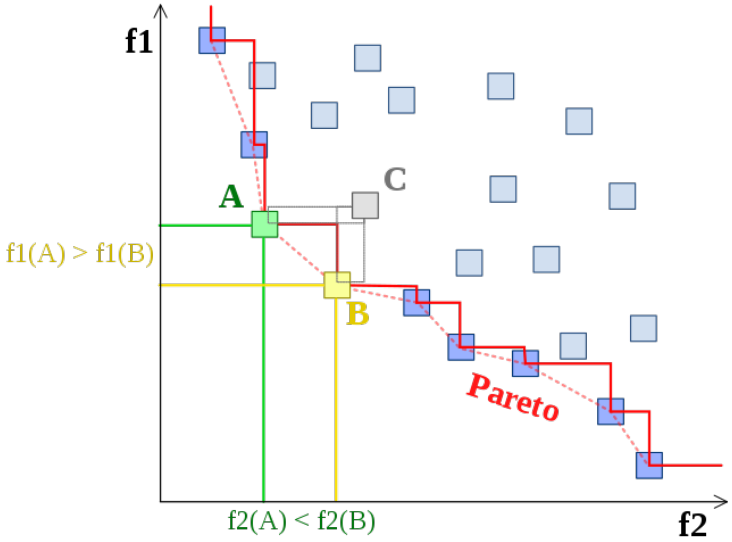
Algoritmo 2: Algoritmo do NSGA-II

```
begin
    P_0 ← novaPopulação(N)
    avaliaPopulação(P_0)
    F ← fastNonDominatedSort(P_0)
    Q_0 ← cruzamento(P_0)
    avaliaPopulação(Q_0)
    for t ← 1 to g do
        R_t ← P_t ∪ Q_t
        F ← fastNonDominatedSort(R_t)
        P_{t+1} ← novaPopulação(0)
        j ← 1
        while |P_{t+1}| + |F_j| ≤ N do
            atribuiCrowdingDistance(F_j)
            P_{t+1} ← P_{t+1} ∪ F_j
            j ← j + 1
        end
        ordenaPorCrowdedComparison(F_j)
        r ← N − P_{t+1}
        P_{t+1} ← P_{t+1} ∪ F_j[1 : r]
        Q_{t+1} ← cruzamento(P_{t+1})
        avaliaPopulação(Q_{t+1})
    end
end
;                                   /* FONTE: Adaptado de (PINTO, 2021) */
```

formam P_t e Q_t forma a população R_t. Após a geração de R_t, todos os seus membros são classificados de maneira elitista por meio do *Fast Non Dominated Sorting* que os divide em várias fronteiras de dominância, onde a primeira fronteira contém os indivíduos que não são dominados por nenhum outro, a segunda os que são dominados por apenas algum outro indivíduo e assim por diante.

Para que isso seja feito, o *Fast Non Dominated Sorting* utiliza de um operador que indica a dominância de um indivíduo A para um C (veja Figura 2), indicando qual dos dois é dominado pelo outro, sendo possível, que ambos pertençam a mesma fronteira (ex.: indivíduos A e B) e não possuam relação de dominância entre si (CâNDIDO, 2019). Na Figura 2, os pontos representam escolhas viáveis em um problema de minimização, onde valores menores devem ser preferidos aos maiores. O ponto C não está na fronteira de Pareto porque é dominado tanto pelo ponto A quanto pelo ponto B. Os pontos A e B não são estritamente dominados por nenhum outro e, portanto, estão na fronteira.

Figura 2 – Dominância de pontos e fronteira de Pareto



Fonte: Wikipedia / Autor: Johann Dréo, 2006

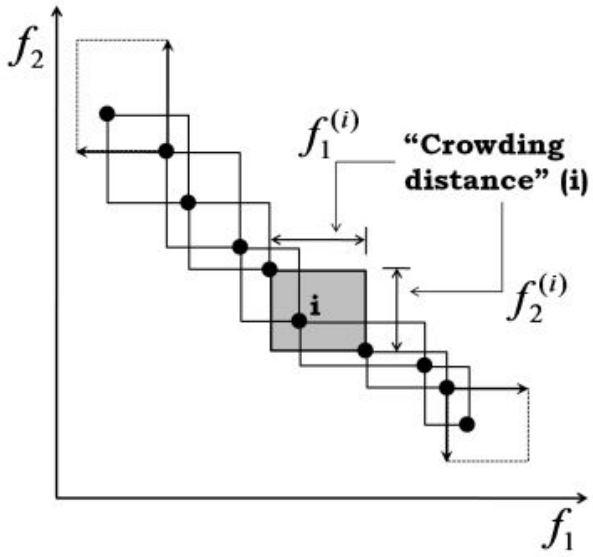
De acordo com Cândido (2019):

> Com o operar de dominância podendo ser descrito pela proposição logica: $\forall j \; \exists k \; (Faj <= Fbj) \; and \; (Fak < Fbk)$, sendo *Faj* a função objetivo *j* da solução *a*. Através dessa operação o algoritmo verifica quem não é dominado por ninguém, estes elementos são os melhores, fazendo parte da fronteira ótima conhecida, com conhecimento da primeira fronteira (melhor conhecida), agora é verificado quem é dominado somente por eles, assim é encontrado a segunda fronteira, e assim por adiante, ou seja, as soluções da n-ésima fronteira são dominadas pelas funções das n-ésima seguintes fronteiras. Através desse mecanismo é classificado as fronteiras.

Após a classificação das soluções em diferentes frentes de Pareto, é calculado para cada fronteira o *Crowding Distance*. Esse método ordena os indivíduos de acordo com sua distribuição na fronteira, priorizando indivíduos que estejam mais espalhados, fazendo com que a próxima população tenha indivíduos com uma maior diversidade em seus genes, como ilustrado na Figura 3. Esse cálculo realizado em cada fronteira é feito por meio das distâncias euclidianas entre as duas soluções vizinhas de cada lado da solução ao longo dos objetivos (BANDYOPADHYAY; CHAKRABORTY; MAULIK, 2015).

O *Crowding Distance* desempenha um papel crucial na manutenção da diversidade na população de soluções, ao calcular a densidade de soluções vizinhas para cada ponto na mesma frente. Funciona medindo a distância multidimensional entre uma solução e suas vizinhas mais próximas em cada objetivo (veja Figura 3), proporcionando uma estimativa de quão "lotada" está a região do espaço de soluções em torno de um ponto específico. Essencialmente, a *Crowding Distance* ajuda a priorizar soluções em regiões menos povoadas

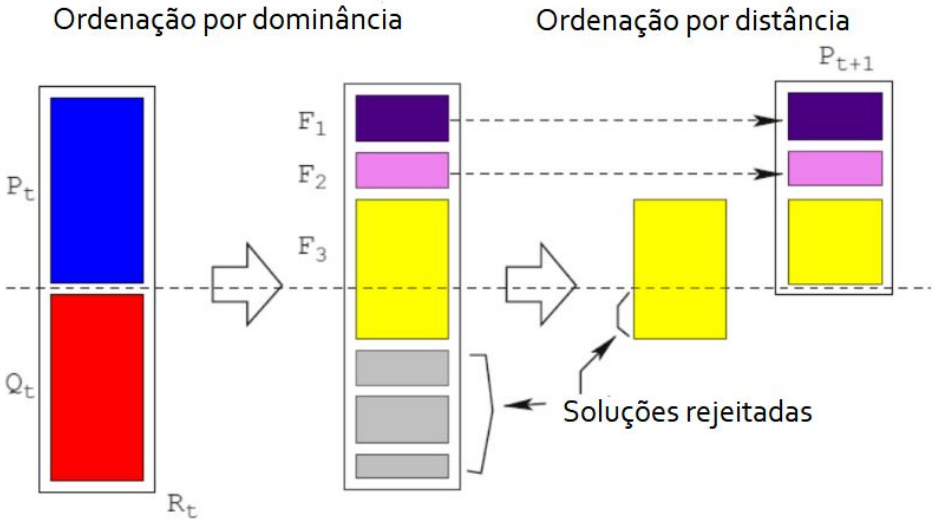
Figura 3 – Exemplificação do *Crowding Distance*



Fonte: Bandyopadhyay, Chakraborty e Maulik (2015)

do espaço de busca, incentivando assim uma distribuição mais uniforme e abrangente de soluções ao longo das frentes de Pareto, o que é crucial para garantir uma boa cobertura do espaço de soluções multiobjetivo.

Figura 4 – Funcionamento básico do algoritmo NSGA-II



Fonte: Adaptado de (PINTO, 2021 apud DEB et al., 2002)

Em suma, a Figura 4 demonstra o fluxo básico do algoritmo. Com todos os elementos devidamente classificados em suas fronteiras, eles são selecionados de acordo com sua ordenação para compor a próxima população, que deve conter a mesma quantidade de membros da população inicial *Pt*, assim o ciclo do NSGA-II é reiniciado e repetido pelo número de gerações definido.

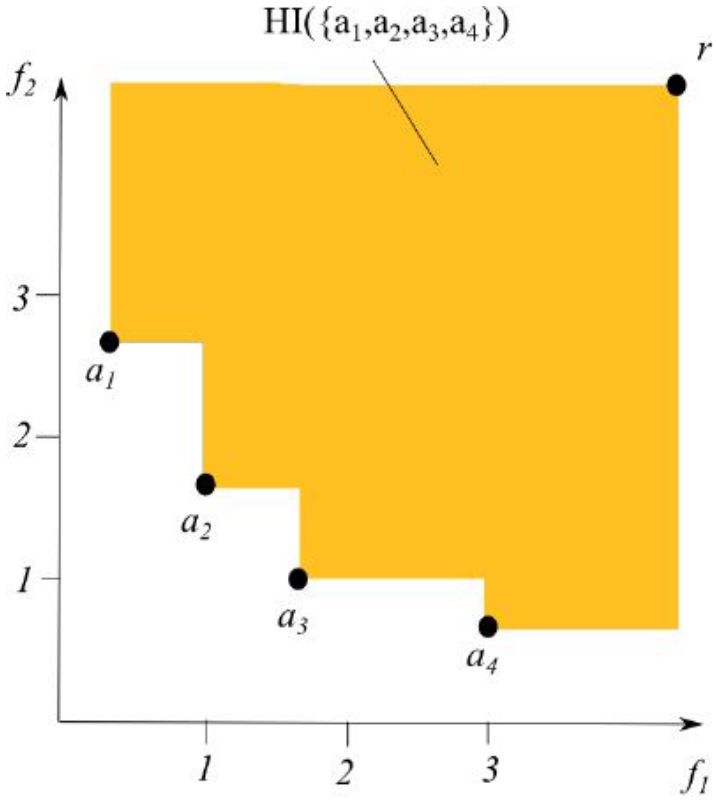
2.5.2 *Hipervolume*

Segundo Pinto (2021):

> Em 1998, (PINTO, 2021 apud ZITZLER; THIELE, 1998) apresentam um estudo comparativo de abordagens evolutivas para otimização multiobjetivo e através dele nos é apresentado o conceito do indicador de hypervolume. Esse indicador é obtido pelo somatório das áreas formadas — neste caso, retângulos — no espaço objetivo por cada ponto da solução Pareto-ótimas e um ponto de referência fixo. Conforme (PINTO, 2021 apud WHILE et al., 2006), "o hypervolume de um conjunto de soluções que mede o tamanho da porção do espaço objetivo que é dominado por essas soluções coletivamente". É dito por (PINTO, 2021 apud ZITZLER et al., 2003) que o hypervolume é a única métrica unária da qual eles estão cientes que é capaz de detectar que um conjunto de soluções é melhor do que outro conjunto.

A Figura 5 exemplifica o hypervolume. O ponto r indica o ponto de referência para a formação dos retângulos.

Figura 5 – Exemplificação do hypervolume



Fonte: Adaptado de Custódio, Emmerich e Madeira (2012)

2.6 Trabalhos relacionados

Nesta seção serão apresentados trabalhos acadêmicos que apresentam propostas semelhantes a desse trabalho.

Em Moro (2014), foi utilizado do problema do carteiro chinês para otimizar a roteirização dos veículos coletores de lixo na cidade de Matelândia/PR. O autor utilizou de aparelhos GPS em cada caminhão para obter a rota real feita por cada veículo e compará-la a rota gerada em seu trabalho, conseguindo realizar otimizações reais em cada rota. Para a representação do mapa da cidade também foi utilizada da estrutura grafo, para a geração de das rotas otimizadas o autor utilizou do algoritmo de Fleury. Entretanto, o cenário onde o trabalho é realizado é uma cidade relativamente pequena, de cerca de 16 mil habitantes, bem como não são realizados testes em cenários diferentes, para comprovar a validade do algoritmo em instâncias maiores. Vale destacar que, em nosso trabalho, são utilizados dois cenários para a realização dos testes: Formiga/MG, uma cidade de cerca de 68 mil habitantes (IBGE, 2023a); e Lagoa da Prata/MG com 51 mil (IBGE, 2023b) habitantes.

Scaburi (2020) em seu trabalho, utilizou, dentre outras abordagens, a meta-heurística do algoritmo genético, para realizar uma otimização na roteirização de entregadores de jornais no município de Curitiba. Em seu trabalho também é utilizada uma forma de agrupamento dos dados com base em características semelhantes entre si, no caso agrupando clientes próximos utilizando-se do problema de localização de facilidades, facilitando dessa forma a geração das rotas. Esse trabalho se destaca por realizar um comparativo entre diversas heurísticas e meta-heurísticas para resolução do problema, deixando claros todos os parâmetros utilizados em cada uma delas, além dos resultados obtidos ao final.

Em Malaquias (2006) também é utilizada da meta-heurística do algoritmo genético para a alocação das entregas do processo logístico de uma distribuidora de medicamentos, a representação das soluções e os operadores genéticos utilizados são baseados no problema do caixeiro viajante. Este trabalho é digno de nota, ao disponibilizar e detalhar diversas abstrações utilizadas no algoritmo genético utilizado, como: representação do indivíduo, operadores de mutação e cruzamento, função objetivo, etc.

No trabalho de Dereci e Karabekmez (2022) são utilizadas diversas heurísticas e meta-heurísticas para encontrar as melhores rotas, através do problema do roteamento de veículos, para a coleta de lixo realizada no local definido, nesse caso o distrito de Umraniye. Esse trabalho merece grande destaque por utilizar de diversas abordagens, como: *simulated annealing*, busca local guiada e busca tabu para encontrar a melhor rota para a realização da coleta de lixo de um município de grande porte, exibindo ao final as rotas obtidas detalhadamente.

3 Materiais e Método

Neste capítulo são apresentados os materiais, ferramentas, softwares entre outros recursos utilizados tanto para busca de informações quanto para o desenvolvimento do código durante este trabalho.

