UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

HUGO DE OLIVEIRA ALVES

SIMULAÇÃO DE ROTAS PARA A DISTRIBUIÇÃO DE URNAS ELETRÔNICAS NO MUNICÍPIO DE SÃO PAULO, UTILIZANDO ALGORITMO GENÉTICO.

LONDRINA

2023

HUGO DE OLIVEIRA ALVES

SIMULAÇÃO DE ROTAS PARA A DISTRIBUIÇÃO DE URNAS ELETRÔNICAS NO MUNICÍPIO DE SÃO PAULO, UTILIZANDO ALGORITMO GENÉTICO.

Simulation of routes for the distribution of electronic voting boxes in the municipality of São Paulo, using genetic algorithm

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Produção da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientador(a): Fernando Henrique Campos.

LONDRINA
2023

HUGO DE OLIVEIRA ALVES

SIMULAÇÃO DE ROTAS PARA A DISTRIBUIÇÃO DE URNAS ELETRÔNICAS NO MUNICÍPIO DE SÃO PAULO, UTILIZANDO ALGORITMO GENÉTICO.

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Produção da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 27/novembro/2023.

_____

Fernando Henrique Campos.
Doutorado.
Universidade Tecnológica Federal do Paraná

_____

Luís Fernando Cabeça.
Doutorado
Universidade Tecnológica Federal do Paraná

_____

Rodrigo Libanês Melan
Mestrado
Universidade Tecnológica Federal do Paraná

LONDRINA
2023

RESUMO

O presente trabalho teve como objetivo a melhoria no processo de roteirização de veículos para a distribuição de urnas eletrônicas destinadas as zonas eleitorais do município de São Paulo, por meio de um estudo de casos. A pesquisa desenvolveu-se a partir da coleta de dados, referente aos respectivos endereços das zonas eleitorais localizadas no município analisado. Para a realização deste trabalho foram coletados 58 endereços distintos, que foram relacionados dentro de uma matriz de distâncias. Essa matriz foi organizada para relacionar as distâncias entre todos os endereços coletados, e serviu como base para a implementação do algoritmo proposto nesse estudo. O objetivo do algoritmo foi simular um conjunto de rotas que minimizasse a distância total percorrida para atender todos os pontos de distribuição necessários, para isso, foi desenvolvido um código computacional por meio da estrutura de um algoritmo genético, programado a partir da linguagem Python, utilizando a IDE (Integrated Development Environment) Pycharm. Além disso, também foi utilizada a ferramenta computacional MS-Excel®, para efeito de comparação das soluções. Ao término, os resultados obtidos são exibidos, evidenciando a eficácia e as potenciais aplicações do algoritmo desenvolvido na pesquisa.

Palavras-chave: Algoritmos Genéticos, Meta-Heurísticas, Problema de Roteamento de Veículos

ABSTRACT

The present work aimed to improve the vehicle routing process for the distribution of electronic voting machines to the electoral zones of the municipality of São Paulo, through a case study. The research was developed from data collection, regarding the respective addresses of the electoral zones located in the analyzed municipality. In total, 58 distinct addresses were collected, which were related within a distance matrix. This matrix was organized to relate the distances between all collected addresses and served as the basis for implementing the algorithm proposed in this study. The objective of the algorithm was to simulate a set of routes that would minimize the total distance traveled to serve all necessary distribution points. For this, a computer code was developed through the structure of a genetic algorithm, programmed from the Python language, using the Pycharm IDE (Integrated Development Environment). In addition, the computational tool MS-Excel® was also used for comparison purposes of the solutions. Finally, the results obtained are presented, demonstrating the efficiency and possible applicability of the proposed algorithm in the research.

Keywords: Genetic Algorithms, Metaheuristics, Vehicle Routing Problem**.**

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AG	Algoritmo genético
GPS	Global Positioning System,
IDE	Integrated Development Enviroment
MS	Microsoft
PCV	Problema Caixeiro Viajante
PRPV	Problemas de Roteirização e Programação de Veículos
TRE	Tribunal Regional Eleitoral
TSE	Tribunal Superior Eleitoral
TSP	Traveling Salesman Problem
UE	Urnas Eleitorais

SUMÁRIO

1. INTRODUÇÃO

De acordo com Novaes (2007), a roteirização de veículos faz parte da dinâmica de vida contemporânea, compreendendo desde os setores logísticos, que realizam a programação e o controle de frotas para a distribuição de bens ou prestação de serviços, ou até mesmo, necessidades pessoais, como por exemplo, viajar com a família ou ir ao trabalho.

Por definição, a roteirização é um procedimento utilizado para a simulação de rotas, definindo um sequenciamento de paradas a serem cumpridas dentro de um percurso, realizado por um ou múltiplos veículos de frota, tendo como objetivo minimizar os custos de deslocamento, dentro de um conjunto de pontos geograficamente dispersos em locais previamente definidos, que necessitam ser atendidos (CUNHA, 2000).

