Weverton Rodrigues Arantes

Uma ferramenta de um aplicativo para propor e exibir rotas para a coleta de resíduos sólidos em ambientes urbanos

Formiga - MG

2023

Weverton Rodrigues Arantes

Uma ferramenta de um aplicativo para propor e exibir rotas para a coleta de resíduos sólidos em ambientes urbanos

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.

Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais

Campus Formiga

Ciência da Computação

Orientador: Mário Luiz Rodrigues Oliveira

Formiga - MG

2023

Arantes, Weverton Rodrigues

A662u Uma ferramenta de um aplicativo para propor e exibir rotas para a coleta de resíduos sólidos em ambientes urbanos / Weverton Rodrigues Arantes – Formiga : IFMG, 2023.

56p. : il. color.

Orientador: Prof. Mário Luiz Rodrigues Oliveira

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) Instituto Federal de Educação,Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Coleta de lixo. 2. Metaheurísticas. 3. CVRP. 4. Algoritmo genético. 5. Colônia de formiga. 6. Enxame de partículas. I. Oliveira, Mário Luiz Rodrigues. II. Título.

CDD 004

Ficha catalográfica elaborada pela Bibliotecária Lívia Renata Santos- CRB/6-2561

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

WÉVERTON RODRIGUES ARANTES

Uma ferramenta para propor e exibir rotas para a coleta de resíduos sólidos em ambientes urbanos

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: 22 de novembro de 2023.

BANCA EXAMINADORA

Prof. Mário Luiz Rodrigues Oliveira (orientador, IFMG)

Prof.º Fernando Paim Lima (IFMG)

Prof.º Walace de Almeida Rodrigues (IFMG)

Documento assinado eletronicamente por **Mario Luiz Rodrigues Oliveira**, **Professor**, em 22/11/2023, às 20:44, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fernando Paim Lima**, **Professor**, em 22/11/2023, às 20:47, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Walace de Almeida Rodrigues**, **Professor**, em 22/11/2023, às 20:54, 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 **1736810** e o código CRC **FB448486**.

Este trabalho é dedicado ao meu pai, minha mãe e amigos
que me ajudaram nesse processo

Agradecimentos

Primeiramente gostaria de agradecer meus pais por tudo que fizeram, aos meus professores pela paciência que tiveram, ao amigos pelos ajuda provida durante todo esse tempo.

"Mesmo desacreditado e ignorado por todos, não posso desistir, pois para mim, vencer é nunca desistir." (Albert Einstein)

Resumo

A coleta de lixo é uma atividade muito importante na sociedade atual, por causa dessa relevância os municípios estabelecem diretrizes legais para o cumprimento e realização dessa atividade. Segundo o plano municipal de saneamento básico da prefeitura de Formiga e conforme a lei Federal nº. 11.445/2007 do código de limpeza urbana, a coleta de lixo é um dos serviços de limpeza urbana que são responsabilidades básicas do poder executivo. Dados de 2008 produzidos pelo IBGE indicam que 98% dos domicílios localizados em zonas urbanas têm serviço de coleta de lixo. Além disso, o lixo coletado no Brasil no ano de 2020 alcançou a taxa média de 379,2 kg/hab/ano, ou seja, o Brasil produz aproximadamente 79 milhões de toneladas de lixo em 2018 (IBGE, 2012). Dada a importância do tema, propõe-se um protótipo de aplicativo para exibir graficamente rotas a serem seguidas na coleta de resíduos sólidos na cidade de Formiga. Neste trabalho, abordou-se o problema para definição de rotas para coleta de resíduos sólidos como instancia do Problema Roteamento de Veículos Capacitado (CVRP). Identificou-se na literatura que as abordagens algorítmicas mais utilizadas na resolução do CVRP são: Algoritmos Genéticos, Colônia de Formiga e Optimização por enxame de partículas. Implementaram-se tais algoritmos e realizaram-se computacionais em alguns *benchmarks*. Os resultados experimentais indicaram que a abordagem utilizando algoritmos genéticos é a mais promissora. Assim, aplicou-se tal abordagem para propor rotas de coleta de resíduos sólidos em bairros da cidade de Formiga. Por fim, foi construído um aplicativo em Flutter para mostrar os resultados e também uma API em Spring Boot para fazer a comunicação entre os códigos.

Palavras-chave: Coleta de Lixo, Meta-heurísticas, CVRP, Algoritmo genético, Colônia de formiga, Enxame de partículas.

Abstract

Garbage collection is a very important activity in today's society, because of this relevance, municipalities establish legal guidelines for compliance and carrying out this activity. According to the municipal basic sanitation plan of the city of Formiga and according to Federal law number 11.445/2007 of the urban cleaning code, garbage collection is one of the urban cleaning services that are basic responsibilities of the executive branch. Data from 2008 produced by IBGE indicate that 98% of households located in urban areas have a garbage collection service. Furthermore, the garbage collected in Brazil in 2010 reached an average rate of 306 kg/inhabitant/year, that is, Brazil produces approximately 160,000 tons of garbage per day (IBGE, 2012). Given the importance of the topic, an application prototype is proposed to graphically display routes to be followed when collecting solid waste in the city of Formiga. In this research, the problem of defining routes for solid waste collection was addressed as an instance of the Capacitated Vehicle Routing Problem (CVRP). It was identified in the literature that the algorithmic approaches most used in solving CVRP are: Genetic Algorithms, Ant Colony and Particle Swarm Optimization. Such algorithms were implemented and computational experiments were carried out on some benchmarks. The experimental results indicated that the approach using genetic algorithms is the most promising. Therefore, this approach was applied to propose solid waste collection routes in neighborhoods in the city of Formiga. Finally, an application was built in Flutter to show the results and also an API in Spring Boot to communicate between the codes.

Keywords: Garbage Collection, Meta-heuristics, CVRP, Genetic algorithm, Ant colony, Particle swarm.

Lista de ilustrações

Lista de tabelas

Lista de Abreviaturas e Siglas

ACO	Colônia de formigas (Ant colony optimization algorithms)
API	Application Programming Interface (Interface de Programação de Aplicação)
CARP	Protocolo de Encaminhamento de Armazenamento de Cache
CPU	Unidade Central de Processamento (Central Processing Unit)
CVRP	Problema de Rotas de Veículos Capacitados (Capacited Vehicle Routing Problem)
GA	Algoritmo genético (Genetic algorithm)
HFVRP	Entrega com Frota Heterogênea (Heterogeneous Fleet Vehicle Routing Problem)
OSM	OpenStreetMap
PSO	Optimização por enxame de partículas (Particle swarm optimization)
RSL	Revisão Sistemática da Literatura
RAM	Memória de Acesso Aleatório (Random Access Memory)
TSP	Problema do Caixeiro Viajante (Traveling Salesman Problem)
VRPTW	Problema de Roteamento de Veículos com Janelas de Tempo
VRP	Problema de Rotas de Veículos (Vehicle Routing Problem)

Sumário

1 Introdução

O desenvolvimento de rotas de coleta de lixo é uma parte essencial da gestão de resíduos sólidos em comunidades de pequeno porte. Como o Orçamento (2006) destaca em seu Manual de Organização dos Serviços de Limpeza Pública para Comunidades de Pequeno Porte, ao fazer a coleta de resíduos sólidos é considerado de suma importância para o sistema de limpeza pública, de um município. Pois é responsável pela remoção dos resíduos das ruas, praças, residências, entre outros.

Fazendo os serviços urbanos de limpeza de forma sugerida, pode-se absorver cerca de 15% dos recursos de um orçamento municipal. Além disso, serviços de coleta de lixo e transporte são responsáveis por 75% a 80% dos custos totais de limpeza de uma cidade. (BHAT, 1996)

Segundo Xavier (2010), a coleta de resíduos sólidos pode ser descrita de uma forma mais generalizada. Podemos colocar da seguinte forma: divide-se os locais e horas por semana da coleta, são divididas frotas de caminhões para coleta e os veículos vão para determinados bairros, também deve-se verificar qual é capacidade máxima que o veículo e sua capacidade de segurança e quando a coleta da área está completa, ele retorna para o depósito, onde ele será estacionado.

Tendo em vista as informações para otimizar os custos, é importante verificar o controle da rota pela qual o veículo passará durante a coleta de resíduos. Isso configura um problema de roteamento, pois assim conseguirá construir rotas baseadas em algoritmos, para essa coleta. (XAVIER, 2010)

Ainda segundo o estudo de Xavier (2010), o reteamento pode ser definido como a determinação da melhor sequência no qual as vias podem ser percorridas pelos veículos. Tendo como o objetivo minimizar os custos operacionais, distâncias percorridas, tempo dos trajetos e/ou o consumo de combustível. As decisões a serem tomadas irão se referir à definição dos melhores trajetos que devem ser executados ao verificar o grafo gerado por aquela malha viária.

Portanto, é de suma importância dar ênfase ao devido tratamento a esse problema de roteamento. Pois é a forma mais eficiente de diminuir os custos para fazer a limpeza urbana. Então, com a criação de rotas para deixar mais eficientes para os caminhões, o consumo de combustível dos veículos deve diminuir.

1.1 Justificativa

Os parâmetros a serem seguidos no processo de coleta dos resíduos na empresa 1AFormiga (2018), são:

- Tipo de resíduos a serem transportados;
- Estimativa do volume de resíduos a ser coletado;
- Definição das frequências de coleta;
- Definição dos horários de coleta (para domiciliar);
- Dimensionamento da frota dos serviços;
- Definição dos itinerários de coleta.

Esses parâmetros são semelhantes quando comparamos com a cidade de Formiga. Mesmo usando esses parâmetros citados para otimizar a coleta de lixo, ainda pode gerar uma série de problemas entre eles. Problemas esses tais como: a alta do trânsito em alguns dias da semana, a colocação menos eficiente dos pontos de lixo, entre outros problemas possíveis.

Segundo Xavier (2010), o consumo de combustível se constitui em uma parcela importante dos custos totais embutidos na coleta de lixo. Esses problemas geram um grande desperdício para as prefeituras locais, com caminhões fazendo gastos com combustíveis ao passar por possíveis rotas redundantes, ou seja, caminhos de forma que o veículo passe repetidamente por essas ruas.

Além disso, ao aplicar as técnicas no estudo de Xavier (2010), na cidade de Montes Claros, obteve-se uma redução no consumo de combustível de 8,2%, quando descontamos o trecho de acesso ao aterro sanitário. Isso pode melhorar a questão do meio ambiente com a diminuição da poluição devido ao menor gasto de combustível. Um exemplo que é possível resolver esse problema é a modelagem do Problema de Roteamento de Veículos, que será mais explorada durante o texto.

1.2 Objetivos

Nesta seção serão apresentados os objetivos a ser alcançados durante a pesquisa.

1.2.1 Objetivo Geral

O objetivo dessa pesquisa é a construção de uma prova de conceito para o problema de rotas e então modelar e implementar o protótipo de um aplicativo para propor e exibir rotas de coleta de resíduos sólidos na cidade de Formiga.

1.2.2 Objetivos Específicos

1. Utilizar a modelagem baseada em teoria dos grafos do Problema de Roteamento de Veículos Capacitado (CVRP) para solucionar o problema de coleta de resíduos sólidos;

2. Identificar as principais abordagens algorítmicas utilizadas para resolver o problema de coleta de resíduos sólidos por meio de uma revisão sistemática de literatura;

3. Medir a eficácia e a eficiência de algoritmos utilizados na solução do problema de coleta de resíduos sólidos, mediante *benchmarks*, e

4. Disponibilizar uma ferramenta que permita a exibir graficamente rotas seguidas na coleta de resíduos sólidos na cidade de Formiga.