3.1 *Publish or Perish*

O software "Publish or Perish"[1] de Anne-Wil Harzing é uma ferramenta que facilita a pesquisa bibliográfica para trabalhos acadêmicos. Ele se destaca na coleta de dados de citações de diversas fontes, como Google Scholar, permitindo que os pesquisadores identifiquem facilmente trabalhos relevantes. Além disso, oferece métricas de impacto, como o número de citações, índice h e índice g, cruciais para avaliar a relevância dos artigos. A ferramenta também é eficiente na busca por artigos, livros e materiais acadêmicos. A análise de publicações ao longo do tempo proporcionada pelo software ajuda a identificar tendências e desenvolvimentos na área de estudo. Sua gratuidade para uso individual a torna uma solução acessível para acadêmicos, sua interface de usuário simplifica a análise de citações, consistindo em uma boa escolha para efetuar revisões de literatura, avaliar o impacto de publicações ou explorar tendências em determinada área de estudo.

3.2 OpenStreetMap

Para uma maior praticidade na obtenção dos dados da cidade, foi utilizada a plataforma OpenStreetMap. Ela consiste em uma iniciativa de código livre para disponibilizar dados geográficos de várias localizações do mundo gratuitamente para qualquer um que precise utilizá-los, desde que os créditos sejam dados corretamente ao projeto OSM-Foundation (2021). Essa iniciativa é mantida graças a colaboração de diversos voluntários que editam o mapa, atualizando ou mapeando novas regiões de acordo com seus conhecimentos locais. A edição do mapa pode ser realizada por diversas ferramentas, que também são de código livre e gratuitas.

Neste trabalho, para representar as ruas da cidade foi escolhida a estrutura de grafo, pois esta facilita tanto na manipulação quanto na representação da solução. A partir dos dados obtidos no OpenStreeMap, as ruas são representadas como arestas (onde o tamanho da rua é o peso da aresta) e as esquinas entre as ruas (onde geralmente o lixo é colocado para coleta) consistem nos vértices.

[1] "Publish or Perish" https://harzing.com/resources/publish-or-perish Acessado em 13 de nov. 2023

3.3 Overpass-turbo

Overpass-turbo [2] é uma API que minera os dados referentes aos mapas disponibilizados pelo OpenStreetMap. Para os dados serem extraídos, o usuário deve inserir uma consulta e executá-la, semelhante ao que acontece em consultas em bancos de dados, assim as informações referentes àquela consulta e área do mapa selecionada serão retornadas, sendo possível exportá-las em diversos formatos de arquivo.

Neste trabalho o Overpass-turbo foi utilizado nos mapas das cidades de Formiga e Lagoa da Prata para que os dados das ruas fossem exportados e manipulados.

3.4 Open Topo Data

Para gerar rotas mais realistas para os caminhões coletores de lixo, também foi considerada a altitude dos pontos que devem ser coletados, porém, através do OSM não foi encontrada uma maneira de obter essa informação dos pontos. Dado isso, para incluir tal informação nos pontos que compõe o grafo da cidade foi utilizada a API de código aberto Open Topo Data [3], que a partir da latitude e longitude de um ponto, retorna a altitude do mesmo.

Então, para cada ponto que compõe o grafo, foi realizada uma requisição, que retorna, dentre outras informações, a altitude daquele ponto em específico. O resultado de cada requisição foi armazenado em um arquivo, sendo posteriormente lido pelo programa e as altitudes são inseridas nos respectivos pontos.

3.5 Google Earth

Para uma melhor apresentação dos resultados, ao final da execução da aplicação é gerado um arquivo no formato KML que pode ser importado e visualizado pela ferramenta Google Earth [4]. Segundo Wikipedia (2023):

> Google Earth é um *software* desenvolvido e distribuído pela Artcom estadunidense do Google cuja função é apresentar um modelo tridimensional do globo terrestre, construído a partir de mosaico de imagens de satélite obtidas de fontes diversas, imagens aéreas (fotografadas de aeronaves) e GIS 3D. Desta forma, o programa pode ser usado simplesmente como um gerador de mapas bidimensionais e imagens de satélite ou como um simulador das diversas paisagens presentes no Planeta Terra.

[2] "overpass turbo."https://overpass-turbo.eu/. Acessado em 2 mai. 2023.
[3] "Open Topo Data."https://www.opentopodata.org/. Acessado em 15 mai. 2023.
[4] "Google Earth."https://www.google.com/intl/pt-BR/earth/about/. Acessado em 7 out. 2023.

3.6 Python

Inicialmente, a linguagem utilizada no projeto foi a Python em sua versão 3.8, mas após alguns problemas dessa versão com algumas bibliotecas, principalmente com a *pygmo*, optou-se pela utilização da versão 3.7 da linguagem, que não causou problemas com a importação e utilização das demais bibliotecas necessárias. Segundo Python-Software-Foundation (2022):

> Python é uma linguagem de programação interpretada, orientada a objetos e de alto nível com semântica dinâmica. Suas estruturas de dados integradas de alto nível, combinadas com tipagem dinâmica e vinculação dinâmica, a tornam muito atraente para o Desenvolvimento Rápido de Aplicativos, bem como para uso como *script* ou linguagem de 'colagem' para conectar componentes existentes. A sintaxe simples e fácil de aprender do Python enfatiza a legibilidade e, portanto, reduz o custo de manutenção do programa. Python suporta módulos e pacotes, incentivando a modularidade do programa e a reutilização de código. O interpretador Python e a extensa biblioteca padrão estão disponíveis em formato fonte ou binário gratuitamente para todas as principais plataformas e podem ser distribuídos gratuitamente.

O ambiente de desenvolvimento utilizado foi a IDE PyCharm [5] para uma maior facilidade na criação e edição dos códigos.

3.6.1 Bibliotecas Python

Para o desenvolvimento do protótipo de *software* neste TCC, foram utilizadas as seguintes bibliotecas Python:

- A biblioteca *geopy* foi utilizada principalmente para o cálculo das distâncias entre os pontos, utilizando a latitude e longitude dos mesmos.
- A biblioteca *networkx*, devido as suas diversas funcionalidades em grafos, foi utilizada para realizar as operações do grafo que representa a cidade abordada neste trabalho.
- A biblioteca *sklearn* foi utilizada para realizar o agrupamento dos pontos de coleta de lixo da cidade em *clusters* através da técnica das k-medianas.
- Para a plotagem de imagens, gráficos e mapas, foi utilizada a biblioteca *matplotlib*.
- A biblioteca *pygmo* foi utilizada para a realização do cálculo do hipervolume das funções-objetivo durante a calibragem do algoritmo.

[5] "PyCharm: o IDE Python da JetBrains para desenvolvedores ...."https://www.jetbrains.com/pt-br/pycharm/. Acessado em 2 mai.. 2023.

3.7 Git e GitHub

Para realizar a hospedagem e versionamento do código produzido durante este trabalho, foi utilizada a ferramenta Git [6] juntamente com a plataforma GitHub [7]. O Git é um sistema de controle de versão de código aberto desenvolvido por Linus Torvalds. GitHub é uma plataforma baseada em nuvem que integra o sistema de versionamento Git, ela armazena e versiona o código desenvolvido e permite que outros desenvolvedores realizem alterações nesse código, deixando todas as alterações documentadas (LONGEN, 2022).

3.8 Hardware

O equipamento utilizado durante o desenvolvimento e testes do programa foi um *notebook* da marca Samsung, modelo NP350XAA, contando com 8GB de memória RAM, processador Intel Core i5-8250U 1.80 GHz e sistema operacional Windows 10.

3.9 Método

No contexto da otimização computacional com meta-heurísticas, o método de pesquisa utilizado neste trabalho foi a pesquisa experimental. Essa abordagem corresponde a um tipo de investigação científica que implica na manipulação deliberada de variáveis independentes, com o propósito de observar e quantificar os efeitos causais nas variáveis dependentes. No caso, as variáveis independentes são os parâmetros de configuração do NSGA-II, estratégias de *crossover* e mutação, enquanto as variáveis dependentes são as métricas de avaliação das soluções geradas.

Nas seções a seguir serão explicadas as principais etapas e decisões tomadas para resolver o problema abordado nesse trabalho de conclusão de curso.

3.10 Estudo e obtenção dos dados do mapa

As cidades escolhidas para a realização do trabalho foram Lagoa da Prata/MG e Formiga/MG, essa última por se tratar do local em que vive o autor deste trabalho e onde se encontra o *Campus* do IFMG onde o mesmo realiza seus estudos.

Dessa forma se viu necessário alguma forma de obter o mapeamento de ruas da cidade, o que foi possível graças à adaptação de um algoritmo desenvolvido por Fernandes (2019) que utiliza dados das ferramentas do OpenStreetMap e também do Overpass-turbo.

[6] "Git."https://git-scm.com/. Acessado em 15 mai.. 2023.
[7] "GitHub: Let's build from here · GitHub."https://github.com/. Acessado em 15 mai.. 2023.

3.11 Estudo e adaptação da meta-heurística NSGA-II

Durante a realização da pesquisa bibliográfica em busca de alguma heurística/meta-heurística com melhores resultados para o problema do roteamento de veículos, através da ferramenta *Publish or Perish* foram encontrados diversos trabalhos que utilizaram de algoritmos genéticos para a resolução desse problema, como, por exemplo: Scaburi (2020) e Malaquias (2006).

Dado isso, as pesquisas foram mais focadas em algoritmos genéticos e em meta-heurísticas que se utilizavam do princípio de tais algoritmos evolutivos. Foi observado que, alguns TCCs da área de ciência da computação desenvolvidos no *campus*, estavam utilizando uma meta-heurística chamada NSGA-II, baseada em algoritmos evolutivos para a resolução de problemas multiobjetivo, tendo sido obtidos bons resultados, como, por exemplo, os trabalhos de: Cândido (2019), Pinto (2021) e Soares (2021). Portanto, a meta-heurística NSGA-II foi escolhida para a realização do trabalho.

A implementação do algoritmo NSGA-II foi baseada em código desenvolvido e validado por Pinto (2021), naturalmente, uma nova modelagem do problema se fez necessária, em especial na abstração dos indivíduos e operações entre eles para que a meta-heurística ficasse mais bem-adaptada ao problema tratado neste trabalho.

3.11.1 Representação de um indivíduo (solução)

Em algoritmos evolutivos, cada indivíduo presente em uma população representa uma solução para o problema abordado, neste caso não poderia ser diferente. Para o problema de confecção das rotas tratado neste trabalho, optou-se por abstrair o indivíduo como uma espécie de parametrização da coleta, ele deve representar: a quantidade de caminhões a ser utilizada no mapa, em quantos agrupamentos o mapa deve ser subdividido e onde começar em cada agrupamento. Para mais informações sobre essa abstração veja a Seção 4.3

3.12 Implementação do cruzamento entre indivíduos

No cruzamento as informações de dois indivíduos são combinadas e passadas adiante, mas no NSGA-II também existe a possibilidade de não cruzamento, em que um indivíduo pode não realizar essa operação e apenas um clone seu será passado para a próxima população.

O operador de cruzamento escolhido foi o uniforme simples, que é melhor descrito na Subseção 4.3.2

3.12.1 Implementação da mutação entre indivíduos

A mutação é um artifício utilizado para realizar perturbações nos resultados de uma população e dessa forma escapar de ótimos locais. Inicialmente foi desenvolvido um operador de mutação simples, em que um gene aleatório do genoma sofria uma leve perturbação em que seu valor era somado em uma unidade, mas após alguns experimentos, foi observado que o mesmo não estava sendo suficiente para evitar ótimos locais, sendo proposto então um novo método que utiliza de mudanças mais radicais no genoma, podendo ter seu valor somado ou subtraído em valores maiores que um. Para mais informações veja a Subseção 4.3.3

3.12.2 Implementação das funções objetivo

As funções objetivo são de extrema importância para o projeto, uma vez que é através delas que o algoritmo "entende" o quão boa ou ruim uma solução é. Após diversos experimentos com diferentes tipos de funções objetivo, definiu-se que seriam utilizadas três: O tempo gasto durante a coleta, a quantidade de caminhões utilizada e a variação de altitude durante a coleta, sendo todas métricas de minimização. Essas métricas são melhor descritas na seção de desenvolvimento nas subseções 4.3.5, 4.3.6 e 4.3.7 respectivamente.

3.13 Calibragem dos parâmetros via Projeto Fatorial 2^k

Para realizar os experimentos com os parâmetros calibrados para a entrada de dados e o tipo de problema a ser abordado pela heurística NSGA-II, gerando bons resultados em um tempo plausível, foram realizadas três rodadas do projeto fatorial 2^k. Essa etapa é importante para que a meta-heurística tenha bem definido seu espaço de busca, não sendo apenas valores previamente calibrados no NSGA-II para uma instância ou problema diferente, o que ocasionaria uma busca ineficiente ou limitada no espaço de soluções, demandando muito tempo adicional em sua execução.

Para realizar a calibração dos parâmetros de forma mais eficaz foi utilizado o cálculo do hiper-volume que apontava o quão boa era uma combinação de parâmetros em relação a outra, ao comparar os hipervolumes da solução obtida por cada combinação de parâmetros do NSGA-II experimentada. Mais detalhes podem ser encontrados na Seção 4.5

3.14 Execução dos experimentos

Após realizados todos os procedimentos acima, pode-se inicializar a execução dos experimentos. Neste caso foram utilizados três cenários.

O primeiro é um ambiente de teste, que corresponde a um bairro da cidade de Formiga/MG, isso foi feito, pois o mapa completo levava mais tempo para executar, o que atrasava a realização iterativa de implementação, testes e ajustes, e da modelagem do problema à heurística do algoritmo genético.

O segundo cenário é a execução no mapa da cidade de Formiga/MG, onde o estudo foi inicialmente proposto, por se tratar da cidade onde se encontra o *campus* onde esse estudo foi realizado e também a cidade em que reside o autor.

Por fim foi proposto um terceiro cenário, que corresponde a cidade de Lagoa da Prata/MG, localizada próxima à cidade Formiga/MG. Esse cenário foi proposto para demonstrar que o algoritmo pode ser executado em outras localidades, e não exclusivamente em uma cidade específica.

4 Desenvolvimento

Neste capítulo são detalhadas e explicadas as decisões tomadas durante todo o processo de desenvolvimento deste projeto, tratando tanto da implementação utilizada quanto das abstrações necessárias para uma melhor resolução do problema.

4.1 Obtenção do mapa da cidade

O mapa da cidade foi obtido através da ferramenta Overpass-Turbo que utiliza o mapa disponibilizado pelo OpenStreetMap para retirar dados. A partir de uma consulta, foi requisitado a API do Overpass-Turbo que retornasse os dados de todas as ruas refentes a área selecionada, que no caso deste trabalho foram escolhidas a cidade de Formiga e de Lagoa da Prata, ambas no estado de Minas Gerais.