Sendo assim, aprimorar a eficiência nos processos de roteirização por meio da simulação de rotas de forma adequada, certamente resultará em ganhos operacionais, reduzindo custos de transporte e tempo de serviço. Desse modo, um modelo de roteirização bem aplicado, traz inúmeras vantagens competitivas não só para o setor empresarial, mas também para as instituições públicas que necessitem desse tipo de apoio (CUNHA, 2000).

Segundo o Tribunal Superior Eleitoral (TSE), a lei orçamentária anual (LOA) autorizou um montante de R$ 1,28 bilhão para custear o processo eleitoral da última eleição vigente no Brasil, realizada no ano de 2020. Ainda de acordo com dados da mesma fonte, o orçamento da Justiça Eleitoral para as eleições municipais foi de R$ 106,6 milhões (transporte, R$ 41,3 milhões, e apoio operacional, R$ 64,8 milhões). Totalizando cerca de 556 mil urnas eletrônicas, distribuídas nos 5.570 municípios que compõem o território nacional.

Ainda de acordo com o TSE, o transporte, o armazenamento, a conservação e a preparação das urnas eletrônicas no processo eleitoral, são de competência e responsabilidade das autoridades públicas. Para determinar um sistema de rotas de distribuição de urnas eletrônicas, em seus respectivos locais de votação é necessário planejamento, e assim definir e otimizar os trajetos percorridos da maneira mais eficiente possível. Para isso, é essencial deliberar as melhores rotas e planejar um itinerário de frota de maneira pragmática. Para situações de

maior complexidade, como no caso da distribuição de urnas eletrônicas, a modelagem computacional torna-se um recurso imprescindível.

1.1 Objetivo geral

O objetivo geral deste trabalho consiste em propor soluções para um problema prático de roteirização relacionado à distribuição de urnas eletrônicas no município de São Paulo, empregando abordagens computacionais

.

1.2 Objetivos específicos

- Aplicar ferramentas computacionais para simular a resolução de problemas práticos;
- Desenvolver uma meta-heurística em linguagem Phyton por meio de um algoritmo genético, para testar a sua validade em uma situação real;
- Demostrar os resultados obtidos através de um panorama visual, com o auxílio de gráficos e tabelas;
- Comparar os métodos aplicados na linguagem de programação em Python com os métodos aplicados no Microsoft Excel (Solver)

1.3 Contextualização do problema

Segundo dados divulgados no último censo (2020) pelo tribunal superior eleitoral (TSE), existem cerca de 147.918.483 de eleitores aptos a exercerem o seu poder de voto, dentro do território nacional. Ainda de acordo com dados da mesma fonte, só no município de São Paulo existem aproximadamente cerca de 8.986.687 de eleitores.

Para suprir a demanda eleitoral durante o período das eleições, as urnas eletrônicas (UE), principal instrumento do processo democrático, de acordo com as normas estabelecidas pelo TSE, são armazenadas nos tribunais regionais eleitorais de cada município existente, e posteriormente são redistribuídas para

os cartórios eleitorais de cada entidade federativa do país. No estado de São Paulo, por exemplo, segundo relatório divulgado pelo Tribunal Regional Eleitoral SP (TRE-SP) foram distribuídos, na última eleição (2020), um montante de 36.711(UE), sendo que desse total, 22.399 foram direcionadas à capital do estado.

Com base nos números divulgados pelo TRE-SP e pelo TSE, é notório que o panorama logístico de distribuição e organização dos eventos eleitorais no país, acaba tornando-se um desafio. Neste sentido, para que as eleições ocorram de forma plena, é necessário sistematizar todos aspectos organizacionais desse evento, sendo que um dos principais desafios neste contexto é a distribuição de forma eficaz, das urnas eletrônicas nos locais de votação.

Dessa forma, como ponto de partida podemos fazer o seguinte questionamento: **De que modo a aplicação da modelagem computacional pode aprimorar a eficácia da distribuição de urnas eletrônicas no município de São Paulo?**

1.4 Justificativa

Definir o melhor percurso, que reduzam os custos logísticos e operacionais é um desafio crucial para qualquer instituição pública ou privada. Em situações em que se pretende realizar operações de distribuição de um determinado bem, é necessário quantificar todos os parâmetros que possam interferir na logística operacional, e com isso, aferir rotas de destinos aos postos que serão trabalhados (SOLIANI et al., 2020).

A organização dos pleitos eleitorais segundo o TSE, requer grande envolvimento dos órgãos responsável pela sua realização, e para que isso ocorra de forma eficiente, é necessário sistematizar todas as operações inerentes ao processo eleitoral. Sendo assim, uma das principais operações nesse sentido, é a distribuição das urnas eletrônicas (UE) em seus respectivos postos de votação. Dessa forma, para que esse processo de distribuição ocorra de forma correta, ou seja, com o menor custo operacional e no menor tempo possível, se faz jus os estudos aplicados de roteirização de veículos para a situação explicitada.