2 Fundamentação Teórica

Neste capítulo serão apresentados alguns conceitos importantes para o entendimento completo desta monografia. Serão apresentados os seguintes conceitos: Grafos, a biblioteca OSMnx Problema do Roteamento de Veículos com Capacidade (CVRP), Algoritmos Genéticos (GA), Colônia de Formigas (ACO), Optimização por Enxame de Partículas (PSO), respectivamente.

2.1 Revisão Sistemática da Literatura

De acordo com Greenhalgh e Peacock (2014), a Revisão Sistemática de Literatura (RSL) é um método sistemático para que se possa fazer uma revisão de literatura de forma que busque identificar, avaliar e sintetizar todas as evidências relevantes daquele problema específico. Além disso, esse tipo de revisão segue um protocolo predefinido para garantir com a sua revisão seja o máximo objetiva e completa. Por isso, esse tipo de avaliação é frequentemente usada em pesquisas que precisem fornecer evidências baseadas em evidências para os pesquisadores conseguirem fazer suas tomadas de decisões.

Na execução de uma revisão sistemática de literatura é necessário seguir uma espécie de algoritmo, de forma que sejam padronizados os passos que devem ser seguidos. Esse algoritmo é baseado em 5 passos:

1. Definição de uma Pergunta de Pesquisa e Estratégia de Pesquisa: A pergunta de pesquisa deve ser específica o suficiente para ser respondida com as evidências existentes, mas ampla o suficiente para permitir uma revisão abrangente da literatura.
2. Critérios de inclusão e exclusão do estudo: os critérios de inclusão são as características que um estudo deve ter para ser incluído na revisão sistemática.
3. Extração e síntese de dados: Os dados extraídos devem ser sintetizados para responder à pergunta de pesquisa. A síntese de dados pode ser realizada por meio de uma meta-análise ou por meio de uma revisão narrativa.
4. Avaliação da qualidade e risco de viés: A avaliação da qualidade e do risco de viés ajuda a determinar a confiabilidade e a validade dos resultados da revisão sistemática.
5. Padrões e diretrizes para relatórios de RSL: Os padrões e diretrizes ajudam a garantir que a revisão sistemática seja relatada de forma clara e completa.

Os pesquisadores usam essa ferramenta para ter um amplo e atualizado conhecimento sobre um determinado tema. Isso pode ser benéfico para a criação de novas

investigações e para a tomada de decisões informadas. Isso faz a revisão sistemática de literatura uma ferramenta importante para a pesquisa científica.

2.2 Problema do Roteamento de Veículos com Capacidade

Segundo Wang e Li (2018), para fazer a geração das rotas do Problema de Rotas de Veículos temos que levar muitos fatores em consideração. Fatores esses como o veículo, a quantidade de armazenamento e rede rodoviária. Nesses problemas o problema de roteamento de veículo (VRP) é muito útil e atende com uma grande precisão. No entanto, quando consideramos capacidade dos caminhões e quantidade de pontos de coleta, o problema fica denominado *Capacitated Vehicle Routing Protocol* (CVRP).

Segundo Garey (1979), os problemas de roteamento normalmente são problemas difíceis. Isso acontece, pois muitas vezes esses problemas são derivados do caixeiro viajante. Como no VRP e no CVRP que são problemas NP-difíceis, ou seja, não se conhece algoritmos para resolvê-los em tempo polinomial. De acordo com Toth e Vigo (2002), o CVRP é um dos problemas mais estudados em otimização combinatória, pois, na prática, ele é muito relevante, além de ser um problema muito complexo.

Segundo Laporte (1992), existe uma característica principal no problema do CVRP que é a sua capacidade. Isso faz com esse problema se torne muito mais complexo que o Problema do Caixeiro Viajante (TSP). Além disso, para se encontrar uma solução ótima para esse tipo de problema é necessário fazer a exploração de um espaço de busca muito grande. No entanto, para fazer essas explorações requer muito tempo computacional.

Na pesquisa de Laporte e Nobert (1987), mostra-se como o CVRP é aplicável em diferentes contextos. A distribuição de alimentos, coleta de lixo e entregas em geral são alguns desses contextos. Além disso, o CVRP também é um utilizado em problemas semelhantes. Problemas este como Problema de Coleta e Entrega com Frota Heterogênea (*Heterogeneous Fleet Vehicle Routing Problem* - HFVRP) e o Problema de Roteamento de Veículos com Janelas de Tempo (*Vehicle Routing Problem with Time Windows* - VRPTW). Em Laporte (1992), o CVRP é apresentado como um problema de otimização combinatorial difícil. O CVRP é principalmente relevante na indústria de transporte, pois sua solução pode diminuir os custos de entrega, além de melhorar a qualidade dos serviços que estão sendo prestados aos clientes.

Na pesquisa Toth e Vigo (2002), mostra como o CVRP tem sido objeto de estudo em diferentes áreas. No seu estudo mostra-se que há resoluções usando inteligência artificial, pesquisa operacional, entre outros. Além disso, observa-se que estão sendo criadas diferentes soluções para resolver o problema, que incluem técnicas heurísticas, exatas e híbridas, que combinam as duas abordagens diferentes em busca de uma melhor solução.

Toth e Vigo (2002), destaca principalmente os algoritmos heurísticos e meta-heurísticos, que estão dentre as principais ferramentas para abordar o CVRP. Esses tipos de algoritmos, apresentam bons resultados, mesmo tendo em vista suas limitações de não garantir o valor ótimo e exigir recursos computacionais consideráveis. Dentre os algoritmos meta-heurísticos mais utilizados para resolver esses problemas estão o algoritmo genético, a colônia de formigas, o *simulated annealing* e o *tabu search*.

2.3 Algoritmos Genéticos

Segundo Mitchell (1998), um algoritmo genético é uma técnica evolutiva, inspirada na seleção genética. Essa recombinação de gene e possíveis mutações podem ajudar a encontrar soluções para problemas de combinação. Ou seja, esses algoritmos são baseados na seleção natural e hereticidade. Então conseguimos ter a evolução de soluções por meio de sucessivas gerações.

O pioneiro da pesquisa em Algoritmos Genéticos, Holland (1975), descreve o funcionamento dos algoritmos genéticos da seguinte forma. Ele é um algoritmo que tenta encontrar o ótimo global de um problema de otimização utilizado um processo iterativo de seleção, *crossover* e mutações. Cada desses indivíduos são mais adaptados a sobreviver e conseguir reproduzir, enquanto há outros que são menos aptos à sobrevivência, então são descartados. Em seguida, os sobreviventes desse processo de seleção passam por uma recombinação, chamada *crossover*, para fazer a criação de novos indivíduos. Por fim, esses novos indivíduos podem passar por uma mutação aleatória, alterando os genes desse indivíduo. Isso faz com que aumente a diversidade da população e evite uma convergência prematura, evitando ótimos locais. Os principais componentes de um algoritmo genético, descritos na literatura, são:

1. Cromossomos: Representam soluções candidatas para o problema em questão. (GOLDBERG, 1989)

2. Genes: São os elementos individuais do cromossomo, correspondendo a características específicas da solução. (GOLDBERG, 1989)

3. Codificação: A representação das soluções do problema em forma de cromossomos. A escolha da codificação depende do domínio do problema. (MITCHELL, 1996)

4. Função de Aptidão (Fitness): Define o quão boa é uma solução em relação ao objetivo. Essa função é crucial, ao guiar a seleção de indivíduos para a reprodução. (HOLLAND, 1975)

5. População: Um conjunto de indivíduos (cromossomos) que representam possíveis soluções para o problema. A população evolui ao longo das gerações. (EIBEN; SMITH, 2015)

6. Seleção: Indivíduos mais aptos têm maior probabilidade de serem selecionados para reprodução. Diversas estratégias de seleção podem ser empregadas, como seleção proporcional ao fitness, torneio e roleta. (WHITLEY, 1994)

7. Recombinação (Crossover): Processo que combina informações genéticas de dois pais para gerar descendentes. O crossover simula a troca de material genético durante a reprodução. (GOLDBERG, 1989)

8. Mutação: Introduz pequenas alterações aleatórias nos cromossomos, aumentando a diversidade genética da população. (MITCHELL, 1996)

9. Critérios de Parada: Define quando o algoritmo deve encerrar. Pode ser um número fixo de gerações, convergência para uma solução ótima, ou outros critérios específicos ao problema. (HOLLAND, 1975)

Segundo Hwang e Kim (2010), os algoritmos genéticos têm sido amplamente estudado em problemas de roteamento. Dentre os problemas nos quais usam a abordagem de algoritmos genéticos, citam-se: o problema do caixeiro-viajante e o problema de roteamento de veículos. Ao observar seus resultados têm demostrado ser eficazes na obtenção das soluções ótimas ou próximas do ótimo em um tempo razoável. Em Goldberg (1989), destaca-se a utilidade e a capacidade de algoritmos genéticos na solução de problemas nos quais deve-se explorar um grande espaço de busca. Cita-se, também, que tal abordagem é eficaz em situações nos quais o espaço de busca não é bem definido.

Segundo Coello (2002), a importância dos Algoritmos Genéticos é que são uma técnica muito versátil. Esses algoritmos são uma técnica que pode consegue lidar com uma grande gama de problemas. Quando observamos literatura em quais problemas ele pode ser aplicado achamos alguns como: problemas com restrições, problemas de otimização multiobjetivo, problemas dinâmicos, entre outros. A sua versatilidade torna os GA's um algoritmo muito valioso e amplamente utilizado por profissionais de diversas áreas.

De acordo com Uzkent e Balcik (2015), essa aplicação usando os algoritmos genéticos em problemas de roteamento tem o potencial de gerar melhorias significativas. Aplicado em questões logísticas pode-se observar redução de custos de transporte e tempo de entrega, além de possibilitar uma tomada de decisões mais estratégicas e sustentáveis.

No estudo de Glover e Laguna (1997), esse algoritmo tem sido amplamente utilizado para a solução do problema de roteamento CVRP. Ao utilizar observa-se uma melhora muito eficaz nas soluções, mesmo quando aplicada em instâncias de diferentes

e complexidades. Além disso, o Algoritmo Genético é estudado de forma que explore as diferentes possibilidades desse algoritmo. Possibilidades essas como: operadores genéticos, estratégias de seleção, esquemas de codificação e outras técnicas.

Na pesquisa de Toth e Vigo (2002), destaca o uso de Algoritmos Genéticos, aplicado ao problema do CVRP. Nesse contexto, ele facilitou a melhoria da eficiência operacional das empresas nas quais foram aplicadas essas técnicas com Algoritmos Genéticos. Ao fazer isso foi possível gerar uma solução na qual reduziu os custos e o tempo de entrega dessa empresa. Além disso, essa solução faz com que as empresas sejam mais sustentáveis, reduzindo a emissão de poluentes, isso faz com que as atividades ambientais gerem menos impacto ao meio ambiente.

2.4 Colônia de Formigas

Segundo, o criador dessa meta-heurística, Dorigo e Stützle (2004), o algoritmo de Colônia de Formigas (ACO) é uma meta-heurística inspirada na forma na qual as formigas buscam comida. Quando observamos a natureza observamos que existem algumas formigas batedoras, essas formigas vão em busca de comida e então vão depositando feromônios para as subsequentes. As próximas formigas têm uma tendência de seguir esse caminho, pois já foram pré-validados por outras formigas e quanto maior a quantidade de feromônio no caminho menos chance tem dessa formiga desviar do caminho. Esse processo descrito faz com que as formigas achem comida de forma mais eficiente. A colônia de formigas é pensada em agentes que buscam soluções computacionais complexas usando a mesma forma que esses animais buscam comida na natureza.