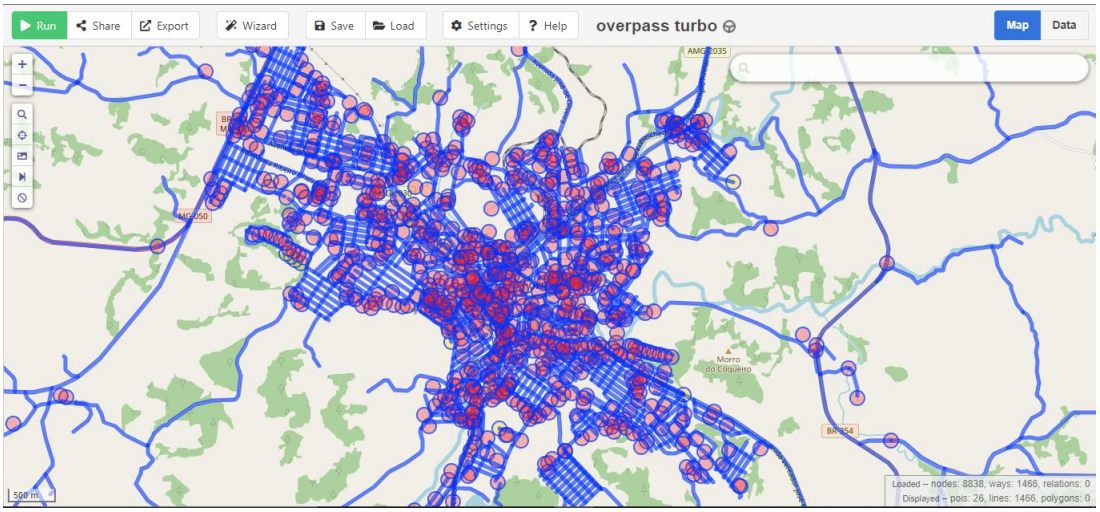
Quadro 1 – Consulta realizada na ferramenta Overpass-turbo

```
[timeout:25];
(
          highway!=pedestrian and "-highway"!=path"
          node["highway"]["highway"!="footway"]

["highway"!="pedestrian"]
["-highway"!="path"]({{bbox}});
          way["highway"]["highway"!="footway"]

["highway"!="pedestrian"]
["-highway"!="path"]({{bbox}});
          relation["highway"]["highway"!="footway"]

["highway"!="pedestrian"]
["-highway"!="path"]({{bbox}});
);

out body;
>;
out skel qt;
```

A consulta utilizada para buscar os dados, presente no Quadro 1, retira o máximo de dados que não serão utilizados para o trabalho, como estabelecimentos comerciais, igrejas, escolas, etc. e busca somente por ruas e rodovias na área da cidade. Apesar da consulta,

alguns dados indesejados ainda acabam sendo obtidos, estes são retirados manualmente durante a execução do código para que a consulta não fique muito extensa. Todos esses dados foram obtidos e baixados em formato de um arquivo XML. A Figura 6 mostra os dados selecionados na consulta.

Figura 6 – Captura inicial dos dados da cidade de Formiga/MG



Fonte: Próprio autor

4.1.1 Estrutura do arquivo XML

A representação de uma rua é feita a partir de uma sequência de pontos, todos esses pontos e suas informações, como latitude e longitude, são também exportadas para o arquivo XML. É possível que também venham informações desnecessárias como a localização de semáforos, estabelecimentos comerciais, igrejas entre outros. Para garantir que essas informações não atrapalhem a análise dos dados, foi feita antes de tudo uma limpeza de dados desnecessários que possam ter vindo juntamente com os dados que serão realmente analisados.

A estrutura do arquivo possui duas seções principais, sendo: seção de informação dos pontos, contendo latitude, longitude, identificador e outras informações a respeito do ponto específico; e a seção com dados das ruas, as quais são formadas a partir dos pontos listados na primeira seção e algumas outras informações como nome da rua, pavimentação, etc.

O local para onde todo o lixo deve ser levado, foi representado por um ponto um pouco mais distante das cidades, a fim de representar um aterro sanitário, por exemplo. Mas vale ressaltar que esse ponto pode ser configurado pelo usuário segundo a sua necessidade, para o algoritmo gerar rotas mais condizentes com sua realidade.

4.1.2 Manipulação do arquivo XML

Para uma maior praticidade ao manipular o documento XML, foi utilizada a biblioteca *xml.etree.cElementTree*[1], sendo uma biblioteca desenvolvida para fornecer uma maior facilidade na leitura e criação de arquivos XML.

Inicialmente a manipulação obtém o elemento "raiz" do arquivo, que corresponde a *tag* que abre o arquivo. A partir disso é iniciado um laço de repetição que passa por todas as *tags* presentes no arquivo analisado, se tornando mais fácil caminhar entre as demais *tags* e seus respectivos atributos, além de *tags* filhas que podem conter mais informações. Assim, utilizando dessa biblioteca, foi feita uma limpeza de dados que acabam vindo com os dados das ruas, como estabelecimentos comerciais, igrejas, semáforos, etc.

Após todas essas informações indesejadas serem retiradas, restaram apenas informações que seriam realmente utilizadas durante o trabalho, sendo as informações sobre as ruas e os pontos que as formam. Essas informações foram mapeadas nas classes *Rua.py* e *Ponto.py* que representam, respectivamente, uma rua e um determinado ponto daqueles que formam a rua. Todos esses dados mapeados foram armazenados em estruturas de dados da linguagem Python chamadas dicionários, para uma maior velocidade ao acessar esses dados.

4.1.3 Visualização do mapa obtido para a cidade

Para confirmar que os dados mapeados pela aplicação estavam condizentes com as informações do mapa, foi realizado um teste com a biblioteca *gmplot*[2]. Neste teste as coordenadas dos pontos e ruas mapeadas pelo algoritmo foram passadas a uma função disponibilizada pela biblioteca, para que a mesma gerasse um arquivo HTML com os mesmos dados plotados na base do mapa do Google Maps, gerando a Figura 7.

Ao aproximar a imagem (veja Figura 8) é possível verificar que algumas poucas ruas não foram mapeadas ou então estão um pouco deslocadas. Isso era esperado, pois os dados foram inicialmente retirados de uma ferramenta diferente do Google Maps, sendo normal haver alguma diferença dado que o OpenStreetMap é uma ferramenta aberta para edição, diferente da ferramenta do Google. Mas após uma verificação minuciosa do mapa, não foram encontradas diferenças significativas entre ambos, confirmando então que os dados mapeados pelo programa estão corretos.

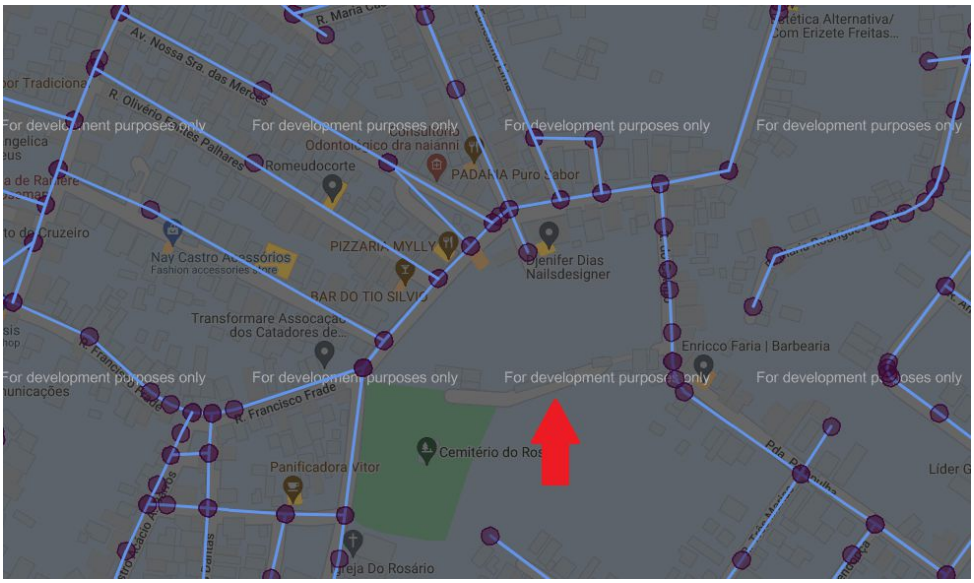
A Figura 8 mostra com mais detalhes, é possível ver indicado pela seta vermelha uma rua que acabou não sendo mapeada.

[1] Disponível em; <https://docs.python.org/2/library/xml.etree.elementtree.html>
[2] Disponível em: <https://pypi.org/project/gmplot/>

Figura 7 – Verificação dos dados filtrados e mapeados

Fonte: Próprio autor

Figura 8 – Detalhe dos dados mapeados



Fonte: Próprio autor

4.2 Grafo da cidade

Com o mapa da cidade gerado e os dados que o formam conferidos, foi possível montar o grafo que representa a cidade utilizando da biblioteca *NetworkX* que possui diversas funcionalidades implementadas para abstração de grafo. Para essa representação do grafo foi utilizada a classe *MultiGraph*, sendo uma forma de representação de grafos

pela biblioteca que aceita laços e arestas paralelas.

Durante a montagem do grafo, foi realizado o cálculo de comprimento das ruas, para representar os pesos das arestas no grafo. Para realizar esse cálculo com uma maior precisão, foi utilizada a função de cálculo da distância geodésica da biblioteca *geopy*. A distância geodésica entre dois pontos é aquela em que, diferente da geometria plana, considera a menor distância entre dois pontos num espaço tridimensional considerando a curvatura da Terra.

Figura 9 – Primeiro grafo montado pelo algoritmo

Fonte: Próprio autor

Iterando sobre cada rua presente no mapa e sobre os pontos que as compõe, o grafo foi montado e, ao final, plotado para visualização, resultando em um grafo bem detalhado e aparentemente mais condizente com o mapa da cidade analisada, como mostra a Figura 9.

4.2.1 Modelagem da quantidade de lixo gerado e simplificação do grafo

Uma informação de suma importância e necessária à simulação são as demandas de lixo de cada ponto de coleta, mas antes disso, foi necessário realizar uma simplificação no mapa da cidade, visto que uma mesma rua possui diversos pontos (segmentos de reta) que definem as curvas que a mesma realiza.

Nessa simplificação, foram destacados os pontos que representam esquinas, pois normalmente é onde o lixo é armazenado nos bairros, eliminando do novo grafo os segmentos

da rua (segmentos) que servem para representar as curvas que ela faz. A Figura 10 mostra o grafo simplificado, estrutura de dados auxiliar gerada a partir do grafo original.

Figura 10 – Grafo simplificado da cidade

Fonte: Próprio autor

Dessa forma, o grafo sofreu uma grande redução de informações, uma vez que foram retirados muitos pontos que não seriam interessantes para o algoritmo. No entanto, destacamos que as informações importantes não foram perdidas, pois para calcular o comprimento da rua foi utilizado o grafo original detalhado, que possui todos os pontos mapeados.

Figura 11 – Comparação entre os grafos gerados

(a) Grafo original (b) Grafo simplificado

Fonte: Próprio autor

Na Figura 11 é mostrada uma comparação entre os dois grafos. Observamos que esse novo grafo possui algumas “discrepâncias” se comparado ao anterior, visto que as curvas realizadas pelas ruas foram suprimidas nesta cópia simplificada do grafo original, mas

como dito acima, nenhuma informação relevante foi perdida, uma vez que o comprimento das ruas é calculado com base no grafo original e os pontos que representam esquinas se mantiveram em ambos os grafos.

O grafo mais detalhado da cidade foi utilizado para confirmar se as ruas mapeadas estavam condizentes com a realidade, o que foi verificado manualmente pelo autor de forma empírica e amostral. Esse grafo mais simples será utilizado nos experimentos para reduzir o espaço de busca a ser explorado pelo algoritmo NSGA-II, visando fornecer uma maior velocidade na análise das informações e resultados.

Para o cálculo da quantidade de lixo gerado em uma determinada rua, foi considerado um comprimento médio (aproximado) da frente de lote/casa na cidade de Formiga, sendo de 10 metros. Esse número foi obtido por meio de uma aproximação realizada no trabalho de Fernandes (2019). Sabendo disso, foi possível definir aproximadamente quantas casas existem na rua, e considerando que cada casa possui em média 3 pessoas (IBGE, 2019), que geram em média 3 quilogramas de lixo por dia (PIRES; OLIVEIRA, 2021), foi calculada a quantidade total de lixo que em média seria gerada pelas casas em cada rua da cidade. Gerada a demanda, a mesma foi distribuída igualmente entre os pontos que formam aquela rua.

4.2.2 Altitude dos pontos de coleta

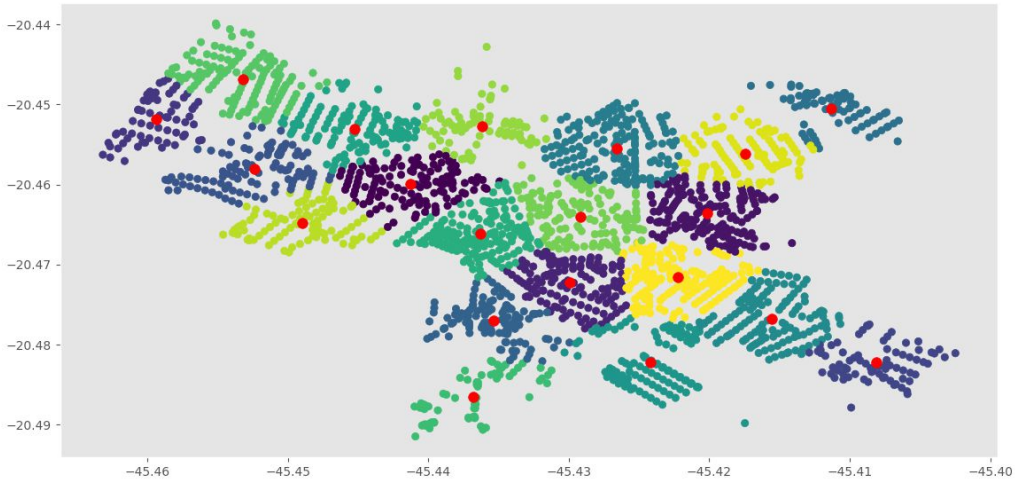
Para tornar o grafo da cidade ainda mais condizente com a realidade, foram adicionadas as altitudes de cada um dos pontos de coleta do mapa. Isso foi feito através da API *Open Topo Data* citada na Seção 3.4. Ao iniciar a aplicação um arquivo contendo o identificador de cada ponto e sua respectiva altitude é lido e assim as informações são importadas para os pontos, dispensando a necessidade de realizar consultas a API a cada execução do algoritmo.

4.2.3 Clusterização dos pontos do grafo utilizando o algoritmo *k-means*

Com o grafo da cidade devidamente montado, é possível agora realizar os agrupamentos que serão percorridos pelos veículos durante a coleta. Isso é feito para acelerar o processamento do algoritmo, em razão de que o grafo da cidade é dividido em grupos menores (como se fossem bairros), e também para evitar que um caminhão tente realizar toda a coleta de lixo da cidade sozinho. O algoritmo utilizado para a realização dos *clusters* foi o *k-means* também conhecido como k-medianas, disponível na biblioteca *Scikit-Learn*. Na Figura 12 é ilustrado um exemplo de agrupamentos realizados pelo algoritmo, onde cada cor representa um *cluster* e cada ponto vermelho indica o centro de um.