Lourenço et al. (2010), afirma que problemas de roteirização de veículos de distribuição, requer em suas soluções, um tempo de processamento computacional normalmente alto, e que se expande de forma exponencial, de acordo com o seu grau de complexidade. Esse tipo de modelagem matemática é categorizado como um problema NP-Difícil. Para problemas dessa categoria, é inviável a utilização de métodos exatos para a sua resolução, sendo necessário a aplicação de abordagens heurísticos ou meta-heurísticas. Como exemplos de métodos heurístico e meta-heurístico, podemos citar os algoritmos genéticos (HOLLAND, 1992), o algoritmo colônia de formigas (DORIGO,1997), o algoritmo Busca Tabu (GLOVER 1990) entre outros modelos, que visam a busca por soluções que se aproximam do melhor resultado possível.

Sendo assim, ao analisarmos as informações supracitadas, podemos observar que, dada a imensa demanda de UE na capital paulista, a logística de distribuição das urnas, entre o tribunal regional eleitoral até as suas respectivas zonas eleitorais, acaba tornando-se um problema complexo, e para que todo esse processo ocorra de forma efetiva, fica explicito a necessidade de avaliarmos rotas que otimizem a distribuição dessas urnas nos seus respectivos locais de entrega. Segundo Bodin (1990) em cenários como esse, é importante o uso de técnicas de roteirização de veículos

Além dos questionamentos levantados, o presente trabalho também tem como objetivo contribuir em benefício à sociedade, dado que o desenvolvimento deste estudo tem como finalidade otimizar atividades operacionais, que tornam os processos dos pleitos eleitorais mais ágeis, facilitando a disposição das urnas eletrônicas nos seus respectivos locais de votação, e com isso contribuir para a redução das despesas públicas.

1.5 Delimitação da pesquisa

A pesquisa realizada neste trabalho delimita-se na aplicação de ferramentas computacionais utilizando meta-heurística em linguagem Python, além do software Microsoft Excel® com o complemento solver, para simular rotas para a distribuição de urnas eletrônicas em cartórios eleitorais, tendo como foco de aplicação o município de São Paulo.

De acordo com o Tribunal Regional Eleitoral de São Paulo (TRE-SP) município da capital paulista, conta com 58 cartórios eleitorais, também conhecidos como zonas eleitorais, aos quais são recebidas as urnas eletrônicas. Nesses cartórios as urnas são preparadas e aferidas, com testes de carga, verificação de componentes, teclado, biometria, som e visualização, para posteriormente serem redistribuídas para as seções eleitorais, que são os locais propriamente ditos de votação. Ainda de acordo com dados da mesma fonte, o município de São Paulo conta com 26.145 seções eleitorais, espalhadas por 2.049 postos de votação, além de 8.986.687 de eleitores

Sendo assim, a pesquisa realizada delimita-se na distribuição de UE entre o tribunal regional eleitoral de São Paulo, que se encontra no centro da cidade analisada, até as suas respectivas zonas eleitorais, que se encontram distribuídas entorno do município, totalizando 58 pontos de entregas, de acordo com dados coletados no site do Tribunal Regional Eleitoral de São Paulo (TRE-SP)

1.6 Ambientação da pesquisa

A pesquisa realizada utilizou dados obtidos através da plataforma digital do Tribunal Regional Eleitoral de São Paulo (TRE-SP), com foco de pesquisa nas zonas eleitorais do município de São Paulo.

O (TRE) é um dos principais órgãos públicos responsáveis pelos pleitos eleitorais, desde a sua organização por completo, até a contagem e apuração efetiva dos votos. As bases de dados do TRE, são bases de dados de fonte aberta para consulta pública, disponíveis no seu respectivo endereço eletrônico, onde foram coletados todos os dados pertinentes a este estudo acadêmico.

1.7 Característica da pesquisa

Segundo Cooper (2011), a pesquisa científica pode ser caracterizada por meio de diversos elementos, incluindo a abordagem adotada, os objetivos delineados, os procedimentos empregados e a finalidade almejada. Esses aspectos são exemplificados de forma visual na Figura 1.

Figura 1: Caracterização da pesquisa

Fonte: Adaptado pelo autor (2022)

O enquadramento da pesquisa realizada neste estudo está devidamente demarcado na Figura 1. Com relação a finalidade do estudo, podemos classificá-lo como sendo um estudo aplicado, uma vez que o intuito é formular soluções a partir de dados e ferramentas aplicadas. De acordo com Cooper (2011) a pesquisa aplicada tem como característica a aplicação, utilização e consequências práticas do conhecimento produzido.

No âmbito da abordagem, essa pesquisa caracteriza-se como quantitativa, uma vez que é aplicado a modelagem matemática computacional para expor causas de um fenômeno que relaciona diversas variáveis. A pesquisa quantitativa é um conjunto de técnicas que procura descrever, decodificar, traduzir e, quantificar o significado de uma análise. A abordagem quantitativa, dessa forma, é responsável pelo estudo do uso e coleta de uma variedade de materiais empíricos sobre a égide de métodos, como por exemplo, um estudo de caso ou uma pesquisa de campo.(COOPER; SCHINDLER, 2011).

Quanto aos objetivos, a pesquisa desenvolvida pode ser classificada como sendo descritiva, pois adota relações entre variáveis, população e fenômenos, além de utilizar técnicas padronizadas que descrevem a situação analisada. De acordo com McCombes (2019), A pesquisa descritiva tem como objetivo fornecer uma representação precisa e sistemática de uma população, situação ou fenômeno. Ela tem a capacidade de abordar indagações relacionadas ao quê, onde, quando e como, mas não se destina a responder questões relacionadas ao porquê.