As primeiras abordagens deste algoritmo foram para o CVRP, proposta por Dorigo, Maniezzo e Colorni (1996), nesse momento introduziram o algoritmo conhecido como o *Ant System*. Nesse algoritmo as formigas utilizam feromônios artificiais durante a construção das soluções, no entanto, ao longo do tempo esses feromônios vão evaporando. A intensidade desse feromônio é atualizado a cada iteração consoante as soluções geradas e também com a quantidade de formigas que utilizam aquela aresta. Os principais conceitos do algoritmo de colônia de formiga são:

1. Formigas Virtuais: No ACO, soluções candidatas para um problema são representadas como trilhas de feromônios virtuais. Ela constrói uma solução iterativamente, depositando feromônios nas arestas visitadas. (DORIGO, 1992)

2. Feromônios: Os feromônios representam a qualidade de uma solução em uma determinada parte do espaço de busca. A intensidade dos feromônios influência a probabilidade de outras formigas virtuais escolherem o mesmo caminho. (DORIGO, 1992)

3. Exploração: As formigas virtuais equilibram a procura de novas soluções e seguir caminhos ricos em feromônios. Isso ajuda a evitar a convergência prematura para soluções subótimas. (DORIGO et al., 1997)

4. Atualização dos Feromônios: Após cada iteração, os feromônios são atualizados com base na qualidade das soluções encontradas. Caminhos mais curtos e melhores soluções recebem uma atualização mais forte dos feromônios. (DORIGO; STüTZLE, 2004)

De acordo com Blum e Roli (2003), ao utilizar o ACO ele percebeu que esse algoritmo consegue gerar soluções de alta qualidade, mesmo considerando os problemas de otimização mais complexos conhecidos. Nessa pesquisa ele observou que mesmo em modelos com muitas variáveis e restrições o algoritmo despenhou bem. Além disso, é uma técnica muito maleável e fácil de ser implementada a uma variedade de problemas de otimização. No entanto, é importante destacar que o algoritmo de colônias de formiga não garante sempre a melhor solução, mas uma solução, em tempo menor que algoritmos que garantem a melhor solução e sendo uma solução próxima conhecida como ótimos locais. Além disso, segundo Li e Cao (2018), o ACO consegue encontrar rotas ótimas para problemas de roteamento de veículos com uma alta taxa de eficiência, mesmo quando há múltiplos veículos e clientes.

Segundo Stützle e Hoos (2000), os algoritmos de formigas vem sendo amplamente utilizado em diversas áreas, que possuem uma otimização combinatória. Problemas tais como problemas de roteamento de comunicação entre redes, problemas derivados do caixeiro-viajante, problemas derivados do VRP, entre outras opções que necessitam de rotas. Além disso, ele tem uma acurácia mais bem sucedida entre os algoritmos de meta-heurísticas e vem sendo aplicado com uma gama de problemas gradativo. Essa abordagem cada vez mais mostra resultados promissores em problemas complexos, como os citados.

Dorigo et al. (1997), ele propôs uma versão diferente do ACO, nessa versão ele aplicava duas diferentes heurísticas para conseguir construir soluções iniciais. Em seguida, ao ter soluções iniciais já otimizadas, então passamos por uma fase de otimização por busca local, para conseguir melhorar mais ainda a solução inicial. Então só assim era executado o algoritmo de colônia de formigas. Três anos depois uma abordagem semelhante foi proposta por Stutzle e Hoos (2000), nessa proposta ele também usaria a busca local para melhorar a qualidade da solução inicial. Os algoritmos têm a tendência de otimizar a solução inicial, por ter o problema de ser aleatória.

2.5 Optimização por enxame de partículas

Segundo Kennedy e Eberhart (1995), a otimização por enxame de partículas (PSO) é um algoritmo de otimização que foi construído se baseando em como populações. Isso faz com que eles busquem soluções, via um processo iterativo, simulem um comportamento social de um exame que estão se movendo a fim de encontrar uma solução ótima daquele espaço de busca. Isso faz com que ela seja uma técnica computacional que seja amplamente utilizada para encontrar soluções ótimas para espaços de busca complexos.

Ainda segundo os criadores do algoritmo PSO 1995, a ideia desse algoritmo é bem simples, considerando cada uma das partículas da população inicial elas irão procurar a melhor posição naquele espaço de busca considerando alguns fatores. Nessa técnica ele considera a sua posição atual, a melhor posição que ele já encontrou nessa solução e a melhor solução encontrada pelo seu enxame. Utilizando esses parâmetros faz com que ele tenha um comportamento colaborativo e faça com que aprenda com seu enxame, dando as suas experiências individuais anteriores. Então, dessa forma eles compartilham as suas informações e caminham em direção às melhores soluções em busca do ótimo global da solução.

Além disso, ainda segundos os mesmo autores 1995, cada uma das partículas desse enxame representa uma solução diferente, na qual ela consegue aprender novas soluções. Isso faz com que ela encontre as melhores soluções para aquele enxame. Essa técnica de otimização foi inspirada na natureza, observando alguns animais que se movem em forma de enxames como pássaros e peixes. Esses animais se movimentam de maneira coordenada para em busca de alimento e abrigo e migrando seus locais quando naquela área está desprovida desse. Os fatores importantes na Optimização por Enxame de Partículas incluem parâmetros que influenciam o desempenho e o comportamento do algoritmo:

1. Partículas: No algoritmo de optimização por exame de partículas há uma solução candidata, o algoritmo há representa por uma partícula. Cada uma dessas soluções candidatas ocupa uma posição no espaço de busca associada a uma solução potencial para o problema da otimização. (EBERHART; KENNEDY, 1995)

2. Enxame: O conjunto de partículas forma o enxame. Cada partícula ajusta sua posição no espaço de busca com base em sua própria experiência (melhor posição encontrada até o momento) e na experiência do enxame (melhor posição global encontrada até o momento). (EBERHART; KENNEDY, 1995)

3. Movimento: O movimento de uma partícula é determinado por duas componentes vetoriais: velocidade e posição. A velocidade da partícula é ajustada de acordo com equações que incorporam seu conhecimento atual sobre o espaço de busca. (EBERHART; KENNEDY, 1995)

4. Inercia (ω): Controla a influência da velocidade anterior na atual. Um valor alto favorece a exploração, enquanto um valor baixo favorece a exploração. (EBERHART; KENNEDY, 1995)

5. Fatores de Aprendizado (c_1 e c_2): Controlam a influência das experiências pessoais e coletivas na movimentação das partículas. (EBERHART; KENNEDY, 1995)

Segundo Eberhart e Kennedy (1995), a optimização por enxame de partículas possui equações básicas. As seguintes equações representam o básico desse algoritmo:

Equação 1:

$$v_i * (t+1) = \omega * v_i * (t) + c_1 * r_1 * (pbest_i(t) - x_i * (t)) + c_2 * r_2 * (gbest(t) - x_i * (t))$$

Equação 2: $x_i * (t+1) = x_i * (t) + v_i * (t+1)$

Onde:

- v_i é a velocidade da partícula $_i$ no instante t
- x_i é a posição da partícula $_i$ no instante t
- ω é o fator de inércia
- c_1 e c_2 são fatores de aprendizado
- r_1 e r_2 são números aleatórios entre 0 e 1,
- $pbest_i(t)$ é a melhor posição alcançada pela partícula $_i$ no instante t
- $gbest(t)$ é a melhor posição global encontrada pelo enxame até o instante t

Segundo os estudos de Janson, Middendorf e R"onnqvist (2010), ao utilizar esse algoritmo ele consegue encontrar soluções de alta qualidade para problemas de roteamento com um tempo bastante considerável. O algoritmo é muito útil para lidar com problemas de grande complexidade combinatória. Um exemplo desses problemas é o Problema de roteamento de veículos (VRP), as quais são um problema com uma grande complexidade combinatória, podendo ter uma imensidão de rotas e clientes que precisam ser atendidos.

Segundo a pesquisa de Kennedy e Eberhart (1995), o PSO apresentou uma boa capacidade de convergir em problemas de otimização combinatória. Isso se deve a sua possibilidade de possuir uma natureza probabilística, baseada na sua população, além de possuir uma natureza iterativa. Essa abordagem ainda permite com que as técnicas de exploração de espaços sejam mais eficientes do que outras técnicas de otimizações já conhecidas.

Além disso, o PSO tem sido combinado com outras técnicas de otimização conhecidas, como o algoritmo genético. Isso faz com que ele melhore a sua capacidade de convergir a soluções ótimas em problemas relacionados ao VRP. Essa combinação do PSO com o algoritmo genético, está resultando em melhores soluções e, além disso, conseguiram diminuir o tempo de execução considerando a utilização problemas relacionados ao VRP em comparação com outras abordagens. (SHEN; WU, 2014)

O algoritmo de otimização por partículas têm sucesso em diversas áreas, isso inclui problemas de roteamento de veículos, como o VRP, o CVRP, Além disso, segundo as pesquisas de Kim, Kim e Kim (2016), ao aplicar o PSO para resolver problemas de otimização do tipo CVRP, ele apresentou resultados satisfatórios. Isso comparando com outras técnicas de otimização, como busca linear e busca tabu.

3 Desenvolvimento

Este capítulo apresenta as etapas utilizadas no desenvolvimento do presente trabalho. Aborda a realização da revisão sistemática de literatura e seus resultados, os algoritmos escolhidos, a execução dos algoritmos em *dataset's*, a construção do *dataset* de bairros da cidade Formiga e a execução das meta-heurísticas e a construção da API de comunicação.

3.1 Revisão Sistemática da Literatura

Ao iniciar o projeto foi baseado em cima de duas perguntas para direcionar a pesquisa.

Qual o tipo de modelagem mais usada para desenvolver tipo de roteamento e qual sua complexidade computacional?

Qual a forma mais utilizada para solucionar esse problema?

Ao escolher essas perguntas começamos a separar quais seriam palavras-chave para a construção de uma *string* de busca. Essas chaves escolhidas foram: Otimização Rotas, Modelagem em Grafos, Rotas de Lixo, Coleta de Resíduos Sólidos, Roteirização de Veículos.

Começou-se a gerar diferentes *strings* de buscas, para otimizar a busca. Para fazer os testes de otimização escolhemos o *Google Scholar* para testar *strings*; ele pode retornar artigos de diferentes bases de dados. Para tal, coloca-se 2 tipos de parâmetros diferentes para desenvolver, considerando a otimização, a quantidade de artigos aplicado àquela área e as qualidades desses artigos. Além disso, também escolhemos quais seriam os idiomas que iríamos pesquisar, que foram o inglês e o português.

Ao criar diferentes *strings* de busca, escolhido o mais otimizado para essa pesquisa. Com essa a *string* foi possível obter 704 artigos em língua portuguesa e 16.700 em língua inglesa. As *strings* escolhidas foram as abaixo:

(Coleta de Lixo OR Coleta de Resíduos Sólidos) AND
(Otimização de Rotas OR Grafos) AND
(Algoritmos de Roteirização OR Otimização de Rotas OR Roteirização de veículos)

(Garbage Collection OR Solid Waste Management) AND
(Optimization Routes OR Graphs) AND
(Routing Algorithms OR Optimization Routes OR Vehicle Routing)

No entanto, para seguir a pesquisa escolhemos algumas bases de dados, pois o *Google Scholar* tem algumas limitações para fazer essas pesquisas, como ele faz uma pesquisa muito ampla, muitas vezes ele não retorna muitos artigos que não fazem muito sentido com a pesquisa. Então foram escolhidas as seguintes base dados:

- IEEE Xplore Digital Library
- ACM Digital Library
- ScienceDirect
- Catálogo de Teses e Dissertações
- Biblioteca Digital de Teses e Dissertações
- Portal de Periódicos da CAPES
- CAFe-CAPES

A aplicação da *string* de busca nessas bases retornou 221.201 documentos. Posteriormente realizou a etapa de leitura desses documentos seguindo o protocolo da revisão sistemática da literatura. Ao final do processo sobraram o total de 157 artigos em língua inglesa e disponíveis em bases gratuitas.