Cada um dos agrupamentos gerados foi transformado em um subgrafo para que fosse percorrido separadamente pelo caminhão coletor. Para realizar o trajeto do caminhão,

Figura 12 – Exemplo de clusterização



Fonte: Próprio autor

foi utilizado o algoritmo do caminho euleriano, cujo objetivo é percorrer todas as arestas de um grafo sem que exista repetição, dessa forma é garantido que todas as ruas do agrupamento serão atendidas e todo o lixo recolhido.

Porém, durante a montagem desses subgrafos, alguns acabavam por não serem grafos eulerianos, o que impossibilitava a geração de um caminho euleriano por eles. Dito isso, foi necessária uma conversão desses subgrafos em grafos eulerianos. Segundo definição, para um grafo ser euleriano ele precisa ser conexo e também possuir todos seus vértices com grau par. Observe que a Figura 11 apresenta situações de ruas do tipo término de via (ex.: as ruas em formato de "T"), na qual o caminhão precisaria passar duas vezes pela rua (entrar por ela e voltar por ela) para alcançar determinado cruzamento. Neste exemplo, o vértice em questão (esquina) tem grau ímpar (grau três).

Foi desenvolvida uma função para conectar os grafos desconexos e para garantir que todos os vértices possuíssem grau par, foi utilizada a função *eulerize* da biblioteca *NetworkX*, que transforma um grafo conexo passado por parâmetro em um grafo euleriano. Todas essas operações foram feitas sem que os dados representados pelo grafo fossem alterados, no caso, apenas foi permitido que algumas das ruas fossem atravessadas mais de uma vez (ex.: ruas sem saída ou ruas que ficaram na periferia do subgrafo gerado).

4.2.4 Montagem do *cache* de agrupamentos do grafo

Durante a conversão dos subgrafos da clusterização em subgrafos eulerianos, notou-se que essa rotina demandava muito tempo de execução, e como cada geração do algoritmo realizava essa operação diversas vezes, ocasionou uma lentidão excessiva ao executar a

aplicação. Assim, optou-se por, sempre que o programa for iniciado, montar um *cache* com os agrupamentos previamente calculados (memoização).

Esse *cache* é realizado logo no início da aplicação, quando o grafo simplificado é montado. As possibilidades de agrupamentos do grafo são armazenadas em um dicionário, desde aquela com mínimo de *clusters* até a com o máximo. Assim, a rotina de conversão de grafos eulerianos teve seu número de chamadas significativamente diminuído, e consequentemente, o tempo de execução do algoritmo. Dessa forma, sempre que é necessário obter algum agrupamento, basta acessar o dicionário que contém todas as possibilidades calculadas.

4.3 Implementação do NSGA-II

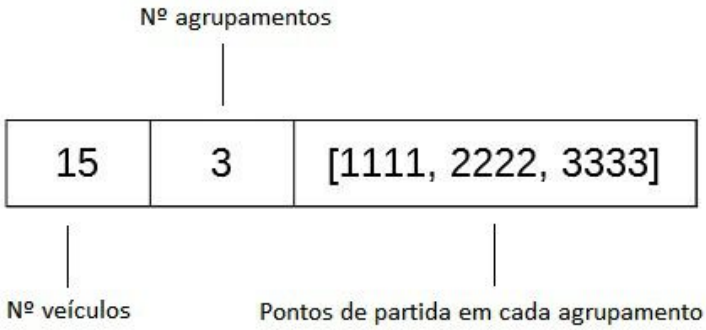
O NSGA-II é uma meta-heurística genérica, pensada para ser usada em diversos problemas, portanto qualquer problema que busque ser resolvido por meio da estratégia do NSGA-II, primeiramente deve ser adaptado e abstraído de forma que a meta-heurística possa resolvê-lo.

Nas próximas seções serão exibidos detalhes de como essa abstração foi realizada para que o problema abordado neste trabalho se encaixasse no formato do NSGA-II.

4.3.1 Abstração do indivíduo

O indivíduo é um elemento extremamente importante em qualquer algoritmo evolutivo, por representar as soluções que o algoritmo irá analisar e retornar após a sua execução. Neste trabalho o indivíduo deve representar em seu genoma, as configurações necessárias para a realização da coleta em todo o mapa pelos veículos coletores. A Figura 13 exemplifica o indivíduo.

Figura 13 – Representação de um indivíduo



Fonte: Próprio autor

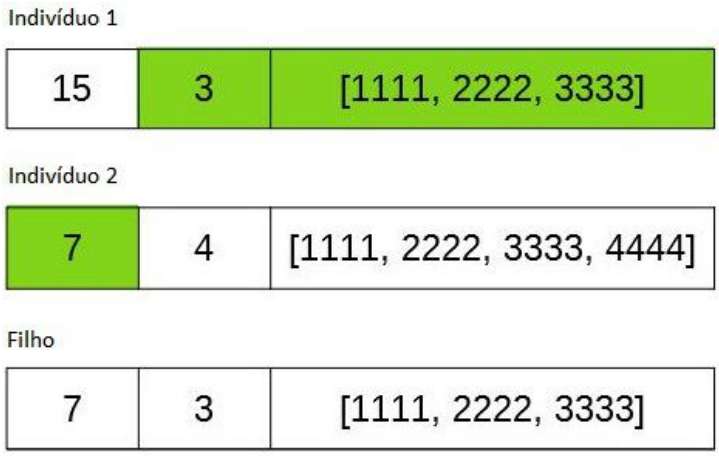
Para isso, foi desenvolvida uma representação do indivíduo que segue a seguinte lógica: o primeiro elemento do genoma é a quantidade de caminhões que realizarão o

percurso pelo mapa, o segundo elemento é a quantidade de agrupamentos nos quais o mapa da cidade será dividido, cada agrupamento será percorrido por um veículo, por fim, o último elemento do genoma é uma lista com os pontos onde deve ser iniciado o percurso em cada um dos agrupamentos, conforme a quantidade gerada.

4.3.2 Cruzamento entre indivíduos

Para a realização do cruzamento entre dois indivíduos para a geração de um terceiro para a população, foi proposto o método de *crossover* uniforme simples. Neste método é utilizada uma probabilidade fixa, neste caso de 50%, de se realizar a troca entre os genomas dos pais.

Figura 14 – Exemplo de cruzamento entre dois indivíduos



Fonte: Próprio autor

A Figura 14 demostra um indivíduo gerado a partir de dois outros utilizando o *crossover* simples uniforme, na qual a primeira informação foi copiada do segundo pai e as outras duas do primeiro. Por exemplo: Dados dois indivíduos e seus genomas, é realizado um sorteio com a probabilidade de 50% de se escolher um deles, aquele que for escolhido, terá o primeiro item de seu genoma, que corresponde ao número de veículos coletores, copiado para o novo indivíduo a ser gerado, enquanto os outros dois itens do genoma serão copiados do indivíduo que não foi selecionado. Os dois últimos itens do genoma, quantidade de agrupamentos e pontos de início, devem vir do mesmo indivíduo, visto que possuem relação entre si.

4.3.3 Mutação

A mutação é uma forma de gerar perturbações nos indivíduos para que novas características sejam incorporadas a eles, trazendo assim uma maior diversidade para a população e escapando de ótimos locais.

Inicialmente a mutação desenvolvida para o algoritmo escolhia apenas um dos dois primeiros genes do indivíduo, aumentava seu valor em uma unidade. Mas foi observado durante os testes da aplicação, que essa abordagem estava causando pouco impacto na evolução dos indivíduos, fazendo com que o algoritmo estagnasse em ótimos locais.

Portanto, foi implementada uma forma de mutação mais radical, que funciona da seguinte maneira: caso a mutação ocorra no indivíduo, é sorteado com uma probabilidade de 50% qual dos dois primeiros genes sofrerá a mutação. Definido isso, é feito um novo sorteio para decidir se a quantidade do gene será somada ou subtraída, e em seguida, um novo sorteio para obter a quantidade a ser somada ou subtraída, que varia de 1 a 10. Após essas operações são feitas verificações para garantir que o máximo/mínimo de caminhões ou de agrupamentos não seja desrespeitado. Quando a mutação ocorre no segundo genoma, que corresponde a quantidade de agrupamentos, é necessário realizar novamente o sorteio do terceiro genoma, que corresponde aos pontos de início das rotas nos agrupamentos. A Figura 15 exemplifica o processo, nela o gene da quantidade de veículos sofre a mutação, com seu valor somado em 4.

Figura 15 – Exemplo de mutação de um indivíduo

Fonte: Próprio autor

Dessa forma, a mutação do indivíduo ficou mais radical, uma vez que alterava mais o indivíduo do que a mutação desenvolvida anteriormente. Foi observado durante novos testes que o desempenho do algoritmo melhorou, não ficando preso em ótimos locais e tendo uma maior variabilidade nas respostas.

4.3.4 Quantidade de agrupamentos (*clusters*) para a cidade

Para a geração do indivíduo, são gerados valores aleatórios num certo intervalo. Durante alguns testes notou-se que em alguns casos o algoritmo estava demorando muito para executar. Ao investigar isso, percebeu-se que alguns indivíduos eram gerados com

valores muito baixos para o parâmetro da quantidade de *clusters*, o que acarretava numa quantidade pequena de agrupamentos, porém cada um com um subgrafo muito grande. Por conta disso, tais configurações no genoma de indivíduos causava grande demora para ser processado pelo algoritmo do caminho euleriano.

Para resolver esse problema, o intervalo da geração da quantidade inicial mínima de agrupamentos do indivíduo foi alterado, deixando de ser um valor fixo (anteriormente o valor 1), e passando para um valor calculado, dado pela divisão da quantidade de lixo total da cidade pela capacidade média do caminhão coletor. Para não deixar o espaço de busca do algoritmo muito limitado, esse valor ainda foi divido por dois. Após essas alterações, notou-se uma melhora no tempo de execução da aplicação, confirmando que o problema era de *clusters* muito grandes sendo processados pelo algoritmo do caminho euleriano.

4.3.5 Função objetivo: Minimização do tempo de trabalho dos caminhões

Essa função objetivo tem como finalidade favorecer configurações que façam com que os veículos coletores percorram o mínimo possível, diminuindo assim também o tempo de coleta de lixo da cidade. Durante a simulação da coleta de lixo, é gerada uma lista com o tempo rodado por cada veículo para a realização da sua respectiva rota, ao final da simulação é verificado qual veículo ficou mais tempo rodando, então seu valor é retornado para o algoritmo minimizar a distância percorrida. Desse modo, evita-se que sejam alocados poucos caminhões para a realização de toda a coleta da cidade, o que resultaria num custo menor, mas consequentemente em várias horas para a finalização do recolhimento de lixo.

Por exemplo, algum genoma pode fornecer uma configuração na qual apenas um caminhão seja responsável por todo o lixo da cidade. Por mais que alguns custos pudessem ser diminuídos, esse veículo levaria dias para terminar toda a coleta, deixando vários pontos de lixo expostos durante várias horas, causando diversos transtornos para a população, além de submeter os trabalhadores a longas jornadas de trabalho.

4.3.6 Função objetivo: Minimização da quantidade de caminhões

Para que um veículo coletor de lixo possa funcionar, existem diversos custos que devem ser considerados, como combustível, manutenção, além dos custos das pessoas que trabalharão naquele veículo. Portanto, para a coleta ser realizada com baixo custo, deve ser utilizada a menor quantidade de veículos coletores possível.

Porém, não se pode simplesmente retornar como resposta a utilização de apenas um caminhão como visto no tópico anterior. O algoritmo também não pode responder com a quantidade máxima de caminhões, que por mais que realizassem a coleta em pouquíssimo tempo, demandariam gastos desnecessários para a cidade.

Assim, tanto uma resposta que se utiliza de muitos veículos quanto uma que utiliza

poucos geram uma coleta ruim, seja pelo gasto desnecessário com muitos caminhões ou pela demora de coleta utilizando-se poucos. Considerando isso, o algoritmo genético deve decidir qual a quantidade de caminhões que não deixa a coleta muito lenta, mas também não a deixa com gastos excessivos.

4.3.7 Função objetivo: Minimização da variação de altitude

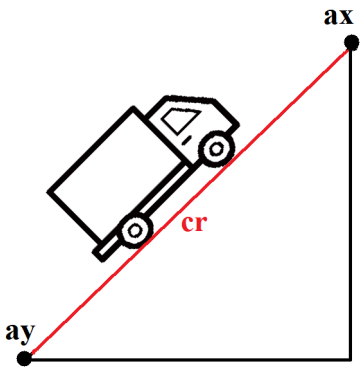
As cidades de Formiga e Lagoa da Prata, utilizadas para a realização dos testes e experimentos deste trabalho, encontram-se na região oeste de Minas Gerais, sendo uma região com um relevo montanhoso. Dito isso, foi considerada essa característica para uma melhor montagem das rotas. Como previamente citado na Subseção 4.2.2, durante a montagem do grafo da cidade, foram adicionadas as altitudes dos pontos de coleta. Isso foi feito para melhorar as decisões do algoritmo na rota e evitar que o caminhão realize subidas e descidas em ruas íngremes sem necessidade, o que acarretaria um gasto de combustível maior para os veículos.

O cálculo foi realizado da seguinte maneira: A medida que o veículo coletor percorre a rota gerada, ele passa em diversos segmentos, de um ponto para o outro através da rua que os liga, cada um desses pontos possui uma altitude diferente, dessa forma é realizado o seguinte cálculo:

$$\Delta a = \frac{|ax - ay|}{cr}$$

Onde ax representa a altitude de um ponto x, ay representa a altitude de um ponto y e cr representa o comprimento do segmento da rua que liga os pontos x e y. A Figura 16 ilustra um cenário onde um veículo transita por uma rua íngreme.

Figura 16 – Exemplo de veículo em rua íngreme



Fonte: Próprio autor

A diferença de altitudes deve ser dividida pelo comprimento do segmento em questão, para identificar o quão íngreme ele é, por exemplo, uma variação de 50 metros em uma rua que possui 1 quilômetro mostra que ela é uma rua com pouca variação, mas a

mesma variação em uma rua com 100 metros de comprimento, indica que aquela é uma rua bastante íngreme.

4.3.8 Avaliação do indivíduo

Como o indivíduo representa um conjunto de parâmetros que definem como deve ser realizada a coleta na cidade, ao avaliá-lo é necessário realizar uma simulação dessa coleta com os parâmetros do mesmo.

Foi desenvolvido um algoritmo que, segundo os parâmetros do indivíduo, obtém o número de agrupamentos correspondentes do *cache* criado e intercala a quantidade de caminhões entre esses agrupamentos, onde cada caminhão deve percorrer um *cluster*. Como mencionado anteriormente, a rota no *cluster* é definida pelo algoritmo do caminho euleriano. Caso durante os percursos a capacidade do veículo seja atingida, ele deve voltar ao depósito, descarregar e retornar ao ponto de onde estava coletando.

Esse processo é repetido até que todo o mapa da cidade seja percorrido e o lixo coletado. Ao final as métricas da coleta daquele indivíduo são calculadas e retornadas.