Em relação aos procedimentos, Cooper (2011) descreve que a pesquisa científica pode ser classificada como sendo: um estudo de caso; pesquisa bibliográfica; documental; participante; ou experimental. A pesquisa desenvolvida nesse trabalho, pode ser então classificada como sendo um estudo de caso, uma vez que é aplicada diretamente em uma entidade federativa (zonas eleitorais do município de São Paulo). De acordo com Yin (2010), um estudo de caso pode ser definido da seguinte forma:

> O estudo de caso é uma investigação empírica que analisa um fenômeno contemporâneo (o caso) em profundidade e em seu contexto de mundo real, especialmente quando os limites entre o fenômeno e o contexto não são claramente evidentes. Caracteriza-se por ser um estudo detalhado e exaustivo de poucos, ou mesmo de um único objeto (YIN, 2010, p. 39)

1.8 Estrutura do trabalho

Este trabalho está estruturado da seguinte maneira: No primeiro capítulo, apresenta-se o contexto da pesquisa, ressaltando os objetivos proposto, a situação-problema, as delimitações, a metodologia aplica na pesquisa, bem como a justificativa da sua realização.

Em seguida, o Capítulo 2 aborda o referencial teórico, que discute os conceitos de roteirização de veículos, suas finalidades, aplicações e variantes, além de contextualizar as abordagens heurística e meta-heurística para a resolução de problemas de otimização, por fim é apresentado o algoritmo genético, com a explicação integral da sua estrutura computacional.

O Capítulo 3 apresenta a forma de como foram obtidos os dados inerentes ao problema, os softwares utilizados, as ferramentas aplicadas e também a abordagem dos parâmetros utilizados no algoritmo desenvolvido.

No Capítulo 4 demostra-se os resultados obtidos, com base na metodologia discutida na seção anterior.

No Capítulo 5 são formalizadas as considerações finais, destacando as contribuições dos estudos, problemas enfrentados, resultados alcançados e pesquisas futuras.

2. REFERENCIAL TEÓRICO

Nessa seção serão abordas algumas definições a respeito dos problemas de roteirização de veículos, métodos heurísticos, algoritmo genético e alguns trabalhos correlacionados.

2.1 Roteirização de veículos

Os problemas de roteirização de veículos também conhecidos como Vehicle Routing Problems (VRPs), são de caráter combinatório, pertencentes a uma área ampla da pesquisa operacional, denominados como problemas de otimização de rede. Nesse contexto, existem diversos problemas clássicos, tais como: o problema do caminho mínimo, problema de transporte, problema de designação, problema do caixeiro viajante, entre outros. (GOLDEN; BALL; BODIN, 1981).

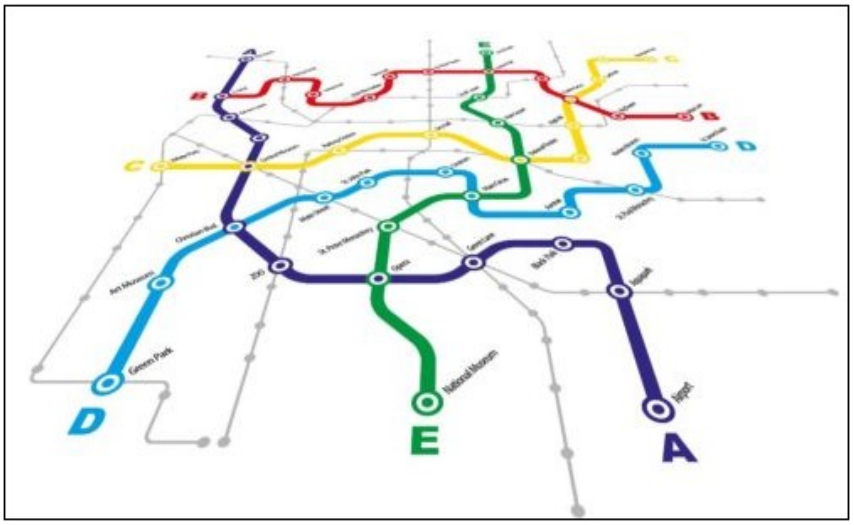
Segundo Bodin (1990), os problemas de roteirização de veículos (VRP) tem como meta encontrar um conjunto de rotas, para minimizar os valores monetários de operação, sendo os custos de transporte mais comuns: a distância total percorrida, o custo total de viagem, encargos rodoviários, valores do combustível, entre outros.

Ainda de acordo Bodin (1990), as variáveis básicas de restrições dentro de um modelo de roteirização (VRP) são: número de veículos da frota; características da demanda; garagem dos veículos; tipo de frota; localidades dos postos de distribuição; limitação de capacidade dos veículos; parâmetros da rede; requisitos de pessoal; tipos de operações relacionadas; tempos estimados de rotas; características particulares da rota, custos e finalidades. Também se trabalha com variáveis de restrições, que podem surgir de acordo com o tipo de problema abordado.