Após essa análise concluiu-se que há uma preferência por parte dos pesquisadores pelo uso de meta-heurísticas na solução do CVRP. Além disso, todas modelagens desse problema são NP-Difícil, ou seja, não se sabe se existe alguma solução em tempo polinomial. Igualmente, como mostra o gráfico da figura 1 a maioria dos pesquisadores usa o CVRP, seguido pelo CARP e no gráfico da figura 2 exibem-se os algoritmos mais utilizados para resolver o problema de roteamento entre caminhões de lixo, a saber: ACO, PSO e Algoritmo Genético.

Figura 1 – Gráfico dos problemas mais utilizados na modelagem.

Figura 2 – Gráfico dos algoritmos mais utilizados.

3.2 Cluster

Um cluster consiste em um grupo de computadores interconectados que trabalham de maneira colaborativa para realizar tarefas complexas. Uma das grades dificuldades do projeto foi lidar com a abundância de datasets que estavam disponíveis no Set-A (1995). Além disso, os algoritmos em computadores normais chegariam a rodar até mesmo dias uma mesma instancia mais complexa. Para a execução dessas tarefas foi utilizado um cluster. Neste contexto, a configuração adequada do cluster desempenha um papel crucial para otimizar o desempenho e garantir a eficiência no processamento de dados em grande escala.

A eficiência de um cluster para processamento de dados em grande escala depende

significativamente da adequada configuração de memória e CPU. Esses elementos são cruciais para garantir o desempenho, a escalabilidade e a capacidade de processamento do cluster. Esse cluster possui uma CPU xeon com 2,4 GHz, além disso, ela possui 16 núcleos. De memória RAM ele possui 23,44 GB e de armazenamento interno ele possui 30 GB.

Outra escolha importante para a desempenho do cluster é a escolha de seu sistema operacional.A escolha do SO afeta diretamente a estabilidade, a segurança e a eficiência operacional do cluster. Para esse cluster foi escolhido uma distribuição Linux, o sistema operacional Ubuntu 22.04.1 LTS.

3.3 Modelagem e Implementações

Ao fazer a análise dos artigos listados na seção anterior, percebemos que a melhor modelagem para esse problema é o Problema de Roteamento de Veículos com Capacidade (CVRP).

A escolha da CVRP se baseou em algumas de suas principais características que impactam muito nesse problema. Uma das suas características é a restrição da capacidade de carga de cada um dos veículos, o que torna esse problema muito mais difícil por serem um problema que pode ser reduzido no Problema da Mochila. Como observado por Martello e Toth (1990), o CVRP um problema de otimização combinatória que envolve o roteamento de veículos que têm que atender demandas pelo grafo levando em consideração a quantidade de carga que ele consegue carregar. O objetivo dele é minimizar o custo total do transporte de uma caga, incluindo o custo da viagem e o custo dos veículos.

Além disso, ele leva em consideração a capacidade de carga dos veículos ao planejar as rotas, isso garante que os veículos não sejam sobrecarregados e que todas as demandas possam ser atendidas. A figura 3 representa um exemplo de uma rota resolvida, usando a modelagem do CVRP. Nessa modelagem temos um depósito e os locais que o veículo irá atender as demandas, então devemos gerar uma rota otimizada de forma que ele atenda todas as demandas e volte ao depósito para se descarregar, de forma que o veículo não fique sobrecarregado e consiga fazer o todo o caminho.

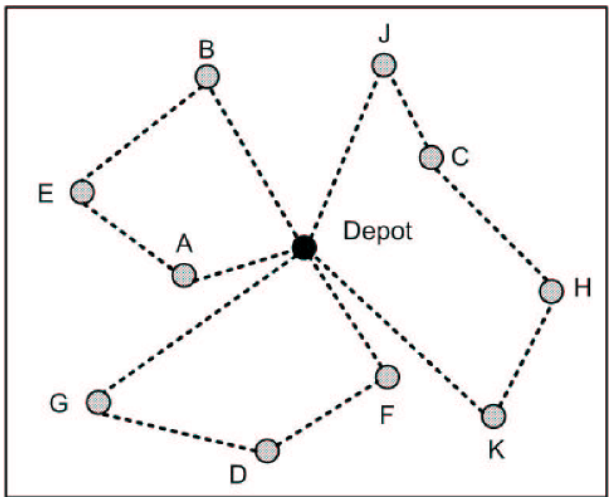

Figura 3 – Rota resolvida por um algoritmo de CVRP.

No problema encontrado, sobre a coleta de lixo em Formiga, temos essa exatamente essa situação. Em Formiga um ou dois caminhões andam por um bairro, neste local tem uma demanda média de lixo a ser coleta, esses caminhões têm que iniciar a sua rota e vão caminhando pelos bairros, mas ao caminhão estar lotado eles tem que retornar ao ponto de coleta para descarregar. Além disso, caso o caminhão esteja lotado sem terminar a rota em um bairro antes de terminar a rota por uma eventualidade, é necessário que ele descarregue no seu respectivo depósito, lixões, aterros sanitários, entre outros. Feito isso, ele retorna ao local para recolher os restantes dos resíduos sólidos daquela região. Se considerarmos a figura 3 um bairro de Formiga, ao coletar os pontos (B,E,A) e estar com o veículo lotado, então teríamos que retornar ao depósito (aterro sanitário) descarregar o caminhão e retornar para os pontos ainda não coletados.

Na modelagem proposta não se mapeou cidade inteira de Formiga, pois haveria pontos de coleta o suficiente para que ficasse inviável trabalhar por conta da execução do algoritmo. Além disso, subdividido a cidade faz a simulação ficar mais realista, pois a coleta é feita por diferentes caminhões em diferentes dias. Ao estar fazendo a coleta de lixo, o local de entrega seria em uma posição séria em um ponto fora do local predestinado da coleta, ou seja, ele estaria em uma posição levemente na saída daquele mapa.

3.3.1 Grafos

Grafos são estruturas matemáticas que consistem em vértices (ou nós) e arestas (ou arcos) que conectam esses vértices. Um dos principais pontos para resolver tal problema é a modelagem de problema em grafos. Os primeiros estudos sobre teoria dos grafos surgiram em Königsberg (território da Prússia até 1945, atual Kaliningrado). A origem da teoria dos grafos iniciou-se com um desafio que ficou conhecido como O Problema das Pontes de Königsberg, conforme pode ser visto em Rabuske (1992). Esse desafio consistia em verificar

a possibilidade de um caminho que passe pelas sete pontes existentes em Königsberg sem repetir nenhuma ponte. O matemático Euler (1736) modelou o problema transformando o caminho das pontes em retas e suas interseções em pontos, criando assim um grafo. (VIEIRA, 2018).

No estudo de Santos (2006), podemos observar como a utilização de grafos é uma forma eficiente para fazer representação um bairro e cidades. Nessa modelagem consideram-se os logradouros como arestas e as esquinas dos logradouros como vértices. Quando se utiliza essa representação é possível colocar as coordenadas geográficas, então assim conseguiremos aplicar uma série de algoritmos de grafos nesse mapa. No entanto, é importante se atentar para algumas situações, como curvas nas ruas e a própria curvatura do planeta terra. Na figura 4 apresentam-se alguns logradouros modelados em forma de grafos.

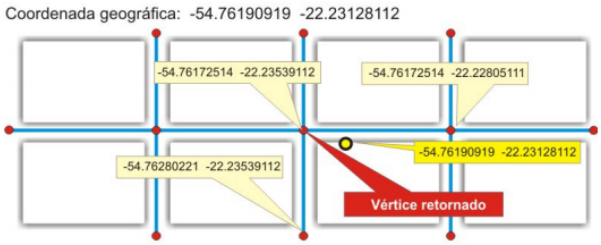

Figura 4 – Modelagem de ruas em grafos.

Na pesquisa de Toth e Vigo (2014), usa-se a modelagem matemática mediante grafos. Essa modelagem tem como uma das técnicas base, para resolver o problema do CVRP, o uso de modelagem do problema por meio de grafos. Além disso, também é possível aplicar uma série de algoritmos específicos para a solução desses problemas.

3.3.2 Algoritmos Genéticos

O algoritmo genético desenvolvido neste projeto foi baseado no código de Varshey (2021). Para a execução desse algoritmo são necessários 3 parâmetros, a saber: nome do arquivo, a quantidade de interações e o número de soluções. Ao passar esses parâmetros durante a leitura do dataset ele constrói uma matriz de distância entre dois pontos. Além disso, ele verifica a capacidade do veículo, dimensão deste dataset e uma lista com valores da coordenada X e outro da coordenada Y, para gerar os pontos geométricos da solução. Feito isso, o algoritmo genético cria uma solução inicial, para começar a executar o algoritmo em cima dessa população inicial.

Nesse algoritmo existem 3 funções: a primeira é para fazer os cálculos dos valores objetivos, como o mínimo de veículos exigido, a distância total percorrida por todos os veículos e a diferença entre o mais barato e o mais caro. Com esses dados é possível executar a segunda função objetivo, ela calcula a distância total percorrida por cada veículo e o

equilíbrio da rota. Na terceira com o valor da distância e o com a solução atual ela verifica o somatório das distâncias.

Então ao executar a função objetiva faz-se a seleção dos valores. Gerando novos valores aleatórios e verificando se eles são melhores ou piores que os valores da solução atual. Então é possível fazer os processos de *crossover* para o algoritmo. Durante esse processo ele constrói duas soluções temporárias possíveis de serem as soluções verdadeiras, respeitando os valores iniciais, como a demanda, as distâncias, a capacidade de cada veículo. Depois do *crossover* faz possíveis mutações desses valores para cada uma das 2 soluções.

Ao fazer esses processos, foi preciso saber qual a melhor solução, então chamamos as funções objetivo novamente para fazer a sua avaliação, dos novos valores. Então fizemos a ordenação dos melhores cromossomos. Então é executado pela quantidade de interações definidas. Por fim é gerado a melhor solução, podendo ser gerado diferentes valores com elas.

3.3.3 Colônia de Formigas

O algoritmo de para a solução por colônia de formigas foi baseado nos seguintes códigos Elsom1945 (2022) e Konowrocki (2022). Esse algoritmo precisa de três parâmetros, sendo eles o nome do arquivo, o total de iterações e a quantidade de formiga. Para esse algoritmo é necessário algumas informações do dataset. Elas são: a capacidade do caminhão, grafo em forma de lista de adjacência, a lista de demanda também em um dicionário, o melhor valor e suas coordenadas. Então usando o grafo é calculado os valores de cada um dos vértices do grafo e quantidade de feromônios de cada vértice.

Quando é feito isso, temos os dados para começar a execução do algoritmo de colônia de formiga. A primeira coisa é escolher um valor aleatório para verificar e checar sua demanda e a capacidade limite do veículo. Feito isso, o ACO cria sua matriz de probabilidades daquele vértice ser o caminho correto. Então enquanto o valor das matrizes de capacidade do veículo não for zero, ele continua fazendo isso e otimizando essas probabilidades. Ao encher o veículo é considerado que a achou o primeiro caminho.

Quando encontramos o melhor caminho onde se chega na restrição de limite do veículo, precisamos atualizar os feromônios, para que na próxima iteração ele esteja melhor que a anterior. Então o algoritmo continua fazendo essa execução durante todo o processo até o número de iterações e o número de formigas chegarem no máximo e retorna a melhor solução.

3.3.4 Optimização por enxame de partículas

O algoritmo de para a solução por otimização por enxame de partículas foi baseado no seguinte códigos Elsom1945 (2022). Esse algoritmo precisa de dois parâmetros, sendo eles o nome do arquivo e o total de iterações. Para esse algoritmo é necessário algumas informações do dataset. Elas são: a capacidade do caminhão, o tamanho do grafo, o grafo em forma de matriz de adjacência, a lista de demanda, dimensão deste dataset e uma lista com valores da coordenada X e outro da coordenada Y, para gerar os pontos geométricos da solução.