4.3.9 Normalização das métricas do indivíduo

Após o indivíduo ser avaliado e o valor das três funções objetivo pontuadas acima ser obtido, percebeu-se que eles naturalmente tinham ordens de magnitude diferentes, já que o tempo de trabalho normalmente retornava valores bem maiores que a quantidade de caminhões e que a variação de altitude. Dessa forma se viu a necessidade de normalizar esses resultados, para que o algoritmo do NSGA-II não favorecesse nenhum desses valores em particular.

Inicialmente utilizou-se o método de normalização min-max presente na biblioteca *Scikit-Learn*. Porém, após a realização dos primeiros testes foi verificado que o algoritmo não estava conseguindo evoluir, e em certos casos até piorando a população. Realizando testes mais focados na etapa de normalização, notou-se que, devido à grande discrepância dos dados, a normalização min-max não estava funcionando plenamente.

Assim sendo, novos métodos de normalização foram pesquisados na biblioteca, para a correção do problema, chegando até o método *MaxAbsScaler*. Esse escalar de normalização, segundo sua documentação, é particularmente utilizado para a normalização de dados muito esparsos entre si, visto que realiza a normalização e não destrói o intervalo que existe entre as informações transformadas, diferentemente da normalização min-max.

4.4 Testes preliminares e ajustes para validação do algoritmo

Para a realização de testes preliminares com o algoritmo desenvolvido, se tornou inviável a utilização do mapa completo da cidade, visto que para cada execução do projeto algumas horas eram necessárias. Portanto, foi mapeada uma porção menor do mapa da cidade, correspondente a um bairro, para que a realização dos testes preliminares não ficasse tão demorada. Vale esclarecer que tais testes foram realizados durante a implementação do algoritmo (momento de ideação e modificações), ou seja, antes do experimento de validação e coleta dos resultados.

4.5 Calibração dos parâmetros por meio do Projeto Fatorial 2^k

Segundo definição de Soares (2021), o projeto fatorial 2^k consiste na realização de experimentos para encontrar a melhor configuração para cada fator do algoritmo a partir de duas opções, sendo assim 2 níveis para cada k parâmetro de configuração. Isso é feito para o algoritmo ser melhor calibrado e obtenha mais rapidamente melhores resultados para o problema tratado em questão. Os experimentos de calibração foram realizados nos seguintes parâmetros:

- Número de gerações
- Tamanho da população
- Probabilidade de cruzamento
- Taxa de mutação

Conforme os parâmetros citados acima, foram avaliadas as métricas do valor de hiper-volume e do tempo de execução do algoritmo. Foram realizadas três rodadas do projeto fatorial, na primeira rodada foram escolhidos valores para definir os limites, tanto inferior quanto superior, dos parâmetros. Com o passar das rodadas, os resultados são interpretados e os valores de limite são alterados para, nesse caso, fornecer um menor tempo de execução e um menor valor de hiper-volume.

Para a execução desta etapa do trabalho, foi desenvolvida uma forma de executar em ordem aleatorizada cada uma das configurações do projeto fatorial. Isso foi feito para que se evite que as iterações (“repetições”) de uma mesma configuração sejam executadas em sequência, o que poderia trazer resultados indesejados, sendo que o computador pode iniciar algum processo em segundo plano e acabar prejudicando o tempo de execução daquela sequência de parâmetros.

4.5.1 Primeiro experimento do projeto fatorial 2^k

A Tabela 1 mostra os valores iniciais utilizados no primeiro experimento fatorial. Esses valores foram obtidos de testes anteriores ao projeto fatorial 2^k, durante o desenvolvimento do algoritmo e apresentaram visualmente bons resultados. Lembrando que o objetivo neste caso é diminuir o valor do hiper-volume, para minimizar o resultado das funções objetivo.

Tabela 1 – Níveis dos fatores no primeiro projeto fatorial

	FATORES				**FATORES**				**FIXO**
k = 4	A (gerações)		B (população)		C (taxa de mutação)		D (prob. cruzamento)		Iterações
	1	50	1	50	1	0.20	1	0.60	3
NÍVEIS	2	100	2	100	2	0.40	2	0.70	3

Fonte: Autor.

Os parâmetros destacados em verde mais escuro na tabela acima, foram aqueles que obtiveram uma melhor combinação entre valor do hiper-volume e tempo de execução. Dessa forma pode-se observar que o parâmetro do número de gerações obteve uma leve tendência a diminuir seu valor, visto que o valor de 50 gerações obteve melhores resultados. No caso dos parâmetros do tamanho da população e probabilidade de cruzamento notou-se o oposto, uma vez que ambos tenderam a aumentar seus valores, obtendo melhores resultados com a população com 100 indivíduos e uma probabilidade de cruzamento de 70%. No caso da taxa de mutação, observou-se uma grande tendência do algoritmo para seu maior valor, favorecendo muito uma taxa de mutação de 40%.

Portanto, conforme o primeiro experimento fatorial, a configuração que melhor favoreceu os resultados do algoritmo foi 1, 2, 2 e 2 (50 gerações, 100 indivíduos na população, taxa de mutação de 40% e taxa de cruzamento de 70%).

4.5.2 Segundo experimento do projeto fatorial 2^k

No segundo projeto fatorial, os valores dos parâmetros foram ainda mais explorados. Então, segundo os resultados do experimento anterior, as seguintes alterações foram realizadas: O limite mínimo da quantidade de indivíduos por população foi de 50 para 80, enquanto o limite máximo foi alterado de 100 para 130. A taxa de mutação inferior foi de uma probabilidade de 20% para 40%, enquanto limite superior foi de 40% para 60%. A probabilidade de cruzamento inferior foi de 60% para 70%, enquanto o limite superior foi de uma taxa de 70% para 80%.

Em relação à quantidade de gerações, mesmo com o experimento anterior apontando que a mesma deveria sofrer uma diminuição dos seus limites, devido ao tempo de execução

ser mais alto, foi decidido aumentá-los para dar ainda mais tempo do algoritmo amadurecer os resultados. Dessa forma, a quantidade de gerações sofreu um aumento em seu limite inferior, saindo de 50 para 60 gerações, enquanto seu limite superior foi de 100 para 120 gerações. A Tabela 2 mostra essas alterações.

Tabela 2 – Níveis dos fatores no segundo projeto fatorial

	FATORES				**FATORES**				**FIXO**
k = 4	A (gerações)		B (população)		C (taxa de mutação)		D (prob. cruzamento)		Iterações
	1	60	1	80	1	0.40	1	0.70	3
NÍVEIS	2	120	2	130	2	0.60	2	0.80	3

Fonte: Autor.

Assim como na tabela da seção anterior, os parâmetros destacados em verde mais escuro foram os que retornaram melhores resultados. Neste experimento pode notar-se que os parâmetros do número de gerações, taxa de mutação e probabilidade de cruzamento tiveram uma tendência de diminuição, enquanto o tamanho da população tendeu a aumentar.

Visto isso, no segundo experimento do projeto fatorial, observou-se que a configuração que retornou melhores resultados foi 1, 2, 1 e 1 (60 gerações, 130 indivíduos por população, taxa de mutação de 40% e probabilidade de cruzamento de 70%)

4.5.3 Terceiro experimento do projeto fatorial 2^k

Em busca de aprimorar ainda mais os parâmetros do algoritmo, foi proposto o terceiro e último experimento do projeto fatorial 2^k, onde foram realizadas as seguintes alterações: O número de gerações sofreu um aumento, pelo mesmo motivo do experimento anterior, a quantidade de indivíduos também sofreu um aumento, enquanto os valores da taxa de mutação e de cruzamento sofreram uma queda, devido aos resultados do segundo experimento. Todos os valores utilizados podem ser verificados na Tabela 3.

Tabela 3 – Níveis dos fatores no terceiro projeto fatorial

	FATORES				**FATORES**				**FIXO**
k = 4	A (gerações)		B (população)		C (taxa de mutação)		D (prob. cruzamento)		Iterações
	1	90	1	110	1	0.45	1	0.65	3
NÍVEIS	2	110	2	150	2	0.50	2	0.70	3

Fonte: Autor.

Ao final desse terceiro experimento, percebeu-se um uma melhora pouco expressiva nos valores de hiper-volume, entretanto o tempo de execução sofreu grande aumento devido

às alterações dos parâmetros.

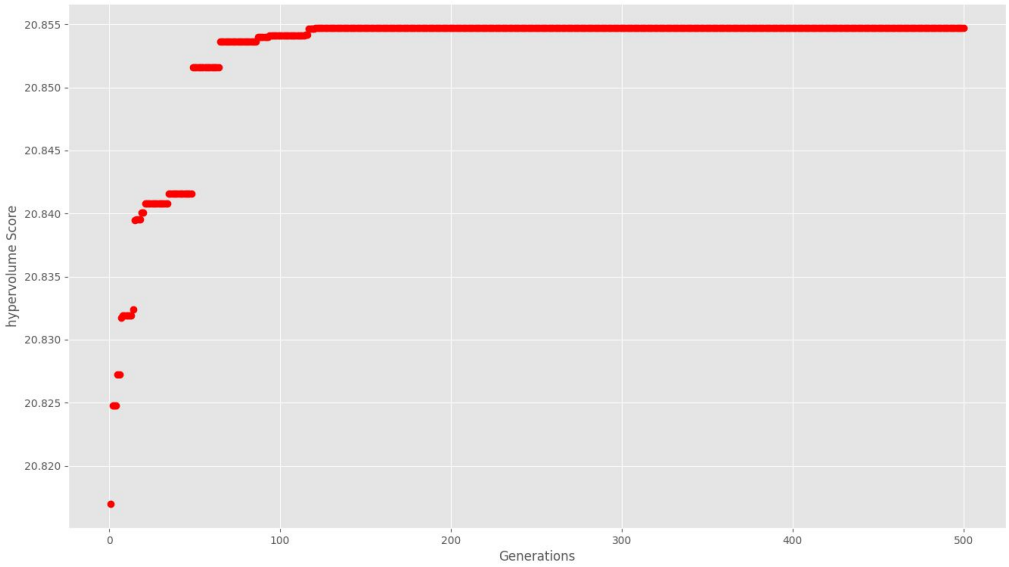
Dessa forma, foi decidido terminar os experimentos do projeto fatorial 2^k e definir os parâmetros utilizados, que podem ser observados na Subseção 4.5.4

4.5.4 Definição dos parâmetros para o NSGA-II

Após a realização dos três experimentos fatoriais, os parâmetros da quantidade de indivíduos por população, taxa de mutação e probabilidade de cruzamento puderam ser definidos. Entretanto, notou-se que o tempo de execução do algoritmo poderia estar tendendo o número de gerações sempre a diminuir, tendo uma maior influência neste parâmetro que o valor do hiper-volume, pois a cada geração adicionada, o tempo de execução aumentava bem mais que o valor de hiper-volume.

Desse modo, foi proposto um novo experimento, onde os valores dos parâmetros B, C e D seriam fixados conforme o resultado do último experimento do projeto fatorial 2^k, enquanto o valor do número de gerações seria extrapolado para verificar até onde o algoritmo apresentava evoluções.

Figura 17 – Evolução do algoritmo ao longo de 500 gerações



Fonte: Próprio autor

Dessa maneira, o número de gerações foi definido em 500, um número bem mais alto que os testados durante os experimentos anteriores, e com base também nos experimentos anteriores, o número de indivíduos por população foi definido em 120, a taxa de mutação em 40% e a probabilidade de cruzamento em 60%. Após o fim da execução do algoritmo, o resultado ilustrado na Figura 17 foi obtido.

Com base na figura acima, pode-se notar que a última evolução significativa do algoritmo fica em torno da 110ª geração. Dessa forma, o número de gerações final para o algoritmo, considerando também uma margem de erro, foi definido em 150 gerações.

Então, ao final de todos esses experimentos, a parametrização final do algoritmo ficou a seguinte:

- Número de gerações: 150
- Tamanho da população: 120
- Taxa de mutação: 40%
- Probabilidade de cruzamento: 60%

5 Resultados e Análise

Neste capítulo serão apresentados todos os dados obtidos a partir da execução do algoritmo confeccionado nos mapas selecionados, e também analisados os resultados obtidos.

5.1 Resultados com o mapa de Formiga-MG

O primeiro mapa em que o algoritmo foi aplicado foi o da cidade de Formiga-MG. Inicialmente, o algoritmo retornava apenas uma resposta, que correspondia ao indivíduo que otimizava em igual peso todas as três funções objetivo juntas, mas optou-se por retornar outras duas soluções também, que seriam o indivíduo que mais minimizou o tempo de coleta e o que mais minimizou o número de caminhões.

Todas essas soluções são retiradas da fronteira de Paretto, retornada pelo algoritmo ao fim da sua execução. Dessa forma, com mais opções de soluções, o usuário pode escolher aquela que melhor se adequa a realidade do seu problema. Lembrando que a fronteira possui outras diversas soluções, mas essas três foram filtradas para apresentar os resultados de uma maneira mais prática.

Abaixo seguem os dados obtidos durante a aplicação do algoritmo.

5.1.1 Solução minimizando todas as métricas de forma balanceada

Essa solução visa minimizar todas as métricas de forma balanceada (tempo de coleta, variação de altitude e número de caminhões). No Quadro 2 são apresentados os números encontrados.

Analisando o resultado acima, pode-se notar que o algoritmo conseguiu recolher todo o lixo da cidade, o que é um ótimo indicativo, e também utilizou de um número reduzido de caminhões, apenas 8, sendo que todos eles foram utilizados, percorrendo quilometragens bem próximas uma das outras, o que mostra que todos ele foram bem utilizados, sem que um transitasse muito mais que outro.

Além disso, o tempo total gasto pela coleta de lixo totalizou 19 horas, o que dividindo em escalas de 8 horas por dia, resulta em pouco mais de dois dias para a coleta ser feita na cidade, o que é um tempo plausível, visto que normalmente os caminhões coletores não passam diariamente pela mesma rua para coletar os resíduos em ambas as cidades utilizadas como cenário.