Conforme Taillard (1999), a dinâmica do VRP se da seguinte maneira: primeiramente se inicia a rota de distribuição em um ponto de origem, e finaliza o trajeto no mesmo ponto de início, de forma que a demanda de todos os pontos seja atendida, os pontos de atendimentos são denominados como nós, em um sistema de grafos, cada nó é visitado apenas uma vez, por um único veículo, e a capacidade de cada veículo é limitada. Outras formulações apresentam diferentes restrições, tais como: tempo máximo de viagem, vários objetivos,

janelas de tempos, tamanho da frota, entre outras, que serão contextualizadas nos tópicos subsequentes. A Figura 2 a seguir apresenta um modelo de simulação de rotas para múltiplos veículos.

Figura 2: Simulação de rotas para múltiplos veículos.



Fonte: Adaptado de Silva (2010)

2.1.1 Modelos de roteirização.

A roteirização de veículos agrega um grande conjunto de situações, com variados tipos de problemas. Segundo Bondin et al. (1983), os modelos de roteirização podem ser do tipo: Combinados de roteirização, programação e roteirização pura. Para as aplicações de roteirização pura, variáveis condicionais de tempo não são consideradas na definição do sequenciamento de rotas (coletas ou entregas). Nesse caso, as estratégias para soluções são condicionadas apenas aos aspectos geográficos de localização dos postos de atendidos. A tabela a seguir sintetiza os principais métodos de roteirização pura, segundo Bodin et al. (1983).

Quadro 1: Principais modelos de roteirização pura

Denominação	número de roteiros	localização dos clientes	limite de capacidade nos veículos	número de bases	demandas
Problema do caixeiro viajante	um	nós	não	uma	determinísticas
Problema do carteiro chinês	um	arcos	não	uma	determinísticas
Problema de múltiplos caixeiros viajantes	múltiplos	nós	não	uma	determinísticas
Problema de roteirização em nós com uma única base	múltiplos	nós	sim	uma	determinísticas
Problema de roteirização em nós com múltiplas bases	Múltiplos	nós	sim	múltiplas	determinísticas
Problema de roteirização em nós com demandas incertas	Múltiplos	nós	sim	uma	estocásticas
Problema de roteirização em arcos com limite de capacidade	Múltiplos	arcos	sim	uma	determinísticas

Fonte: Adaptado Bodin et al. (1983)

Ainda de acordo com Bodin et al. (1983), grande parte dos problemas combinados de roteirização e programação, ocorrem com restrições presentes, como janelas de tempo (horário de atendimento) ou também com restrições entre tarefas, como por exemplo: em uma operação de coleta e entrega, ambas devem estar contidas na mesma distribuição. Portanto, nesse tipo de problema são considerados os aspetos temporais como também os aspectos de deslocamento geográfico.

Sendo assim, uma das principais diferença entre as roteirizações puras e os combinados de roteirização e programação, são as formas em que as restrições estão inseridas nos problemas analisados, segundo Bodin et al. (1983). O Quadro 2, elucida as principais variáveis e restrições recorrentes em problemas combinados de roteirização e programação:

Quadro 2: Principais restrições e variáveis aplicadas a roteirização

Principais Variáveis
Uma ou múltiplas bases
Diferentes tipos de veículos
Coletas e entregas – coletas de retorno ("backhauls")
Janelas de tempo
Tempos de carga e descarga
Velocidades variáveis
Contratação de terceiros
Limite de peso e volume
Múltiplos compartimentos por veículo
Duração máxima do roteiro
Contabilização de horas extras
Horários de início e término de viagem
Roteiros com pernoite; troca de motoristas
Locais de parada fixos (e.g. almoço)
Restrições de tamanho de veículo e equipamentos para um cliente
Zonas de entregas e possibilidade de fracionamento de carga;
Barreiras físicas e restrições de circulação de veículos
Mais de um roteiro por veículo (quando veículo retorna cedo à base)

Fonte: Adaptado de Bodin et al. (1983).

2.1.2 Travel salesman problem (TSP)

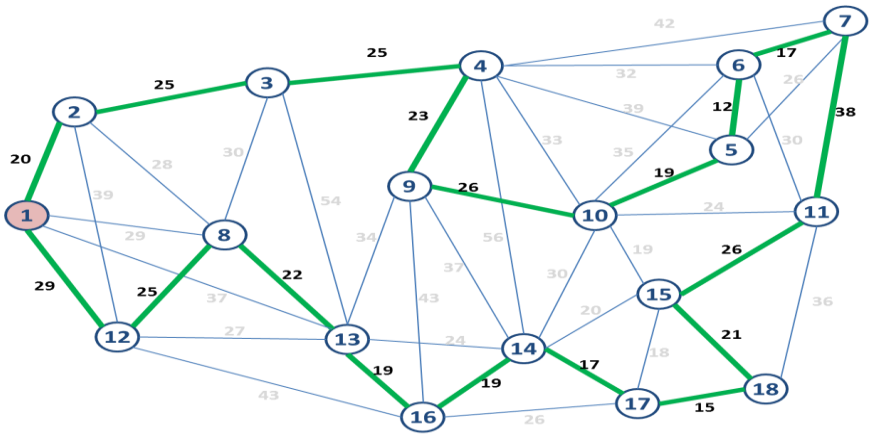
O método Travel salesman problem (TSP), também conhecido como problema do caixeiro viajante ou PCV, conforme Kretzschmar (2017) é definido por meio de um conjunto de n locais relacionados em uma matriz de distância entre cada ponto, tendo o seguinte propósito: o caixeiro viajante deve partir de uma cidade origem e visitar cada um dos $n - 1$ pontos de destino, apenas uma única vez e retornar ao ponto inicial, transcorrendo a menor trajetória possível.