Ao ler a entrada, já se pode iniciar a execução do algoritmo do PSO. Primeiramente é criado duas lista aleatórias que representam a posição das partículas e a velocidades das partículas. O tamanho delas é a quantidade de partículas pelo tamanho do grafo.

Semelhante ao algoritmo genético, o PSO, também uma função de treinamento. A função de treinamento fornece o candidato de aptidão, ou seja, a distância calculada para cada geração. Para fazer isso, primeiramente ela retorna os clientes classificando a posição da partícula atribuída. Após serem classificados, então retornam a distância mínima percorrida por todos os veículos, ou seja, candidato a aptidão. Para definir o melhor pessoal e global para o valor mínimo de aptidão do candidato é comparado o valor do treinamento que se acabou de fazer com um valor do treinamento anterior. Se faz isso até o momento que seja satisfeito a condição inicial passada por parâmetros por número de iterações.

Ao verificar o melhor então recalcula-se a posição das partículas e velocidade das partículas novamente para verificar se houve melhora de desempenho e fazer os ajustes necessários. Por fim, retornamos os clientes classificando a posição da partícula atribuída a eles. Então continuamos a execução do algoritmo verificando os clientes atendidos por cada veículo. Se faz isso para verificar se a capacidade máxima do veículo já foi atingida ou podemos armazenar o espaço de armazenamento. Por fim é apresentado os melhores valores para aquele algoritmo.

3.3.5 OSMnx

A biblioteca criada por Boeing (2017), utiliza a API de código aberto do *OpenStreetMap* (OSM), que possui Licença *Open Data Commons Open Database License* (ODbL) (OPENKNOWLEDGE, 2015). O OSM é um projeto colaborativo global, que oferece dados geoespaciais abertos, aos seus usuários. Lançado em 2004, o projeto é administrado pela *OpenStreetMap Foundation*, uma organização sem fins lucrativos sediada no Reino Unido (OPENSTREETMAP, 2023).

Zhang (2018), mostrou como a biblioteca OSMnx apresenta recursos para as análises de redes espaciais. A biblioteca permite a criação de redes de ruas, que consistem em

uma rede de ruas conectadas em uma malha. A rede pode ser criada para uma área específica, selecionando-se um ponto de origem e um raio de busca. A partir dessa rede, a biblioteca permite a criação de gráficos de rede, que permitem a análise de propriedades como comprimento das ruas, distâncias entre pontos e grau dos nós.

Segundo Boeing (2017), um de seus principais pontos fortes é sua capacidade de extrair redes de ruas em alta qualidade. Utilizando o *OpenStreetMap* faz com que exista uma ampla gama de áreas urbanas que possam ser extraídas. A biblioteca consegue baixar desde de dados brutos do *OpenStreetMap*, até mesmo fazer a limpeza dos dados, fazendo a coleta apenas das partes necessários para a solução do problema. Além disso, a biblioteca possui recursos para processar os dados e construir redes viárias precisamente e voltada para que usuário consiga ver qualquer lugar do mundo.

Além disso, segundo Zhang (2018) a biblioteca OSMnx, permite além das visualizações dos dados de redes, também realizar cálculos de distância dessas rotas. A visualização de rotas possibilita entender a estrutura da rede do problema abordado. Ademais, permite identificar problemas possíveis como gargalos ou problemas de tráfego. A biblioteca OSMnx permite a visualização ser em 3D, dessa forma representado o ambiente urbano da maneira mais realista possível.

Na pesquisa de Gad (2020), listam-se alguns recursos da biblioteca, tais como: cálculo de rotas, cálculo de distâncias e tempos de viagem, identificação de sub-redes e identificação de pontos de interesse. Essa biblioteca não apenas extrai os dados do *OpenStreeMap*, mas também oferece ferramentas de análise de malhas viárias. Isso permite estudar as características das redes de ruas dos locais.

3.4 Configuração dos Experimentos

Segundo Wickham e Grolemund (2017), R é uma linguagem de programação poderosa e flexível para computação estatística. A linguagem R é amplamente utilizada em pesquisas acadêmicas. Conforme Matloff (2018), a linguagem R é versátil e poderosa, podendo ser usada em uma ampla gama de tarefas, incluindo análise de dados, visualização de dados, aprendizado de máquina, entre outros.

Com o objetivo de obter uma maior confiabilidade aos testes realizados, desenvolveu-se, na linguagem R, dois algoritmos. O primeiro algoritmo determina a quantidade de vezes que os testes deveriam ser executados. O segundo algoritmo foi utilizado para realizar um teste de hipótese. Tais algoritmos foram baseados na referência Ramos (2016).

O primeiro algoritmo tem 3 principais parâmetros que são o valor de confiança, que foi escolhido 95%, a margem de erro, o escolhido foi 0,05% e por fim o desvio padrão, o escolhido 0,01%. Após ter feito isso o próximo passo foi calcular o tamanho da amostra,

primeiramente encontramos o valor crítico de para esse nível de confiança usando os valores da curva normal. Então fazemos o cálculo do tamanho da amostra carregando os valores dos parâmetros. Então por fim o tamanho da amostra foi de 1537 execuções necessárias para poder executar um teste de hipótese.

O segundo algoritmo dada uma lista dos resultados, melhor valor e uma tolerância de dados. Ele executa o *script* e compara quais valores estão dentro da tolerância aceita. Para esse algoritmo foi usado uma de 0,3 maior que o valor do algoritmo. Por fim, ele retorna a porcentagem de valores bons.

4 Resultados

Neste capítulo apresenta-se resultado deste trabalho. São exibidas as análises dos testes executados e o protótipo do aplicativo móvel.

4.1 Executando *Benchmark*

Para testar a acurácia desses algoritmos citados na seção (**??**) é importante os executar em diversos *dataset's*. Para esse projeto foi escolhido o Set-A (1995) desenvolvido por Augerat, Cordeau e Laporte (1995). O Set-A (1995) consiste em 10 instâncias do problema CVRP, cada uma com 100 clientes e um número variável de veículos e capacidade de veículos. Essas instâncias foram geradas aleatoriamente, com um conjunto de parâmetros selecionados para criar instâncias de tamanho médio e alta complexidade. (AUGERAT; CORDEAU; LAPORTE, 1995)

Além disso, cada instância tem uma solução ótima fornecida pelo artigo, obtida usando um algoritmo de *branch-and-cut*. Segundo o Augerat, Cordeau e Laporte (1995),os valores para as soluções desses dataset's, os limites superiores e inferiores, foram encontrados por um algoritmo de *branch-and-cut*. Esse algoritmo foi implementado em uma ferramenta de software conhecida como CPLEX.

Para executar esses *benchmark's* foi criado um algoritmo para determinar os parâmetros, citados na tabela 1 e os respectivos nomes de cada um dos *benchmark's* do Set-A (1995). Para isso, ele lista todas as instâncias do Set-A (1995) e executa cada um deles 10 vezes com os parâmetros representados na tabela 1.

	GA	ACO	PSO
Configuração 1	Iterações: 1000 Soluções: 2000	Iterações: 1000 Formigas: 50	Iterações: 1000
Configuração 2	Iterações: 5000 Soluções: 10000	Iterações: 5000 Formigas: 60	Iterações: 5000
Configuração 3	Iterações: 10000 Soluções: 20000	Iterações: 10000 Formigas: 70	Iterações: 10000

Tabela 1 – Tabela dos parâmetros usados para a execução dos *benchmark's*.

4.1.1 Algoritmo Genético

Fazendo essas execuções usando o algoritmo genético podemos fazer algumas observações. O algoritmo genético obteve o resultado mais próximo do resultado dos três algoritmos. No entanto, também foi o algoritmo que gastou o maior tempo de execução.

A figura 5, mostra um exemplo de resultado do *dataset* A-n32-k5, além disso, a figura 6, mostra a como a função objetivo do algoritmo genético desempenhou durante as iterações. Na figura 5 o ponto em vermelho é considerado o depósito, o local nas quais o veículo tem que voltar para descarregar. Além disso, a figura 7, mostra o diagrama de dispersão do algoritmo genético.

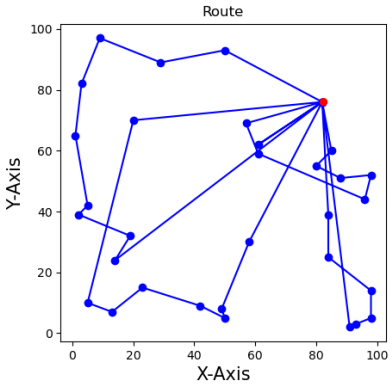

Figura 5 – Grafo de resultado do algoritmo genético para o dataset A-n32-k5.

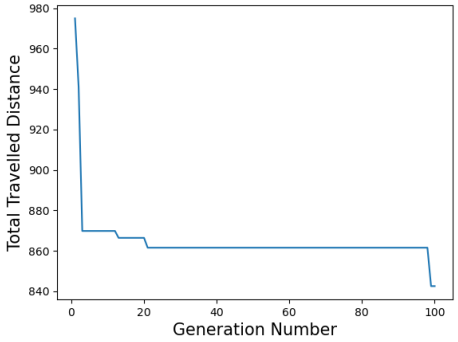

Figura 6 – Gráfico da função objetivo do algoritmo genético.

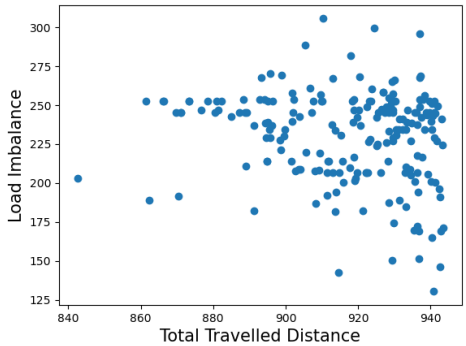

Figura 7 – Gráfico de dispersão do algoritmo genético.

Por fim realizou-se a análise estatística GA e executou o algoritmo a quantidade de vezes para o tamanho da amostra (1537). A tabela 2, mostra os resultados do algoritmo

GA, em geral. Nela podemos ver dados como o melhor valor, a média dos resultados e tempo que demorou cada execução e a qualidade da solução em relação ao resultado ótimo dessa solução. Além disso, usando os parâmetros da segunda execução do GA, como mostra a tabela 1.

Pode-se observar também que o GA foi a abordagem que apresentou a solução de melhor qualidade dentre os três algoritmos, muito embora tenha registrado o maior tempo de execução.