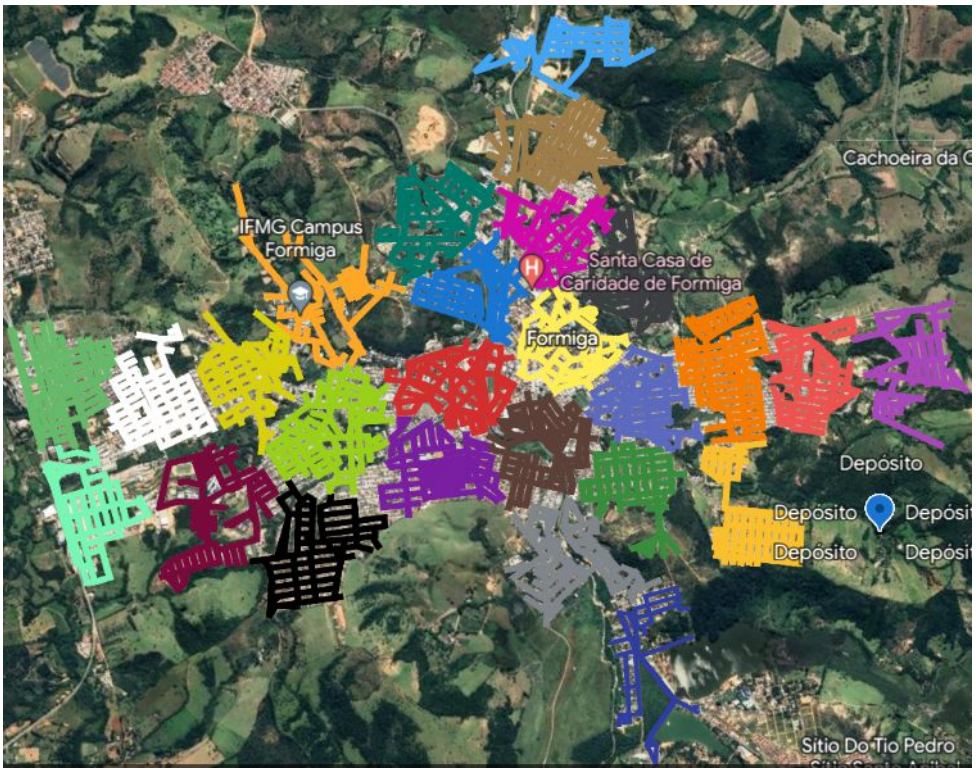
A variação de altitude obteve um valor de 3,89, que representa o somatório do

Quadro 2 – Solução minimizando as métricas de forma balanceada

```
*** RESULTADOS OBTIDOS DA COLETA ***
Quantidade de lixo recolhido (kg): 69007 (100%)
Número de caminhões utilizados: 8
Número de agrupamentos realizados pelo mapa: 26
Quilometragem total do percurso (km): 1145,29
Quilometragem individual de cada caminhão (km):
    Caminhão 0: 191,17      Caminhão 1: 170,79      Caminhão 2: 87,27
    Caminhão 3: 129,13      Caminhão 4: 155,07      Caminhão 5: 149,82
    Caminhão 6: 124,18      Caminhão 7: 137,86
Maior tempo gasto por um caminhão (horas):  19
Menor tempo gasto por um caminhão (horas):   8
Somatório de variação de altitude da rota:  3,89
```

cálculo realizado quando os caminhões vão de um ponto ao outro, como foi explicado na Subseção 4.3.7, o que indica que o algoritmo tomou boas decisões e evitou passar por locais muito íngremes muitas vezes.

Figura 18 – Rotas com todas as métricas minimizadas - Formiga-MG



Fonte: Próprio autor

A Figura 18 mostra as rotas no site *Google Earth*, cada rota está destacada com uma cor diferente, enquanto a Figura 19 mostra mais detalhadamente uma rota em um

determinado bairro.

Figura 19 – Detalhamento de rota - Formiga-MG

Fonte: Próprio autor

5.1.2 Solução minimizando a métrica do número de caminhões

Nesta solução, a métrica da minimização do número de caminhões é atendida com uma maior prioridade pela heurística, diminuindo os gastos com os veículos, mas, em contrapartida, aumenta o tempo de coleta e em alguns casos a quilometragem rodada.

Essa pode ser uma boa solução caso o número de caminhões disponíveis pela prefeitura seja muito limitado, tendo em mente que o preço de um caminhão coletor de lixo provavelmente é alto. No Quadro 3 seguem os resultados para esse cenário.

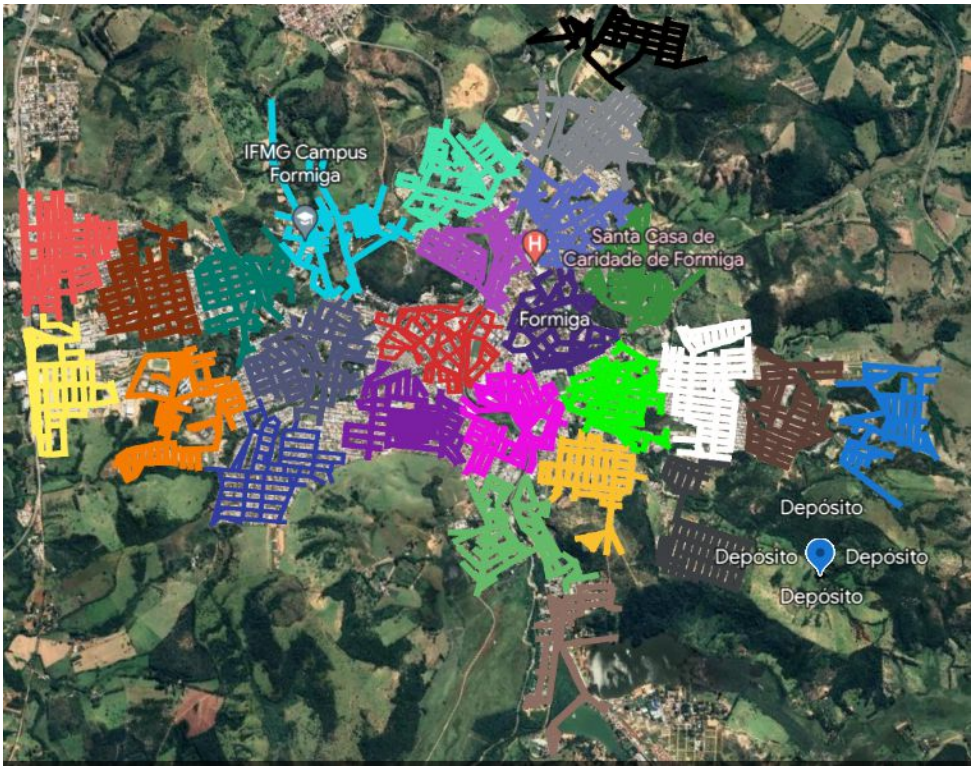
Quadro 3 – Solução minimizando a métrica do número de caminhões

```
*** RESULTADOS OBTIDOS DA COLETA ***
Quantidade de lixo recolhido (kg): 69007 (100%)
Número de caminhões utilizados: 5
Número de agrupamentos realizados pelo mapa: 26
Quilometragem total do percurso (km): 1144,61
Quilometragem individual de cada caminhão (km):
    Caminhão 0: 266,35      Caminhão 1: 228,04      Caminhão 2: 202,92
    Caminhão 3: 216,77      Caminhão 4: 230,53
Maior tempo gasto por um caminhão(horas):  26
Menor tempo gasto por um caminhão(horas):  20
Somatório de variação de altitude da rota:  3,89
```

Tendo em vista os resultados apresentados, pode-se notar que o número de caminhões teve uma queda, de 8 para apenas 5, e como no resultado anterior, todos eles foram

bem utilizados, rodando quilometragens bem próximas umas das outras. A Figura 20 mostra o mapa das rotas propostas por esse cenário.

Figura 20 – Rotas com foco na minimização no número de caminhões - Formiga-MG



Fonte: Próprio autor

Em decorrência do menor número de veículos, o tempo de coleta aumentou, já que são menos caminhões para lidar com a mesma quantidade de lixo, resultando em 26 horas de coleta na cidade, o que ainda é um número plausível, pois tendo em vista novamente uma jornada de 8 horas por dia, em aproximadamente 3 dias a coleta da cidade seria realizada por completo. Neste caso, a quilometragem final ainda foi reduzida em um quilômetro. A variação de altitude se manteve a mesma do resultado anterior.

5.1.3 Solução minimizando a métrica do tempo de coleta

No Quadro 4, exibimos a solução de um indivíduo da fronteira de Paretto cuja métrica de maior prioridade é a minimização do tempo gasto na coleta.

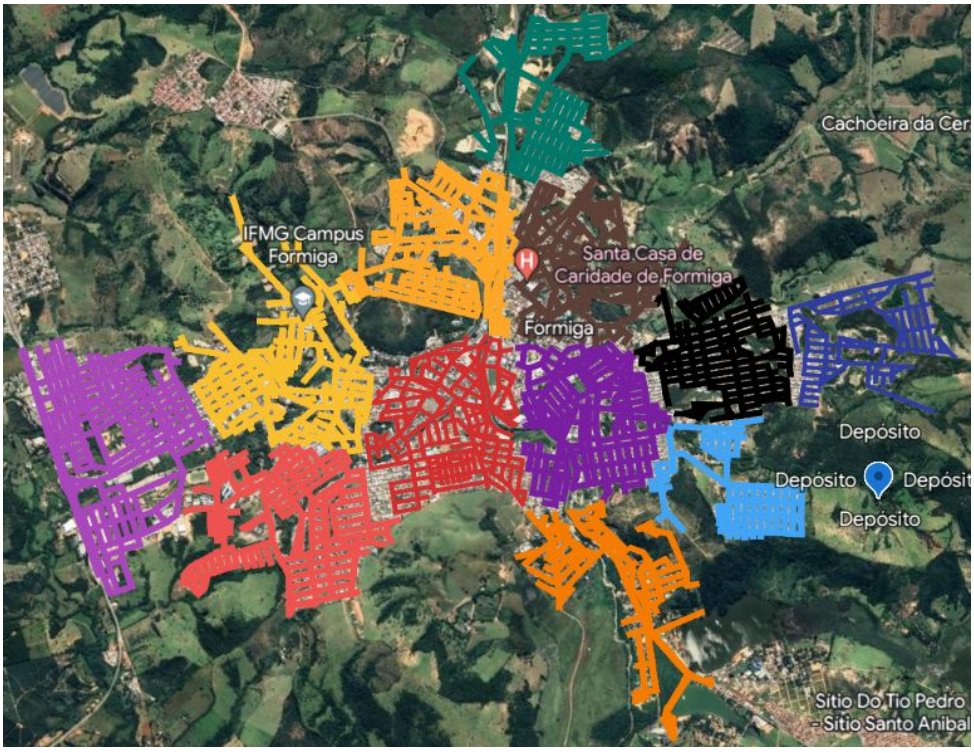
Analisando os dados, pode-se observar que o tempo da coleta sofreu uma queda significativa, totalizando apenas 9 horas, o que significa que a coleta poderia ser realizada quase que diariamente pela cidade. Em contrapartida, o número de caminhões obteve um aumento significativo, sendo necessários 12 veículos, que com certeza elevaria os custos da coleta. Com mais caminhões, também foi possível diminuir a quantidade total de quilômetros rodados, que caiu para apenas 855 quilômetros. A variação de altitude sofreu

Quadro 4 – Solução minimizando a métrica do tempo de coleta.

```
*** RESULTADOS OBTIDOS DA COLETA ***
Quantidade de lixo recolhido (kg): 69007 (100%)
Número de caminhões utilizados: 12
Número de agrupamentos realizados pelo mapa: 12
Quilometragem total do percurso (km): 855,96
Quilometragem individual de cada caminhão (km):
    Caminhão 0: 78,58       Caminhão 1:  63,66      Caminhão 2:  74,93
    Caminhão 3: 89,78       Caminhão 4:  85,74      Caminhão 5:  72,80
    Caminhão 6: 76,99       Caminhão 7:  93,08      Caminhão 8:  33,49
    Caminhão 9: 72,67       Caminhão 10: 50,08      Caminhão 11: 64,16
Maior tempo gasto por um caminhão(horas):  9
Menor tempo gasto por um caminhão(horas):  3
Somatório de variação de altitude da rota: 3,93
```

uma leve piora, indicando que o caminhão coletor precisou passar por locais muito íngremes mais vezes. A Figura 21 mostra o mapa com as rotas sugeridas para a cidade.

Figura 21 – Rotas com foco na minimização no tempo de coleta - Formiga-MG



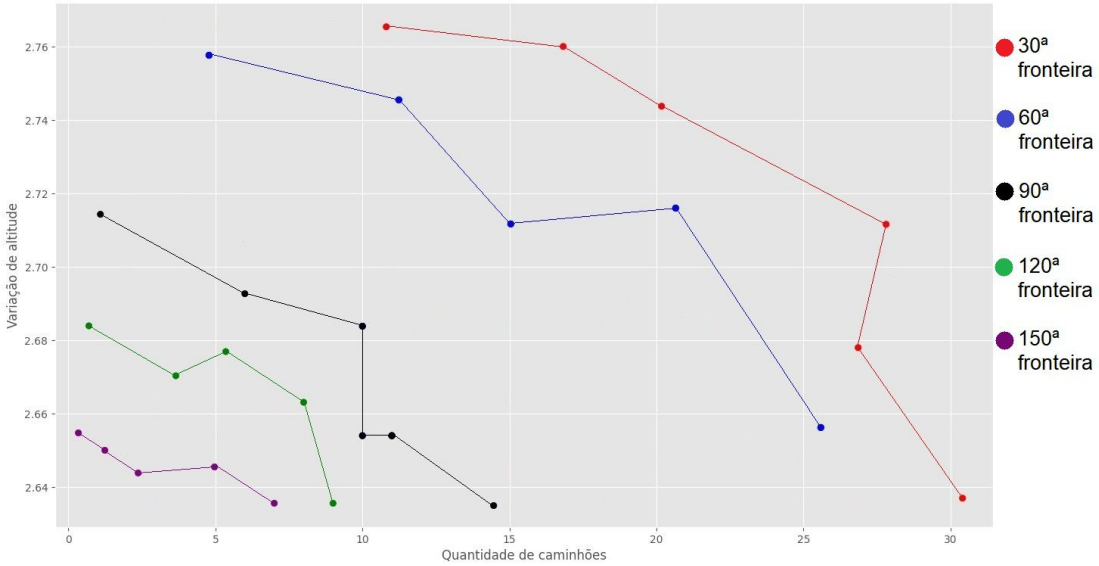
Fonte: Próprio autor

Essa solução é uma boa opção para cidades que possuem uma frota de veículos e uma receita maior, onde é necessário recolher rapidamente todo o lixo gerado pela cidade.

5.1.4 Evolução das fronteiras de Paretto

Ao decorrer das gerações foram exibidas algumas fronteiras para demonstrar a evolução dos resultados do algoritmo, na imagem abaixo é apresentada essa evolução do algoritmo no mapa de Formiga. A cada 30 gerações, de um total de 150, a melhor fronteira era exibida, totalizando em 5 fronteiras armazenadas.

Figura 22 – Evolução das fronteiras de Paretto - Formiga-MG



Fonte: Próprio autor

Na Figura 22 cada ponto representa um indivíduo da fronteira de Paretto, neste caso são utilizados como base seus valores do número de caminhões e da quilometragem total, sendo que cada fronteira é separada por uma cor diferente.

5.2 Resultados com o mapa de Lagoa da Prata-MG

Após a realização dos experimentos citados nas seções acima, optou-se em escolher alguma outra cidade para verificar se a validade do algoritmo se aplicava também em outros mapas. Dessa forma, foi escolhida a cidade de Lagoa da Prata-MG, sendo uma cidade próxima e um pouco menor que a cidade de Formiga.

A obtenção dos dados do mapa foi feita da mesma forma: realizando uma consulta na ferramenta *overpass-turbo* e baixando os dados no formato OSM, que mais adiante foi formatado pelo algoritmo.