Segundo Guedes (2009) o método de otimização do TSP que pode ser usado em diversas aplicações, desde o planejamento operacional logístico até o desenvolvimento de microchips por exemplo. O TSP está classificado no conjunto dos problemas denominados como NP-Hard, isso quer dizer que algoritmos com limitação polinomial, não são capazes de resolvê-lo, sendo assim

apenas instâncias com poucas variáveis podem ser resolvidas de maneira exata. Nesse caso, para problemas de maiores dimensões acabam sendo inviável a utilização de abordagens exatas. Porém, segundo o autor a utilização de métodos heurísticos aplicados a problemas NP-Hard, conseguem atingir boas soluções, próximas a ótima. De acordo com Gutin (2007) o TSP tem uma ampla diversificação de áreas onde pode ser aplicado, tais como: áreas industriais em geral, empresas de logística, fabricação de microchips, sequenciação de genomas, entre outros.

Na Figura 3 podemos verificar um exemplo de um percurso definido pelo método TSP em sua forma tradicional, adotando um grafo com n = 18 vértices, sendo 1 o vértice inicial. A solução é analítica e combinatória, a fim de encontrar o menor percurso que atenda todos os pontos.

Figura 3: Solução genérica do TSP



Fonte: Adaptado pelo autor (2022)

Dado um conjunto de pontos de distribuição, como na Figura 2 e uma matriz que relaciona as distâncias entre eles, o TSP consiste então, em definir um trajeto que parta do ponto de origem, passe por todos os outros pontos apenas uma única vez, retorne ao ponto de origem no final do trajeto e, por fim, forneça uma rota que defina o menor percurso possível.

O TSP é um método de otimização recorrentemente analisado na área da pesquisa operacional, segundo Silva (2010), o caixeiro viajante pode ser resolvido de diferentes formas, a tabela a seguir demonstra alguns dos principais métodos de resolução do TSP.

Quadro 3: Métodos de resolução do TSP

Metodologia	Descrição
Métodos exatos	Correspondem a resolução do problema de forma natural, porém inviável para problemas de grande porte, uma vez que é obtida pela combinação de todas as possibilidades.
Métodos heurísticos	Não garantem uma solução ótima, mas busca por soluções aproximadas e tempo de execução aceitáveis sem possuir limites formais de qualidade.
Algoritmos genéticos	Tratam de uma implementação segundo a teoria da seleção natural de Darwin, onde, indivíduos mais aptos sobrevivem. Utiliza de operadores de reprodução, cruzamento e mutação, gerando novas populações a partir destas operações, gradativamente aprimoradas.
Algoritmos meméticos	São considerados extensões de algoritmos genéticos, diferenciando-se apenas pela questão de evolução, que neste caso, simula uma evolução cultural e não biológica.
Algoritmos transgenéticos	Têm como base o processo de Endossimbiose/*Quorum Sensing*, onde a evolução não ocorre pela troca de informações dentro da população de cromossomos, mas entre a população de cromossomos e uma população de vetores, transgenéticos.
Otimização por nuvens de partículas	Metodologia baseada no comportamento do voo em bando dos pássaros, em que uma função mediria os esforços de cada pássaro em manter uma distância ótima de seu vizinho. Composto por uma função objetivo que avalia e direciona partículas no espaço de soluções conforme velocidade de cada uma.

Fonte: Adaptado de Silva (2010)

2.2 Métodos heurísticos e meta-heurísticos

A palavra Heurística, advém de uma expressão de origem grega chamada *heuriskein*, que significa descobrimento. Encontra partida, a expressão meta-heurística é a junção de outras duas palavras de origem grega, *heuriskein* mencionada anteriormente e a palavra *meta,* que significa depois, conferindo em seu sentido etimológico uma descoberta “superior”. (TALBI 2009).

Segundo Osman (1996) as metas-heurísticas podem ser vistas como um método de múltiplas iterações, que agrega diversos conceitos para explorar um espaço amostral de varredura de forma inteligente, evitando soluções de ótimos locais. Nos dias atuais, essa expressão é utilizada para designar um método de solução para problemas de otimização, baseados na busca local com o objetivo de encontrar a solução ótima global, através do processamento de múltiplas

iterações, porém não se encontra uma solução ótima, mas sim uma aproximação dela (FOULDS, 1984).