GA				
	Melhor Resultado	Média Resultado	Média Tempo	Qualidade Solução
A-n32-k5.vrp	784	799,21	55,12 s	100%
A-n33-k5.vrp	661	661	54,51 s	100%
A-n33-k6.vrp	742	744,68	55,52 s	100%
A-n34-k5.vrp	778	781,41	56,02 s	100%
A-n36-k5.vrp	799	818,77	56,42 s	100%
A-n37-k5.vrp	669	669,78	55,04 s	100%
A-n37-k6.vrp	949	963,32	61,72 s	100%
A-n38-k5.vrp	730	749,32	55,47 s	100%
A-n39-k5.vrp	822	828,13	58,37 s	100%
A-n39-k6.vrp	831	880,14	58,57 s	100%
A-n44-k6.vrp	937	948,18	61,52 s	100%
A-n45-k6.vrp	944	967,64	60,47 s	100%
A-n45-k7.vrp	1146	1189,81	62,30 s	98,87%
A-n46-k7.vrp	914	962,85	60,27 s	100%
A-n48-k7.vrp	1073	1094,94	61,44 s	98,81%
A-n53-k7.vrp	1010	1013,45	61,10 s	99,76%
A-n54-k7.vrp	1167	1217,11	61,23 s	97,94%
A-n55-k9.vrp	1073	1094,43	60,54 s	98,96%
A-n60-k9.vrp	1354	1360,52	62,12 s	96,72%
A-n61-k9.vrp	1034	1044,21	60,24 s	99,81%
A-n62-k8.vrp	1288	1294,48	61,59 s	97,32%
A-n63-k1.vrp	1314	1318,52	62,45 s	96,32%
A-n63-k9.vrp	1616	1788,82	64,54 s	92,09%
A-n64-k9.vrp	1401	1421,35	62,51 s	95,13%
A-n65-k9.vrp	1174	1226,24	61,42 s	98,23%
A-n69-k9.vrp	1159	1206,77	61,46 s	97,82%
A-n80-k1.vrp	1763	1891,92	65,15 s	90,77%

Tabela 2 – Tabela de resultados do algoritmo GA.

4.1.2 Colônia de Formiga

O segundo algoritmo executado foi o de colônia de formigas. Esse algoritmo teve um desempenho mediano, tanto para encontrar o resultado ótimo, como no seu tempo de execução. A figura 8, mostra um exemplo de resultado do dataset A-n32-k5. Na figura 8 o ponto em vermelho é considerado o depósito, o local nas quais o veículo tem que voltar para descarregar.

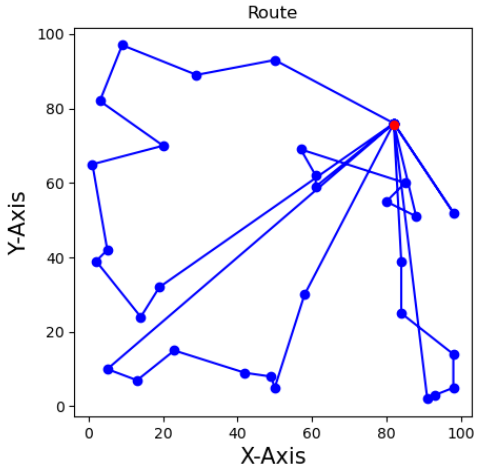

Figura 8 – Grafo de resultado do algoritmo de colônia de formiga para o dataset A-n32-k5.

Então para finalizar a sua execução, o processo do ACO executou o algoritmo a quantidade de vezes para o tamanho da amostra (1537). A tabela 3, mostra os resultados do algoritmo ACO, em geral. Nela podemos ver dados como o melhor valor, a média dos resultados e tempo que demorou cada execução e a qualidade da solução em relação ao resultado ótimo dessa solução. Além disso, usando os parâmetros da segunda execução do ACO, como mostra a tabela 1.

Podemos observar também que no ACO com ele ficou na média quanto ao tempo e também quanto ao resultado. Ele se aproximou mais do resultado do que o PSO, no entanto, não conseguiu manter a qualidade do GA. Podemos observar que mais de 50% das execuções, na grande maioria dos valores, foram aprovados no teste de hipótese.

ACO				
	Melhor Resultado	Média Resultado	Média Tempo	Qualidade Solução
A-n32-k5.vrp	784	819,22	34,23 s	73.40%
A-n33-k5.vrp	661	661,90	29,10 s	95,12%
A-n33-k6.vrp	742	764,62	32,33 s	73.40%
A-n34-k5.vrp	778	811,43	33,13 s	73.40%
A-n36-k5.vrp	799	838,76	37,12 s	73.40%
A-n37-k5.vrp	669	701,21	30,10 s	94,12%
A-n37-k6.vrp	949	998,70	56,13 s	84,84%
A-n38-k5.vrp	730	749,31	31,12 s	89,64%
A-n39-k5.vrp	822	868,64	43,19 s	86,76%
A-n39-k6.vrp	831	980,31	44,57 s	82,89%
A-n44-k6.vrp	937	1018,25	55,57 s	81,93%
A-n45-k6.vrp	944	1027,24	56,11 s	77,94%
A-n45-k7.vrp	1146	1289,83	60,54 s	78,98%
A-n46-k7.vrp	914	1181,77	54,51 s	73.40%
A-n48-k7.vrp	1073	1143,21	60,10 s	72.12%
A-n53-k7.vrp	1010	1194,91	59,13 s	70.91%
A-n54-k7.vrp	1167	1307,43	61,43 s	64.40%
A-n55-k9.vrp	1073	1124,92	60,12 s	71.82%
A-n60-k9.vrp	1354	1460,29	65,10 s	63.32%
A-n61-k9.vrp	1034	1081,32	59,13 s	70.41%
A-n62-k8.vrp	1288	1374,4	62,50	65.17%
A-n63-k1.vrp	1314	1408,26	65,50	60.79%
A-n63-k9.vrp	1616	1755,84	70,55 s	34.10%
A-n64-k9.vrp	1401	1621,37	67,33 s	61.10%
A-n65-k9.vrp	1174	1426,22	60,32 s	55,9%
A-n69-k9.vrp	1159	1381,73	60,20 s	59,12%
A-n80-k1.vrp	1763	2261,12	80,12 s	30.13%

Tabela 3 – Tabela de resultados do algoritmo ACO.

4.1.3 Optimização por enxame de partículas

Por fim, o algoritmo de otimização por partículas. Ele foi o algoritmo que melhor desempenho em tempo, no entanto, foi o algoritmo que pior desempenhou em questão encontrar a melhor solução para o *benchmark*. A figura 9, mostra um exemplo de resultado do dataset A-n32-k5. Na figura 9 o ponto em vermelho é considerado o depósito, o local onde o veículo tem que voltar para descarregar.

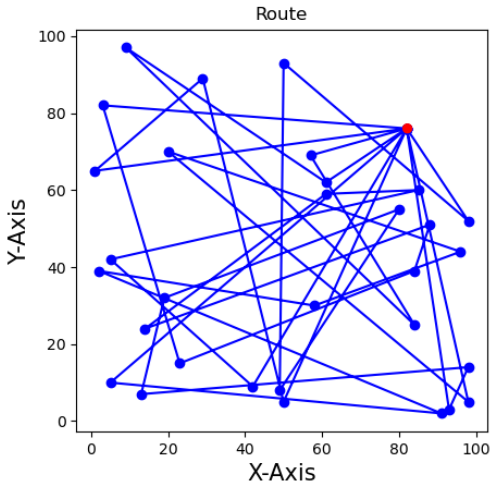

Figura 9 – Grafo de resultado do algoritmo de otimização por partículas para o dataset A-n32-k5.

Então para finalizar a sua execução foi necessário fazer a execução do PSO na quantidade de vezes para o tamanho da amostra (1537). A tabela 4, mostra os resultados do algoritmo PSO, em geral. Nela podemos ver dados como o melhor valor, a média dos resultados e tempo que demorou cada execução e a qualidade da solução em relação ao resultado ótimo dessa solução. Além disso, usando os parâmetros da segunda execução do PSO, como mostra a tabela 1.

Podemos observar que o PSO com essa quantidade de parâmetros executou muito rápido comparado aos outros algoritmos. No entanto, ao observar seu valor médio e os valores que convergiram, observamos que com essa quantidade de iterações não foi o suficiente para ele convergir a ponto de chegar próximo aos valores ótimos. Ao comparar com os outros algoritmos podemos perceber que ele teve o pior desempenho.

PSO				
	Melhor Resultado	Média Resultado	Média Tempo	Qualidade Solução
A-n32-k5.vrp	784	1714,40	2,09 s	0.00%
A-n33-k5.vrp	661	1439,64	2,11 s	0.00%
A-n33-k6.vrp	742	1442,00	2,23 s	0.00%
A-n34-k5.vrp	778	1591,73	2,18 s	0.00%
A-n36-k5.vrp	799	1707,48	2,28 s	0.00%
A-n37-k5.vrp	669	1602,62	2,34 s	0.00%
A-n37-k6.vrp	949	1842,27	2,45 s	0.00%
A-n38-k5.vrp	730	1716,30	2,43 s	0.00%
A-n39-k5.vrp	822	1805,00	2,48 s	0.00%
A-n39-k6.vrp	831	1910,24	2,50 s	0.00%
A-n44-k6.vrp	937	2084,18	2,79 s	0.00%
A-n45-k6.vrp	944	2369,79	2,88 s	0.00%
A-n45-k7.vrp	1146	2241,99	2,87 s	0.00%
A-n46-k7.vrp	914	2187,50	2,95 s	0.00%
A-n48-k7.vrp	1073	2456,71	3,06 s	0.00%
A-n53-k7.vrp	1010	2617,64	3,39 s	0.00%
A-n54-k7.vrp	1167	2791,09	3,43 s	0.00%
A-n55-k9.vrp	1073	2678,06	3,65 s	0.00%
A-n60-k9.vrp	1354	3174,97	3,90 s	0.00%
A-n61-k9.vrp	1034	2679,48	4,03 s	0.00%
A-n62-k8.vrp	1288	3296,13	3,91 s	0.00%
A-n63-k1.vrp	1314	3142,23	4,14 s	0.00%
A-n63-k9.vrp	1616	3627,01	4,06 s	0.00%
A-n64-k9.vrp	1401	3219,18	4,07 s	0.00%
A-n65-k9.vrp	1174	3342,23	4,12 s	0.00%
A-n69-k9.vrp	1159	3352,71	4,29 s	0.00%
A-n80-k1.vrp	1763	4395,59	4,97 s	0.00%

Tabela 4 – Tabela de resultados do algoritmo PSO.

4.1.4 Construção do arquivo de resultados

Por fim, construiu-se um arquivo JSON para compilar os dados obtidos na pesquisa. Nesse arquivo há os seguintes dados: o melhor valor, a média dos resultados e tempo que demorou cada execução, a mediana dos resultados e tempo que demorou cada execução, e seu teste de hipótese. A figura 10 mostra uma parte desse JSON.

```
"A-n46-k7.vrp": {
  "best": 914,
  "PSO": {
    "result": [...],
    "media_result": 2187.502085227667,
    "mediana_result": 2187.502085227667,
    "hipotese": "0.00%",
    "time": [...],
    "media_time": 2.952138303081634,
    "mediana_time": 2.952138303081634
  },
  "ACO": {
    "result": [...],
    "media_result": 1181.7725724962365,
    "mediana_result": 1181.7725724962365,
    "hipotese": "73.40%",
    "time": [...],
    "media_time": 2054.516827986165,
    "mediana_time": 2054.516827986165
  },
  "GA": {
    "result": [...],
    "media_result": 962.8570864841495,
    "mediana_result": 962.8570864841495,
    "hipotese": "100.00%",
    "time": [...],
    "media_time": 3046.27056465117,
    "mediana_time": 3046.27056465117
  }
},
```


Figura 10 – JSON construído como forma de compilação dos dados.

Em suma, podemos observar que o algoritmo genético, em geral, teve um melhor desempenho ao encontrar o resultado ótimo para o dataset. Além disso, foi possível observar pelo grafo gerado que também é o que tem o menor número de voltas ao depósito e também como sua otimização sua função objetivo foi melhorando com o tempo. Além disso, o algoritmo de colônia de formigas teve um desempenho mediano, mas próximo ao genético, com um tempo de execução menor. Por fim, a otimização por partículas, que desempenhou muito bem quanto ao tempo, mas desempenho de muito mal ao buscar a solução ótima do *benchmark*.