Para a execução do algoritmo no novo mapa foram utilizados os mesmos parâmetros de número de gerações, tamanho da população, taxa de mutação e possibilidade de

cruzamento do mapa de Formiga. Isso foi feito evitar a realização de um novo projeto fatorial 2^k, que demanda um certo tempo para ser feito, e também porque as duas cidades são similares e se encontram na mesma região (centro-oeste de Minas Gerais). Porém, para executar o algoritmo para uma cidade de uma região diferente ou então com características muito discrepantes, recomenda-se que seja realizado um novo projeto fatorial para os parâmetros serem melhor calibrados para aquele cenário.

Assim como no mapa de Formiga, abaixo são apresentados três diferentes resultados obtidos após a execução da aplicação no mapa de Lagoa da Prata.

5.2.1 Solução minimizando todas as métricas de forma balanceada

Analisando os resultados do Quadro 5, pode-se observar que todo o lixo da cidade foi recolhido em um tempo de aproximadamente 16 horas, que pode ser facilmente alocado em dois dias de coleta em uma jornada de 8 horas diárias. A coleta demandou 8 caminhões para ser realizada, e apenas os veículos 5, 6 e 7 rodaram menos que os demais, mas ainda assim fizeram boas quilometragens.

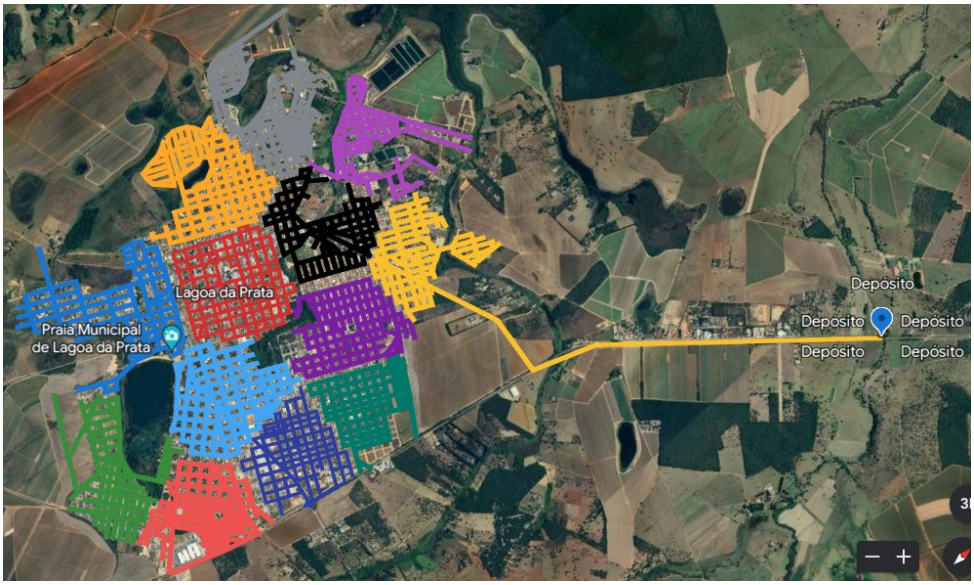
Quadro 5 – Solução minimizando todas as métricas balanceadamente

```
*** RESULTADOS OBTIDOS DA COLETA ***
Quantidade de lixo recolhido (kg): 70802 (100%)
Número de caminhões utilizados: 8
Número de agrupamentos realizados pelo mapa: 13
Quilometragem total do percurso (km): 979,99
Quilometragem individual de cada caminhão (km):
    Caminhão 0: 135,44      Caminhão 1: 133,16      Caminhão 2: 136,39
    Caminhão 3: 164,76      Caminhão 4: 161,96      Caminhão 5: 77,12
    Caminhão 6: 74,67       Caminhão 7: 96,50
Maior tempo gasto por um caminhão(horas): 16
Menor tempo gasto por um caminhão(horas):  7
Somatório de variação de altitude da rota: 2,69
```

Por ser uma cidade um pouco menor e possuir uma geografia menos acidentada que Formiga, a variação de altitude foi de apenas 2,69, indicando também que o algoritmo tomou boas decisões e evitou subir e descer desnecessariamente em ruas íngremes.

Na Figura 23 são exibidas as rotas realizadas pelo algoritmo.

Figura 23 – Rotas com todas as métricas minimizadas - Lagoa da Prata-MG



Fonte: Próprio autor

5.2.2 Solução minimizando a métrica do número de caminhões

Nesta solução, novamente todo o lixo da cidade foi coletado, e o número de caminhões sofreu uma leve queda, de apenas um caminhão. Tais resultados são exibidos no Quadro 6.

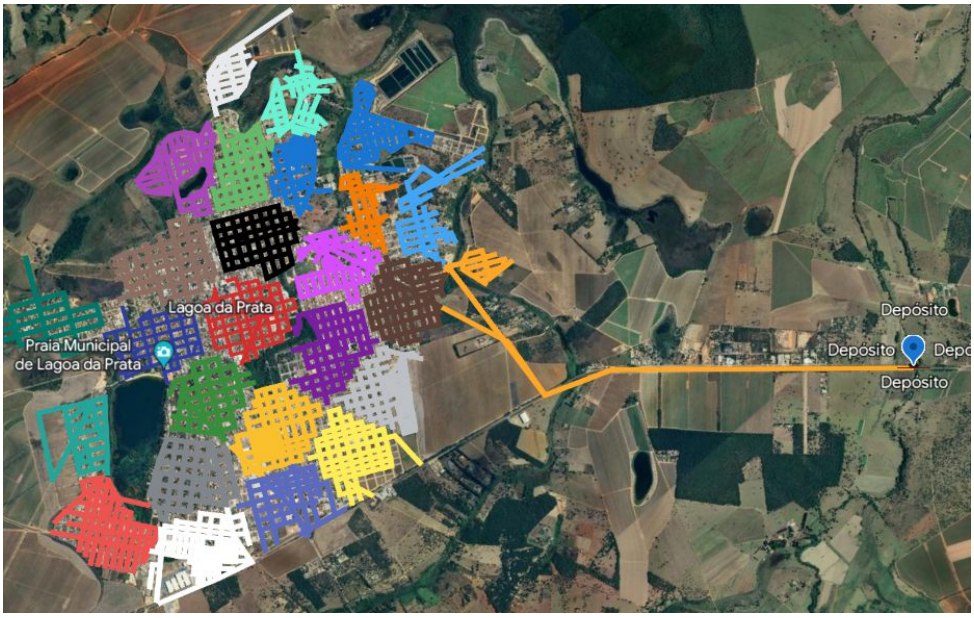
Quadro 6 – Solução minimizando a métrica do número de caminhões

```
*** RESULTADOS OBTIDOS DA COLETA ***
Quantidade de lixo recolhido (kg): 70802 (100%)
Número de caminhões utilizados: 7
Número de agrupamentos realizados pelo mapa: 26
Quilometragem total do percurso (km): 1521,48
Quilometragem individual de cada caminhão (km):
    Caminhão 0: 228,37      Caminhão 1: 220,81      Caminhão 2: 216,19
    Caminhão 3: 261,71      Caminhão 4: 236,27      Caminhão 5: 192,78
    Caminhão 6: 165,35
Maior tempo gasto por um caminhão(horas): 26
Menor tempo gasto por um caminhão(horas): 16
Somatório de variação de altitude da rota: 2,65
```

Em contrapartida, o número de horas necessárias para a realização da coleta aumentou em 10 horas, sendo necessária aplicar a coleta em pouco mais de 3 dias, considerando uma jornada de 8 horas por dia. Neste caso apenas o caminhão 6 obteve uma quilometragem menor que os demais, mas ainda assim foi bem aproveitado.

A quilometragem final sofreu um grande aumento, passando para de 979,99 km para 1521,48 km. Enquanto isso, a variação de altitude sofreu uma leve queda. Na Figura 24 seguem as rotas deste cenário.

Figura 24 – Rotas com foco na minimização no número de caminhões - Lagoa da Prata-MG



Fonte: Próprio autor

5.2.3 Solução minimizando a métrica do tempo de coleta

O resultado da minimização focada no tempo de coleta é exibido no Quadro 7.

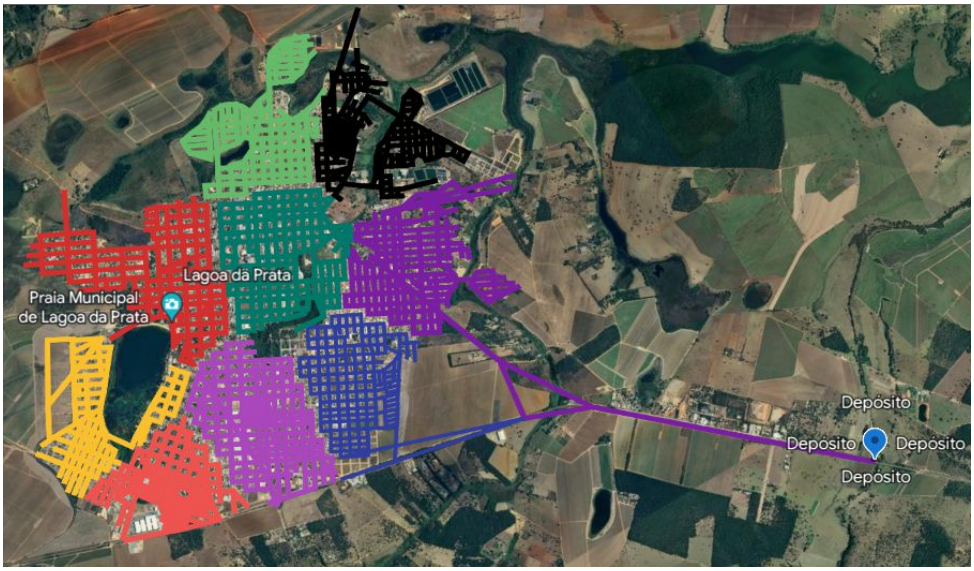
Quadro 7 – Solução minimizando a métrica do tempo de coleta

```
*** RESULTADOS OBTIDOS DA COLETA ***
Quantidade de lixo recolhido (kg): 70802 (100\%)
Número de caminhões utilizados: 9
Número de agrupamentos realizados pelo mapa: 9
Quilometragem total do percurso (km): 872,15
Quilometragem individual de cada caminhão (km):
    Caminhão 0: 88,71       Caminhão 1: 121,07      Caminhão 2: 122,95
    Caminhão 3: 81,70       Caminhão 4: 85,46       Caminhão 5: 88,52
    Caminhão 6: 89,75       Caminhão 7: 89,76       Caminhão 8: 104,22
Maior tempo gasto por um caminhão(horas): 12
Menor tempo gasto por um caminhão(horas):  8
Somatório de variação de altitude da rota: 2,75
```

Apesar do número de caminhões utilizados ser o maior entre todas as soluções (utilizando 9 veículos) e a variação de altitude ter sofrido um leve aumento, a quilometragem

final foi a menor entre todas as soluções, totalizando em apenas 872,15 km.

Figura 25 – Rotas com foco na minimização no tempo de coleta - Lagoa da Prata-MG



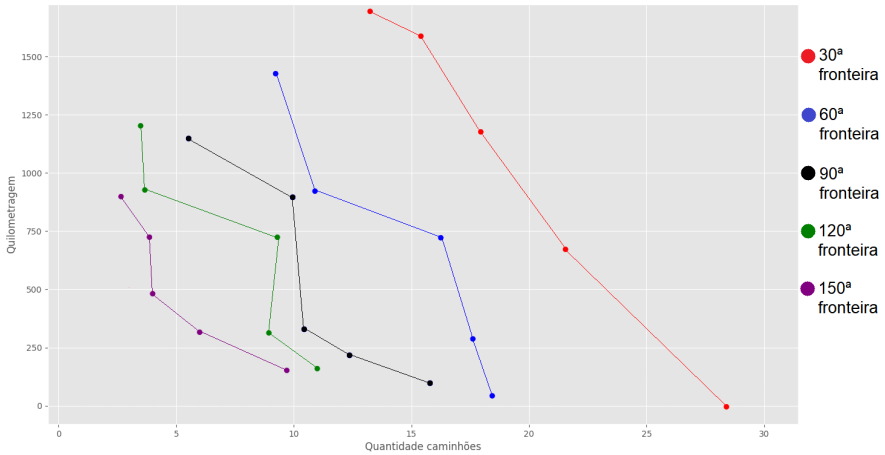
Fonte: Próprio autor

O tempo total de coleta, de aproximadamente 12 horas, poderia ser escalado facilmente em dois dias de coleta. Na Figura 25 seguem as rotas realizadas pelos veículos.

5.2.4 Evolução das fronteiras de Paretto

Ao aplicar o algoritmo na cidade de Lagoa da Prata, também foram armazenadas as fronteiras de Paretto, da mesma forma realizada no mapa de Formiga.

Figura 26 – Evolução das fronteiras de Paretto - Lagoa da Prata-MG



Fonte: Próprio autor

A Figura 26 ilustra essa evolução, na qual foram utilizados como base os valores da quilometragem e da quantidade de caminhões de cada indivíduo, para fins de ilustração.

6 Considerações Finais

Este trabalho teve como objetivo apresentar um protótipo de software capaz de realizar o roteamento dos caminhões coletores de lixo de determinada cidade, apresentando as rotas que cada caminhão deve fazer, o tempo de coleta, a quilometragem total e individual de cada caminhão, além da variação de altitude durante a realização da coleta. Desta forma, foram abordados tanto o problema do Roteamento de Veículos Capacitados (PRVC) quanto o problema do Carteiro Chinês Com Vento (PCCV). Nos experimentos para análise dos resultados, foram utilizadas como cenário de exemplo as cidades de Formiga/MG e de Lagoa da Prata/MG.

A aplicação desenvolvida é capaz de: receber um arquivo no formato OSM que descreve o mapa rodoviário de uma cidade, simular o mapa da cidade por meio de um grafo utilizando dados reais de metragem das ruas e altitude dos pontos, utilizar o algoritmo *k-means* para a clusterização dos pontos do mapa, aplicar a meta-heurística NSGA-II para realizar a parametrização de como a coleta será realizada na cidade e, ao terminar sua execução, apresentar diferentes resultados para o usuário poder escolher qual melhor se encaixa a sua realidade. Todo o processo de desenvolvimento está registrado no github[1] do autor, além dos códigos e arquivos utilizados.

Atrás dos melhores resultados e durante a busca pelo espaço de soluções, foi aplicado o projeto fatorial 2^k, para a calibragem dos parâmetros do número de gerações, tamanho da população, taxa de mutação e probabilidade de cruzamento utilizados pelo NSGA-II, permitindo que a meta-heurística explore melhor o espaço de soluções.

Apesar de não termos utilizado os dados das rotas efetivamente realizadas pelas prefeituras das cidades analisadas (pois não conseguimos obter tais dados), o software desenvolvido durante a confecção deste trabalho é capaz de, a partir de um mapa de uma determinada cidade obtido através do *OpenStreetMap*, gerar, de forma otimizada, trajetos para os veículos realizarem a coleta dos resíduos da cidade e armazená-los no local destinado para tal fim, utilizando-se da menor rota possível calculada.

Vale ressaltar que, como cada veículo possui uma capacidade máxima de armazenamento de lixo, nosso algoritmo experimenta diversas formas de agrupar os pontos de coleta da cidade para que essa capacidade do caminhão seja utilizada de forma mais eficiente.