Ainda de acordo com Talbi (2009), as heurísticas e as meta-heurística podem ser classificadas como métodos que visam estratégias de diversos propósitos aplicados a uma vasta área da pesquisa operacional. Reeves (1993) descreve as técnicas envolvendo heurística como a busca por boas soluções, que não necessariamente será a solução ótima. Porém os valores encontrados serão aproximações apropriadas e o tempo de processamento computacional para a geração desses resultados será bastante reduzido, o que torna as heurísticas muito eficientes em problemas de maior complexidade. Além disso, segundo o mesmo autor, as heurísticas permitem a modelagem de problemas de caráter mais realístico, com um bom grau de acurácia, permitindo maior flexibilidade na modelagem de restrições em problemas práticos.

De acordo com Cunha (1997), a meta-heurística é um modelo heurístico incrementado para a resolução de forma genérica de problemas de otimização, principalmente para a otimização combinatória, envolvendo problemas NP-completos. Segundo o autor, algumas das principais meta-heurística construtivas são: o Algoritmo genético; Simulated annealing; GRASP; Busca tabu; Colônia de formigas (otimização); Colônia de abelhas (otimização); Lichtenberg Algorithm

Novaes (2007) afirma que as formas heurísticas para a simulação de rotas otimizadas partem do pressuposto da ligação de pontos adjacente, interligando sempre o ponto vizinho que esteja mais próximo, porém este não é o método mais eficiente para a otimização de rotas, principalmente para problemas mais complexos, em contrapartida, esse método tem uma velocidade de resposta rápida e pode ser utilizado como solução inicial para outros métodos complementares de otimização.

Sendo assim, podemos perceber a grande influência que os métodos heurísticos e meta-heurísticos exercem no contexto da pesquisa operacional. O quadro a seguir ilustra essa situação, demostrando como diversos problemas de otimização podem aplicar essa metodologia de abordagem para formular uma solução.

Quadro 4: Métodos de resolução heurísticos e meta-heurísticos

Referência Bibliográfica	Tipo de Problema	Restrições	Método de Solução	Função objetivo
Taillard (1999)	HFFVRP	capacidade dos veículos	**heurística** geração de colunas	min custos fixos + custos de viagem
Tarantilis et al. (2004)	HFFVRP	capacidade dos veículos	**metaheurística** BATA	min custos fixos + custos de viagem
Cunha (1997)	HFFVRPTW	capacidade dos veículos janelas de tempo duração máxima jornada	**heurística** relaxação lagrangiana	min custos fixos + distância total + tempo total rotas
Rochat e Semet (1994)	HFFVRPTW	capacidade dos veículos janelas de tempo duração máxima jornada tipo de veículo	**metaheurística** busca tabu	min distância total + tempo total rotas
Golden et al. (1984)	FSMVRP	capacidade dos veículos	**heurísticas construtivas** economias roteiro gigante **heurísticas de melhoria** troca de arcos	min custos fixos + distância total
Desrochers e Verhoog (1991)	FSMVRP	capacidade dos veículos duração máxima jornada	**heurística construtiva** economias	min custos fixos + distância total
Gouvêa (1992)	FSMVRP	capacidade dos veículos duração máxima jornada	**heurísticas construtivas** economias roteiro gigante **heurística de melhoria** troca de arcos	min custos fixos + custos de viagem
Salhi e Rand (1993)	FSMVRP	capacidade dos veículos duração máxima jornada	**heurística de melhoria** combinação de rotas eliminação de rotas inserção clientes em outra rota fracionamento de rotas troca de nós entre rotas	min custos fixos + distância
Teixeira e Cunha (2002)	FSMVRP	capacidade dos veículos	**heurísticas construtivas** economias	min custos fixos + custos de viagem
Gendreau et al. (1999)	FSMVRP	capacidade dos veículos	**metaheurística** busca tabu	min custos fixos + custos de viagem
Wassan e Osman (2002)	FSMVRP	capacidade dos veículos	**metaheurística** busca tabu	min custos fixos + custos de viagem
Yaman (2006)	FSMVRP	capacidade dos veículos	**heurística** geração de colunas	min custos fixos + custos de viagem
Liu e Shen (1999)	FSMVRPTW	capacidade dos veículos janelas de tempo	**heurística construtiva** inserção sequencial baseada em economias	min custos fixos + distância total
Dullaert et al. (2002)	FSMVRPTW	capacidade dos veículos janelas de tempo	**heurística construtiva** inserção sequencial	min custos fixos + distância total

Fonte: Adaptado de Prado (2006)

Desse modo, conforme destacado por Novaes (2007), as meta-heurísticas emergem como eficientes técnicas para a busca de conjuntos de soluções em problemas de análise combinatória, como é o caso do Problema do Caixeiro Viajante (PCV). Dentre os modelos de meta-heurística, merece destaque o uso de algoritmos genéticos, os quais serão abordados na seção subsequente.

2.3 O algoritmo genético

Conforme destacado por Lobo (2000), a estrutura do algoritmo genético constitui uma abordagem voltada para a resolução de problemas de maior complexidade, os quais não podem ser solucionados eficazmente por métodos exatos. Essa estrutura se destina a lidar com problemas que envolvem uma complexidade considerável, proporcionando uma alternativa eficaz diante das limitações dos métodos exatos. Esse tipo de abordagem é orientada através da teoria da seleção natural e dos princípios da genética. Através dos conceitos e características da evolução genética, Holland (1992) formulou a hipótese de uma estrutura de dados computacional que se baseia nos processos da evolução genética, com intuito de encontrar soluções para problemas de alto grau de complexidade.