4.1.5 Executando em Formiga

O próximo passo ao testar o algoritmo em *benchmark's* é colocar o algoritmo para desempenhar no foco do estudo que seria a cidade de Formiga. Para isso foi criado um

dataset da cidade de Formiga. Para a construção desse dataset usada a biblioteca OSMNX, para construir o grafo dos bairros da cidade. A figura 11 mostra uma foto do bairro Alto dos Pinheiros em forma de grafo. Nesse figura consideramos que todos os resíduos sólidos estão anexados nos nós do grafo, ou seja, eles estão em suas respectivas esquinas. Além disso, foi como não foi possível saber a quantidade exata de lixo consideramos uma quantidade lixo para a coleta uniforme em todos os nós do grafo.

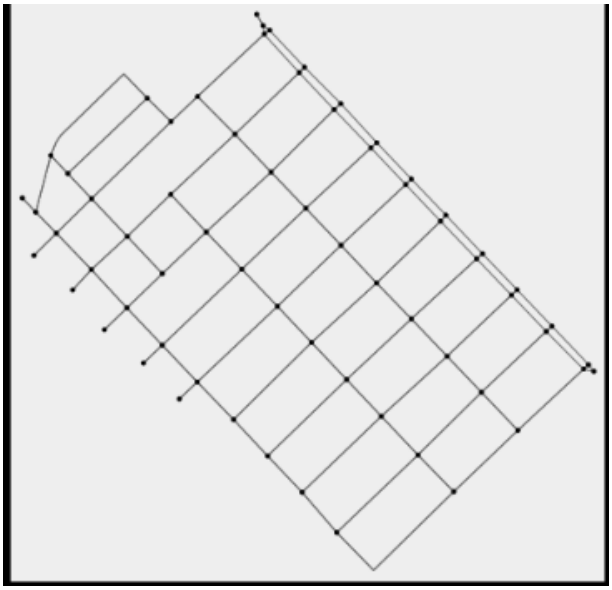

Figura 11 – Versão de grafo do bairro Alto dos Pinheiros.

Após a construção do dataset dos bairros da cidade de Formiga, foram preparados para fazer a execução deles nos algoritmos. No entanto, ainda havia algumas pendências, pois utilizando os algoritmos em benchmarks todos os pontos que a solução consideram eram apenas um plano cartesiano, com pontos em posições x e y. No entanto, ao consideramos uma cidade, não podemos simplesmente calcular a distância entre dois pontos, pois destarte considera-se apenas retamente entre os dois pontos. Quando estamos em uma cidade precisamos considerar possíveis curvas, além disso, precisamos considerar a curva do planeta Terra. Por isso, o cálculo da distância mínima entre os dois pontos foi usando a biblioteca OSMNX, essa biblioteca usa os valores já calculados pelo *OpenStreetMap* para fazer esse cálculo.

Além disso, precisou-se fazer outra manipulação, os *benchmark's* do Set-A (1995) são grafos fortemente conexos, de forma que todos os vértices estavam ligados a todos os outros vértices. A literatura prevê que para o CVRP o grafo deve ser fortemente conexo, então foram feitas as seguintes manipulações.

A primeira seria colocar criar um grafo fortemente conexo e em arestas inexistentes colocaríamos um valor equivalente a infinito para o algoritmo tentar evitar essas arestas. No entanto, essa solução fez com os resultados ficassem ilegíveis, pois algumas vezes o valor infinito não era abandonado durante a evolução.

A segunda solução seria colocar o peso do somatório dos pesos das arestas até local, por exemplo, seria o grafo da figura 12 podemos observar que neste grafo o vértice 1 e o vértice 3 não possuem conexão e essa solução o peso dele seria 10. Isso acontece, pois para chegar no vértice 3 ele tem que passar pelo caminho 1 -> 2 -> 3 ou pelo caminho 1 -> 4 -> 3, pelo primeiro caminho podemos observar que o custo é 10 e pelo segundo o custo é 84, então se escolheria o primeiro caminho. Além disso, passar por um caminho que não deveria existir é colocado uma punição para que ele evite passar novamente por caminhos assim.

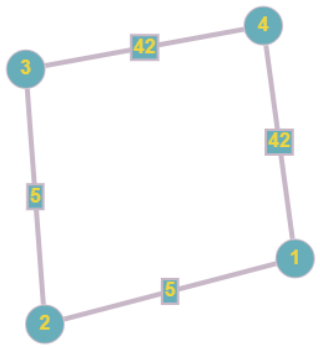

Figura 12 – Grafo de Exemplo.

Então executou o algoritmo genético na cidade de Formiga, com as alterações descritas acimas e utilizando a configuração 2 da tabela 1. Podemos observar que o esse algoritmo teve um desempenho semelhante ao que aconteceu no benchmark. Além disso, consideramos que os pontos de coleta de lixo seriam nas esquinas e elas formaram os vértices de nosso grafo. A figura 13 mostra a melhor solução encontrada entre os algoritmos.

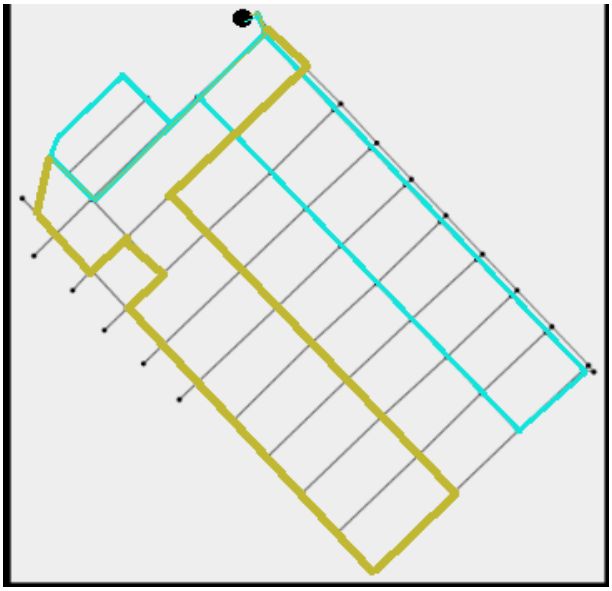

Figura 13 – Resultado do Algoritmo Genético em Formiga.

4.2 API

Nesse projeto nossa api terá como objetivo fazer a comunicação entre os dados das heurísticas e os *frontend mobile*. Para a construção da API foi escolhido o *framework Spring Boot*. Ele é um *framework* de *backend* em Java com o objetivo de simplificar a utilização das bibliotecas do próprio *Spring*. Além disso, ele tem o foco de aumentar a produtividade do programador no desenvolvimento de aplicativos e diminuir a complexidade dos códigos escritos em Java. Segundo o fundador do *Spring Framework* Pivotal Software, Inc. (2021), o *Spring Boot* é uma forma fácil e rápida de fazer a criação de diferentes aplicativos adicionando apenas algumas configurações padrões, como local da conexão com banco e a porta que o servidor deve executar.

A construção do serviço usando o *framework* começa por requisição do tipo *get*, com a *url: api/route* e passando como parâmetros dois pontos da diagonal do quadrado que foi escolhido para ser a área para as heurísticas executarem. Ao fazer é chamado um algoritmo que faz a chamada da meta-heurística. Primeiramente precisamos verificar qual é o sistema operacional, pois sistemas operacionais *UNIX* usam barra como separador, mas o Windows usa contra barra. Feito isso se cria uma *thread* para chamar o *Python* como se estivesse sendo ativado por um terminal. Por fim ler as *strings* que é a saída do programa *Python* e retornar para o aplicativo *mobile*.

4.3 Aplicativo Mobile

O aplicativo *mobile* visa desenhar as rotas visualmente para o usuário. Para a construção do aplicativo mobile foi escolhido o *framework* do Flutter. Ele é um *framework*

de *frontend* para aplicativos *mobile* e atualmente está sendo usado para desenvolvimento de aplicações web e também *desktop*. O Flutter foi desenvolvido e lançado pela Google em 2017. Desde seu lançamento ele vem amplamente adotado pela comunidade mobile, por sua flexibilidade de desenvolvimento, sua eficiência e a produtividade gerada por *framework*.

Além disso, um dos principais pontos dele é a possibilidade de gerar códigos para diferentes plataformas a partir de um único código base. Por exemplo, com o mesmo código ele pode gerar código para: *Mobile Android* e IOS, *Web*, *Desktop* Linux, Mac e Windows. Além disso, ele pode ser compilado em diferentes linguagens em Android Kotlin ou Java e IOS Swift ou Objective-C.

Para a construção do aplicativo foi necessário o uso de algumas bibliotecas do Flutter: flutter_osm_plugin (2022), para fazer a construção do mapa usando o *OpenStreetMap* e geolocator (2022), para fazer a construção de pontos e pegar a localização de pontos específicos da cidade de Formiga. Além disso, para o funcionamento do aplicativo é necessário que o usuário forneça a permissão de usar a localização e identificar seu local atual no mapa.

No algoritmo em Flutter usamos os valores recebidos pela api e os transforma em pontos da biblioteca geolocator (2022). Ao copiar os pontos gerados pelo projeto da api então se converte em pontos, então a biblioteca flutter_osm_plugin (2022) ela constrói a rota baseando vértices enviados para o aplicativo. A figura 14 mostra o resultado final, com a rota criada no aplicativo mobile.

Figura 14 – Resultado do Algoritmo Genético em Formiga.

5 Conclusões

Com base na pesquisa desenvolvida, é possível concluir que a aplicação de técnicas como as meta-heurísticas aplicadas podem ser efetivas para solucionar o problema do lixo urbano na cidade de Formiga. No entanto, é importante salientar necessário que para podermos aplicar essas técnicas e ver sua efetividade total precisamos de uma disponibilidade de dados que sejam precisos e confiáveis advindos da prefeitura local.

Segundo Zhang et al. (2017), a utilização de algoritmos genéticos para otimizar a coleta em regiões urbanas pode ser bastante eficiente. Isso faz com que a quantidade de lixo das cidades seja diminuída, além de também refletir em uma diminuição na quantidade da quantidade de lixo que não consegue ser coleta por superlotação dos veículos.

O algoritmo de colônia de formigas também é uma abordagem promissora para resolver o problema da coleta de lixo. Segundo Dorigo e Gambardella (1999), ele permite com que os veículos que fazem a coleta de lixo percorrer as rotas de forma mais eficiente de forma que seja reduzido o tempo da coleta.

A abordagem usando a técnica enxame de partículas, mesmo não apresentando bons resultados, pode ser promissora se combinada com a técnica de programação linear. Kennedy e Eberhart (1995) menciona que tal abordagem apresenta bons resultados.

Os experimentos realizados permitiram concluir que é factível propor rotas eficientes para coleta de lixo, usando as abordagens mencionadas. Também observou-se que os algoritmos convergiram em um tempo viável e os melhores resultados foram obtidos com a abordagem usando algoritmo genético. Importa registrar que esta abordagem usando algoritmo genético foi que a demandou mais tempo de execução. A abordagem utilizando enxame de partículas foi a menos promissora e não convergiu em nenhum dos casos de teste. A abordagem utilizando colônia de formiga apresentou resultados intermediários, situando-se melhor que a abordagem usando enxame de partículas e pior que algoritmos genéticos. Tais resultados corroboram e mantém-se em linha com o observado na literatura sobre o problema de coleta de resíduos sólidos. Por fim, é importante registrar que foi possível construir rotas em alguns bairros da cidade de Formiga e transpor-los para o aplicativo mobile em flutter por meio de uma API.

Em relação ao aplicativo é importante ressalvar a necessidade de realizar uma melhor validação. Há de se realizar testes com dados mais precisos e detalhados da cidade de Formiga. É importante considerar dados como densidade demográfica do local, quantidade de lixo média por ponto de coleta, a infraestrutura de transporte da cidade, entre outros. Esses são dados são necessários para que se possa confirmar totalmente a acurácia desses algoritmos e permitir afirmar com certeza que as rotas proposta por essa aplicação faz com

que os veículos coletores de resíduos sólidos estão passando pelo caminho mais eficiente.