Sobre a redução de custos da coleta, apesar dos dados utilizados pelas prefeituras não terem sido obtidos, tais como: o valor exato de um caminhão de lixo, a quantidade média de combustível gasta por cada veículo, as rotas feitas pela cidade por cada caminhão, etc., as soluções geradas pelo algoritmo genético proporcionam diversas respostas distintas

[1] Disponível em; <https://github.com/Luc4s99/Trabalho-de-Conclusao-de-Curso>

para que os responsáveis pela administração da coleta escolham aquela que melhor atende as demandas da cidade. Por exemplo: uma cidade que possui poucos veículos coletores, pode optar por uma resposta que se utiliza de menos caminhões, mas que demande um maior tempo para concluir a coleta, caso isso seja economicamente interessante para aquela cidade.

Por fim, considerando que o algoritmo testa quantidades diferentes de agrupamentos para os caminhões fazerem uma ou mais viagens conforme o necessário, essas quantidades de agrupamentos também visam reduzir a distância a ser percorrida coletivamente pelos caminhões, considerando que eles irão sair de um determinado ponto, realizar a coleta do lixo e armazenar todos os resíduos coletados em um determinado local, como, por exemplo, um lixão ou um aterro sanitário.

6.1 Principais contribuições

Abaixo são listadas as principais contribuições realizadas durante este trabalho:

- Aplicabilidade do software desenvolvido neste trabalho em cenários de outras cidades ou regiões, com a recomendação de recalibrar os parâmetros do NSGA-II para explorar adequadamente um maior espaço de soluções.
- Modelagem do mapa da cidade como dois grafos: (i) grafo completo, contendo todos os segmentos de rua para o cálculo preciso da distância percorrida e inclinação; (ii) grafo simplificado contendo um segmento por rua, focando nas interseções entre ruas, o que reduziu a "explosão combinatória" de possibilidades e viabilizou obter mais rapidamente melhores soluções.
- Descrição de informações necessárias para a simulação da coleta de lixo no mapa de uma cidade. Por exemplo, implementação de um cálculo dinâmico da quantidade de lixo gerado em cada rua, em função da extensão da rua e tamanho "padrão" de lote, pela quantidade média de lixo gerado por uma família brasileira.
- Cache de clusterização: por ser uma operação pesada, porém necessária a cada nova geração de indivíduos, utilizamos para os clusters gerados pelo k-Means a estratégia de "memoização" tipicamente utilizada em programação dinâmica.
- Disponibilização[2] de todo o código-fonte desenvolvido e dos dados de entrada e saída, para viabilizar a replicação dos resultados obtidos neste trabalho.
- Complementação da documentação do código-fonte de Fernandes (2019), para obtenção do mapa via OverpassTurbo/OpenStreetMaps e geração do grafo representando as vias da cidade.

[2] Disponível em; <https://github.com/Luc4s99/Trabalho-de-Conclusao-de-Curso>

- Possibilidade de adaptação do código base desenvolvido neste trabalho para ser aplicado em outros problemas de otimização, utilizando-se de novos parâmetros e/ou abordagens.

6.2 Trabalhos futuros

No decorrer do trabalho, diversas ideias surgiram para complementar o algoritmo, trazendo respostas mais precisas e condizentes com a realidade. Porém, devido à limitação de tempo para realização deste trabalho, à complexidade envolvida ou algum outro motivo inviabilizante, tais ideias ainda não foram implementadas. Assim, listamos abaixo algumas desses ideias, como possibilidades de continuidade deste trabalho:

- Adicionar a condição de janelas de tempo ao problema abordado, fazendo com que cada ponto de coleta de lixo tenha um intervalo de tempo específico para ser recolhido.
- Obter informações sobre a mão e contramão das ruas, levando isso em consideração ao gerar as rotas.
- Desenvolver uma rotina que, dada uma cidade específica, busque e realize a entrada dos dados automaticamente, através da biblioteca OSMnx, por exemplo, sem necessidade de interferências manuais, atualizando também alguns parâmetros conforme o porte da cidade, como, por exemplo, o número de agrupamentos.
- Buscar alguma forma de tornar a apresentação dos resultados mais interativa para o usuário (apesar de serem geradas imagens dos agrupamentos, salvas as rotas em arquivo importável no Google Maps), demonstrando com ainda mais detalhes as rotas e os números encontrados.
- Levantar junto a prefeitura da cidade de Formiga, de Lagoa da Prata ou então de alguma outra cidade, dados reais sobre as condições de coleta de lixo, como: número de veículos coletores da cidade, capacidade dos veículos, quantidade de lixo produzida pela cidade, etc. Aplicar tais dados como um *benchmark*, visando adaptar o algoritmo para retornar resultados mais próximos da realidade e validar a otimização da solução proposta.
- Aplicar diferentes meta-heurísticas para a resolução do problema, comparando seus resultados com o *NSGA-II* utilizado neste trabalho.
- Explorar o *NSGA-II*, ou qualquer outra meta-heurística, em diferentes abordagens para resolução do problema, como utilizá-lo na confecção das rotas, na definição da ordem de visitação dos agrupamentos gerados no mapa da cidade, na decisão sobre o tamanho dos agrupamentos e demais decisões.

- Utilizar da computação paralela para realizar a avaliação de cada subgrafo ou indivíduo de forma paralelizada, em uma CPU diferente ou em GPU. Realizar também, um *cache* dos agrupamentos convertidos para grafos eulerianos, otimizando o tempo de execução do algoritmo.

Referências

BANDYOPADHYAY, S.; CHAKRABORTY, R.; MAULIK, U. Priority based dominance: A new measure in multiobjective optimization. *Information Sciences*, v. 305, p. 97–109, 2015. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025515000560>. Citado 2 vezes nas páginas 24 e 25.

CUSTóDIO, A.; EMMERICH, M.; MADEIRA, J. Recent developments in derivative-free multiobjective optimisation. *Computational Technology Reviews*, Civil-Comp, Ltd., v. 5, p. 1–30, set. 2012. ISSN 2044-8430. Disponível em: <http://dx.doi.org/10.4203/ctr.5.1>. Citado na página 26.

CâNDIDO, J. C. d. M. *UM ANALISADOR DE DESEMPENHO PARA SIMULADOR DE REDE VEICULAR CANBUS*. 108 f. Monografia (Bacharel em Ciência da Computação) — Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Formiga, Minas Gerais, 2019. Citado 3 vezes nas páginas 23, 24 e 32.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. In: . Institute for Operations Research and the Management Sciences (INFORMS), 1959. v. 6, n. 1, p. 80–91. Disponível em: <https://doi.org/10.1287/mnsc.6.1.80>. Citado na página 19.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, Institute of Electrical and Electronics Engineers (IEEE), v. 6, n. 2, p. 182–197, abr. 2002. Disponível em: <https://doi.org/10.1109/4235.996017>. Citado 2 vezes nas páginas 22 e 25.

DERECI, U.; KARABEKMEZ, M. E. The applications of multiple route optimization heuristics and meta-heuristic algorithms to solid waste transportation: A case study in turkey. *Decision Analytics Journal*, v. 4, p. 100113, 2022. ISSN 2772-6622. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2772662222000480>. Citado na página 27.

DETOFENO, T. C.; STEINER, M. T. A. Optimizing routes for the collection of urban solid waste: A case study for the city of joinville, state of santa catarina. *Iberoamerican Journal of Industrial Engineering*, v. 2, p. 124–136, 2010. ISSN 2175-8018. Disponível em: <http://incubadora.periodicos.ufsc.br/index.php/IJIE/article/download/160/pdf_71>. Citado na página 18.

EISELT, H. A.; GENDREAU, M.; LAPORTE, G. Arc routing problems, part i: The chinese postman problem. *Operations Research*, Institute for Operations Research and the Management Sciences (INFORMS), v. 43, n. 2, p. 231–242, abr. 1995. Disponível em: <https://doi.org/10.1287/opre.43.2.231>. Citado 2 vezes nas páginas 19 e 20.

FERNANDES, S. R. D. *SOFTWARE PARA AUTOMATIZAÇÃO DO PROJETO DE REDES ÓPTICAS PASSIVAS (PON)*. 57 f. Monografia (Bacharel em Ciência da Computação) — IFMG Campus Formiga, Formiga, Minas Gerais, 2019. Citado 3 vezes nas páginas 31, 41 e 65.

G1. *Prefeitura de SP gasta R$ 56 milhões por mês com recolhimento de lixo*. 2012. Disponível em: <http://g1.globo.com/sao-paulo/sao-paulo-mais-limpa/noticia/2012/05/prefeitura-de-sp-gasta-r-56-milhoes-por-mes-com-recolhimento-de-lixo.html>. Citado na página 16.

HARTIGAN, J. A.; WONG, M. A. Algorithm AS 136: A k-means clustering algorithm. *Applied Statistics*, JSTOR, v. 28, n. 1, p. 100, 1979. Disponível em: <https://doi.org/10.2307/2346830>. Citado na página 21.

IBGE. *Pesquisa de orçamentos familiares 2017-2018 : primeiros resultados*. [S.l.: s.n.], 2019. v. 1. Citado na página 41.

IBGE. *Formiga*. 2023. Disponível em: <https://cidades.ibge.gov.br/brasil/mg/formiga/panorama>. Citado na página 27.

IBGE. *Lagoa da Prata*. 2023. Disponível em: <https://cidades.ibge.gov.br/brasil/mg/lagoa-da-prata/panorama>. Citado na página 27.

JAIN, A. K. Data clustering: 50 years beyond k-means. In: *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2008. p. 3–4. Disponível em: <https://doi.org/10.1007/978-3-540-87479-9_3>. Citado na página 21.

LAPORTE, G. The vehicle routing problem: An overview of exact and approximate algorithms. In: . Elsevier BV, 1992. v. 59, n. 3, p. 345–358. Disponível em: <https://doi.org/10.1016/0377-2217(92)90192-c>. Citado na página 19.

LONGEN, A. *O Que é GitHub e Como Usá-lo*. 2022. Disponível em: <https://www.hostinger.com.br/tutoriais/o-que-github>. Citado na página 31.

MALAQUIAS, N. G. L. *Uso dos Algoritmos Genéticos para a Otimização de Rotas de Distribuição*. 97 f. Monografia (Mestrado em Ciências) — Universidade Federal de Uberlândia, Uberlândia, Minas Gerais, 2006. Citado 2 vezes nas páginas 27 e 32.

MORO, M. F. *O PROBLEMA DO CARTEIRO CHINÊS APLICADO NA OTIMIZAÇÃO DE ROTAS USADAS NA COLETA DE LIXO RECICLÁVEL: UM ESTUDO DE CASO*. 59 f. Monografia (Bacharel em Engenharia de Produção) — Universidade Tecnológica Federal do Paraná, Medianeira, Paraná, 2014. Citado na página 27.

OLIVEIRA, T. B. *Clusterização de dados utilizando técnicas de redes complexas e computação boinspirada*. 95 f. Monografia (Mestre em Ciência da Computação) — USP Campus São Carlos, São Carlos, São Paulo, 2008. Citado na página 21.

ORE, O.; WILSON, R. J. Connected graphs. In: _____. *Graphs and Their Uses*. [S.l.]: Mathematical Association of America, 1990. (Anneli Lax New Mathematical Library), cap. Connected Graphs, p. 24–36. Citado na página 21.

OSM-FOUNDATION. *OpenStreetMap Foundation*. 2021. Disponível em: <https://wiki.osmfoundation.org/wiki/Main_Page>. Citado na página 28.

PINTO, T. O. R. *Um escalonador de processos adaptativo baseado no round-robin e otimizado com NSGA-II*. 51 f. Monografia (Bacharel em Ciência da Computação) — IFMG Campus Formiga, Formiga, Minas Gerais, 2021. Citado 4 vezes nas páginas 23, 25, 26 e 32.

PIRES, Y.; OLIVEIRA, N. *Aumento da produção de lixo no Brasil requer ação coordenada entre governos e cooperativas de catadores*. 2021. Disponível em: <https://www12.senado.leg.br/noticias/infomaterias/2021/06/aumento-da-producao-de-lixo-no-brasil-requer-acao-coordenada-entre-governos-e-cooperativas-de-cata
Citado na página 41.

PYTHON-SOFTWARE-FOUNDATION. *What is Python? Executive Summary*. 2022. Disponível em: <https://www.python.org/doc/essays/blurb/>. Citado na página 30.

ROMAIS, R. *Aplicação Algorítimo Genético*. 2012. Disponível em: <https://pt.slideshare.net/RodrigoRomais/aplicao-algortimo-gentico>. Citado na página 19.

SANTO, S. de Estado da Saúde do E. *Enchentes e alagamentos podem causar doenças transmitidas pela água*. 2013. Disponível em: <https://saude.es.gov.br/enchentes-e-alagamentos-podem-causar-doencas>. Citado na página 16.

SCABURI, A. *OTIMIZAÇÃO NA LOGÍSTICA DE DISTRIBUIÇÃO DE MERCADORIAS POR MEIO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS: UM ESTUDO DE CASO PARA ENTREGA DE JORNAIS*. 123 f. Monografia (Mestrado em Engenharia de Produção) — Universidade Federal do Paraná, Curitiba, Paraná, 2020. Citado 3 vezes nas páginas 22, 27 e 32.

SOARES, M. F. V. *Otimização da quantidade e do posicionamento de pontos de acesso utilizando NSGA-II*. 60 f. Monografia (Bacharel em Ciência da Computação) — IFMG Campus Formiga, Formiga, Minas Gerais, 2021. Citado 2 vezes nas páginas 32 e 49.

WALDEMAN, M. Não há planeta para tanto lixo. In: . [s.n.], 2011. Disponível em: <http://www.mw.pro.br/mw/geo_pos_doc_planeta.pdf>. Citado na página 16.

WHILE, L. et al. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation*, v. 10, n. 1, p. 29–38, 2006. Citado na página 26.

WHITLEY, D. Handbook of metaheuristics. In: _____. Switzerland: Springer, International Series in Operations Research Management Science, 2019. cap. Next Generation Genetic Algorithms: A User's Guide and Tutorial, p. 244–274. Disponível em: <https://doi.org/10.1007/978-3-319-91086-4_8>. Citado na página 22.

WIKIPEDIA. *Google Earth*. 2023. Disponível em: <https://pt.wikipedia.org/wiki/Google_Earth>. Citado na página 29.

WWF. *Brasil é o 4º país do mundo que mais gera lixo plástico*. 2019. Disponível em: <https://www.wwf.org.br/?70222/Brasil-e-o-4-pais-do-mundo-que-mais-gera-lixo-plastico>. Citado na página 7.

ZITZLER, E.; THIELE, L. Multiobjective optimization using evolutionary algorithms — a comparative case study. In: EIBEN, A. E. et al. (Ed.). *Parallel Problem Solving from Nature — PPSN V*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 292–301. Citado na página 26.

ZITZLER, E. et al. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, v. 7, n. 2, p. 117–132, 2003. Citado na página 26.