6 Trabalhos futuros

Como trabalho futuro propõe avaliar o desempenho de diferentes meta-heurísticas em cenários de CVRP com diferentes graus de complexidade, tais como variação no número de veículos, variação no número de clientes, variação na capacidade dos veículos, entre outros. Além disso, será avaliado o desempenho das meta-heurísticas em instâncias reais de CVRP, comparando-as com soluções encontradas por algoritmos exatos.

Adicionalmente sugerem-se os seguintes tópicos como trabalho futuros:

1. Testar rotas em diferentes bairros da cidade de Formiga e também outras cidades para aferir sua eficácia;
2. Incorporar outras técnicas de otimização, como a programação linear, em modelos de coleta de lixo otimizada para melhorar ainda mais a eficiência;
3. Desenvolver sistemas inteligentes que possam prever com precisão as quantidades de lixo geradas em diferentes áreas e otimizar a coleta com base nessas informações.
4. Construir um separador de bairros para que a meta-heurística possa ser focada na resolução do problema em uma área previamente melhorada
5. Monitorar o desempenho do aplicativo e testar sua usabilidade;

Referências

1AFORMIGA. *Gerenciamento de resíduos*. 2018. Acesso em 30/01/2022. Disponível em: <https://1aformiga.com.br/#gerenciamento>. Citado na página 15.

AUGERAT, P.; CORDEAU, J.-F.; LAPORTE, G. An augmented lagrangian relaxation heuristic for the capacitated vehicle routing problem. *Operations Research*, INFORMS, v. 43, n. 3, p. 395–404, 1995. Citado na página 36.

BHAT, V. N. A model for the optimal allocation of trucks for solid waste management. *Waste Management & Research*, Sage Publications Sage CA: Thousand Oaks, CA, v. 14, n. 1, p. 87–96, 1996. Citado na página 14.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, v. 35, n. 3, p. 268–308, 2003. Citado na página 22.

BOEING, G. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, Elsevier, v. 65, p. 126–139, 2017. Citado 2 vezes nas páginas 33 e 34.

COELLO, C. A. C. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, v. 191, p. 1245–1287, 2002. Citado na página 20.

DORIGO, M. Ant colony system: A cooperative learning approach to the traveling salesman problem. 1992. Disponível em: <https://www.semanticscholar.org/paper/Ant-System%3A-Optimization-by-a-Colony-of-Cooperating-Dorigo-Maniezzo/67d7d11797f05f4e5b881ca60ff695a6f6d48fd6>. Citado na página 21.

DORIGO, M.; GAMBARDELLA, L. M. Ants system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 1, n. 1, p. 53–66, 1999. Citado na página 50.

DORIGO, M. et al. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 1, n. 1, p. 53–66, 1997. Citado na página 22.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, IEEE, v. 26, n. 1, p. 29–41, 1996. Citado na página 21.

DORIGO, M.; STüTZLE, T. *Ant colony optimization*. [S.l.]: MIT press, 2004. Citado 2 vezes nas páginas 21 e 22.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. [S.l.: s.n.], 1995. p. 39–43. Citado 2 vezes nas páginas 23 e 24.

EIBEN, A. E.; SMITH, J. E. *Introduction to Evolutionary Computing*. [S.l.]: Springer, 2015. Citado na página 20.

ELSOM1945. *Routing-problem–CVRP*. 2022. Acesso em 01 de fevereiro de 2022, às 14:31. Disponível em: <https://github.com/Ellsom1945/Routing-problem--CVRP/>. Citado 2 vezes nas páginas 32 e 33.

EULER, L. *Solutio problematis ad geometriam situs pertinentis*. [S.l.]: Lausanne, 1736. Citado na página 31.

FLUTTER_OSM_PLUGIN. *flutter_osm_plugin*. 2022. <https://pub.dev/packages/flutter_osm_plugin>. Accessed on 10/11/2022 at 19:23. Citado na página 48.

GAD, A. F. Osmnx: A python package to work with graph-theoretic openstreetmap street networks. *Journal of Open Source Software*, The Open Journal, v. 5, n. 53, p. 2707, 2020. Citado na página 34.

GAREY, M. R. Computers and intractability: A guide to the theory of np-completeness. *Revista Da Escola De Enfermagem Da USP*, v. 44, n. 2, p. 340, 1979. Citado na página 18.

GEOLOCATOR. *geolocator*. 2022. <https://pub.dev/packages/geolocator>. Accessed on 10/11/2022 at 19:43. Citado na página 48.

GLOVER, F.; LAGUNA, M. *Tabu search*. [S.l.]: Springer, 1997. Citado na página 20.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. [S.l.]: Addison-Wesley, 1989. Citado 2 vezes nas páginas 19 e 20.

GREENHALGH, T.; PEACOCK, R. *How to read a paper: the basics of evidence-based medicine*. [S.l.]: John Wiley & Sons, 2014. Citado na página 17.

HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. [S.l.]: University of Michigan Press, 1975. Citado 2 vezes nas páginas 19 e 20.

HWANG, H.; KIM, D. A hybrid genetic algorithm for the vehicle routing problem. *Expert Systems with Applications*, Elsevier, v. 37, n. 5, p. 3758–3766, 2010. Citado na página 20.

IBGE. *Pesquisa Nacional de Saneamento Básico 2012: Características do Lixo Urbano e do Manejo dos Resíduos Sólidos Municipais*. Rio de Janeiro, RJ, 2012. Disponível em: <https://biblioteca.ibge.gov.br/index.php/biblioteca-catalogo?view=detalhes&id=60583>. Citado na página 7.

JANSON, S.; MIDDENDORF, M.; R"oNNQVIST, M. Particle swarm optimization for the vehicle routing problem with time windows. *Swarm Intelligence*, Springer, v. 4, n. 3, p. 207–226, 2010. Citado na página 24.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, v. 4, p. 1942–1948, 1995. Citado 3 vezes nas páginas 23, 24 e 50.

KIM, B.-S.; KIM, S.-K.; KIM, K.-H. A comparative study of metaheuristics for the vehicle routing problem with time windows. *Computers & Industrial Engineering*, Elsevier, v. 94, p. 163–175, 2016. Citado na página 25.

KONOWROCKI, P. *CVRP_ACO*. 2022. Acesso em 14 de fevereiro de 2022, às 14:50. Disponível em: <https://github.com/pkonowrocki/CVRP_ACO>. Citado na página 32.

LAPORTE, G. The vehicle routing problem: an overview of exact and approximate algorithms. *European Journal of Operational Research*, v. 59, n. 3, p. 345–358, 1992. Citado na página 18.

LAPORTE, G.; NOBERT, Y. Routing and scheduling of vehicles and crews: the state of the art. *Computers & Operations Research*, Elsevier, v. 14, n. 1, p. 63–75, 1987. Citado na página 18.

LI, D.; CAO, J. Research on the ant colony optimization algorithm for vehicle routing problem. *Journal of Physics: Conference Series*, v. 1093, n. 1, p. 012053, 2018. Citado na página 22.

MARTELLO, S.; TOTH, P. The vehicle routing problem with time windows. *Operations Research*, INFORMS, v. 38, n. 6, p. 860–876, 1990. Citado na página 29.

MATLOFF, N. *Statistical Programming with R*. [S.l.]: Chapman and Hall/CRC, 2018. ISBN 978-1138505535. Citado na página 34.

MITCHELL, M. *An Introduction to Genetic Algorithms*. [S.l.]: MIT Press, 1996. Citado 2 vezes nas páginas 19 e 20.

MITCHELL, M. *An Introduction to Genetic Algorithms*. [S.l.]: MIT Press, 1998. Citado na página 19.

OPENKNOWLEDGE. *Open Database License (ODbL)*. 2015. Disponível em: <https://opendatacommons.org/licenses/odbl/>. Citado na página 33.

OPENSTREETMAP. *OpenStreetMap - Copyright e licença*. 2023. Disponível em: <https://www.openstreetmap.org/copyright>. Citado na página 33.

ORçAMENTO, M. do Planejamento e. *Manual de Organização dos Serviços de Limpeza Pública para Comunidades de Pequeno Porte*. Brasília, DF: Ministério do Planejamento e Orçamento, 2006. Citado na página 14.

Pivotal Software, Inc. *Spring Boot Reference Documentation*. San Francisco, CA, 2021. Disponível em: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>. Citado na página 47.

RABUSKE, U. *Introdução à teoria dos grafos*. [S.l.]: Editora da UFSC, 1992. Citado na página 30.

RAMOS, N. Algoritmo guloso randomizado para o problema de alocação de registradores. Instituto de Federal de Minas Gerais - Campus Formiga, 2016. Citado na página 34.

SANTOS, J. A. dos. Modelagem de malhas viárias urbanas aplicando conceitos de grafos. 2006. Citado na página 31.

SET-A. *Vehicle Routing Problem Repository*. 1995. Accessed on 25/04/2022 at 16:24. Disponível em: <http://www.vrp-rep.org/datasets/item/2014-0000.html>. Citado 3 vezes nas páginas 28, 36 e 45.

SHEN, X.; WU, J. Hybrid particle swarm optimization and genetic algorithm for the vehicle routing problem with time windows. *Expert Systems with Applications*, Elsevier, v. 41, n. 2, p. 612–623, 2014. Citado na página 25.

STUTZLE, T.; HOOS, H. H. Ant colony optimization applied to the quadratic assignment problem. *INFORMS Journal on Computing*, INFORMS, v. 13, n. 4, p. 347–369, 2000. Citado na página 22.

STüTZLE, T.; HOOS, H. H. Ant colony optimization. *IEEE Computational Intelligence Magazine*, IEEE, v. 1, n. 4, p. 28–39, 2000. Citado na página 22.

TOTH, P.; VIGO, D. *The vehicle routing problem*. [S.l.]: Society for Industrial and Applied Mathematics, 2002. Citado 3 vezes nas páginas 18, 19 e 21.

TOTH, P.; VIGO, D. *Vehicle routing: problems, methods, and applications*. [S.l.]: SIAM, 2014. Citado na página 31.

UZKENT, B.; BALCIK, B. A survey on the vehicle routing problem and its variants in logistics. *Computers & Industrial Engineering*, Elsevier, v. 90, p. 554–570, 2015. Citado na página 20.

VARSHEY, N. *CVRP_USING_PSO_GA*. GitHub, 2021. Acesso em 14/02/2022 14:29. Disponível em: <https://github.com/niharikavars/CVRP_USING_PSO_GA/>. Citado na página 31.

VIEIRA, M. R. A. Heurística matemática aplicada ao problema da coloração de grafos. 2018. Citado na página 31.

WANG, C.; LI, S. Hybrid fruit fly optimization algorithm for solving multi-compartment vehicle routing problem in intelligent logistics. *Advances in Production Engineering & Management*, University of Maribor, Faculty of Mechanical Engineering, Production . . . , v. 13, n. 4, p. 466, 2018. Citado na página 18.

WHITLEY, D. A genetic algorithm tutorial. *Statistics and Computing*, v. 4, n. 2, p. 65–85, 1994. Citado na página 20.

WICKHAM, H.; GROLEMUND, G. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. [S.l.]: O'Reilly Media, 2017. ISBN 978-1491910749. Citado na página 34.

XAVIER, R. de S. Modelagem e minimização do consumo de combustível para rotas de coleta de lixo. Universidade Federal de Minas Gerais, 2010. Citado 2 vezes nas páginas 14 e 15.

ZHANG, B. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, Elsevier, v. 65, p. 126–139, 2018. Citado 2 vezes nas páginas 33 e 34.

ZHANG, L. et al. Genetic algorithm-based optimization of municipal solid waste collection: A case study in china. *Waste Management*, Elsevier, v. 67, p. 3–11, 2017. Citado na página 50.