Conforme Goldberg (1989), os algoritmos dessa classe utilizam uma estratégia estruturada de busca paralela, através de parâmetros de abordagem aleatórios que tendem a convergir, no qual são direcionados aos pontos de melhor aptidão, otimizando a função objetivo. Apesar da aleatoriedade, essa busca procede de maneira inteligente, buscando informações históricas para a encontrar novas soluções com melhores desempenho.

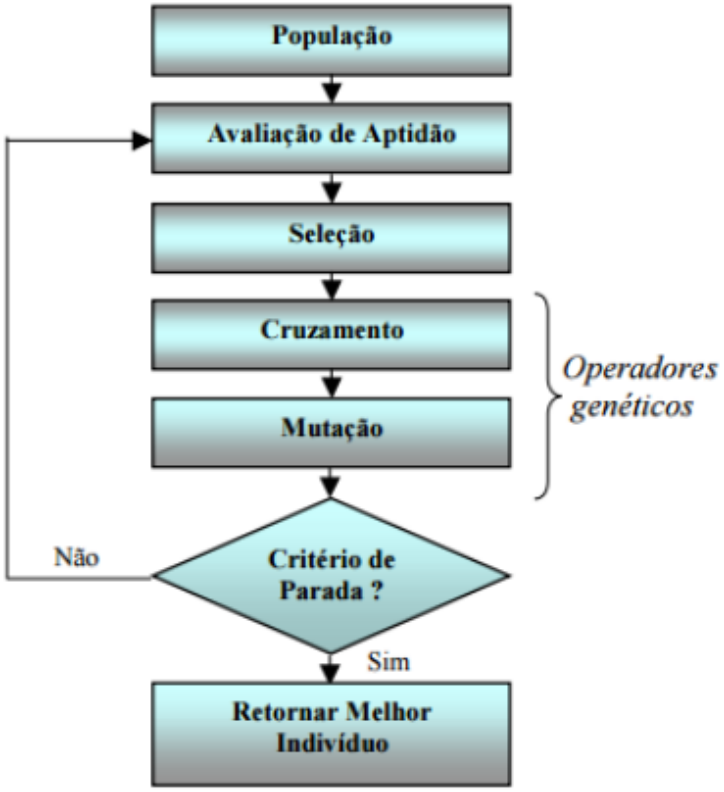
2.3.1 Características gerais do método

A estrutura básica do algoritmo genético, tem seu início por meio da criação da população inicial, com a criação dos cromossomos para apresentar a primeira geração. Em seguida, os indivíduos gerados são avaliados pela sua função de fitness, com isso são selecionados os indivíduos mais adaptados ao ambiente. Logo após, ocorre o *loop* das gerações, onde a população converge por meio de etapas de seleção, reprodução, mutação até serem selecionados os melhores indivíduos para as próximas gerações, e ser atingido o critério de parada. (SILVA, 2011).

Segundo Tanomaru (1995) os algoritmos genéticos pertencerem à classe de métodos probabilísticos de busca e otimização, porém não se trata de uma busca aleatória, o intuito é realizar buscas em regiões do espaço amostral com o

objetivo de se chegar em pontos ótimos globais, a estrutura básica de um algoritmo genético pode ser representada de acordo com a Figura 4 a seguir:

Figura 4: Estrutura básica do algoritmo genético



Fonte: Adaptado pelo autor (2022)

2.3.2 Representação do indivíduo

A representação do indivíduo tem por objetivo apresentar uma solução que engloba o espaço de busca que o problema propõe, alocando os valores para a montagem do cromossomo do indivíduo, sendo que essa representação demonstre todas as possíveis soluções do problema de modo que não interfira nas restrições impostas (SILVA, 2011). Uma das principais representações do indivíduo é a codificação de maneira binária , segundo Holland (1975), é uma forma simples de representação, que consiste na montagem de uma cadeia de valores que podem adotar números entre 0 ou 1. Mesmo sendo uma técnica muito difundida no âmbito acadêmico, existem diversas situações em que este

método acaba não sendo eficaz, principalmente em relação as restrições ou até mesmo pela velocidade de convergência nos procedimentos iniciais de buscas, o que acaba comprometendo a eficiência da sua aplicação. No caso de um problema de roteirização por meio da técnica do *TSP*, os números binários não são uma forma adequada para essa situação, pois geram cadeias muitos extensas de sequências, tornando o custo computacional inviável, além da geração de soluções infactíveis para o problema.

Neste caso, a representação do indivíduo pode ser mais precisa ao adotar o parâmetro de ordem , ou seja, o ponto que estiver na posição *n*, deverá ser a n-ésima na ordem das posições de cada ponto. De maneira geral, esse método de obragem é bastante empregado para representar um ordenamento (cromossomos) dentro de um modelo *TSP* (SILVA, 2011).

2.3.3 Seleção

Cada geração carrega consigo o seu próprio código genético, sendo que a probabilidade de sobrevivência de cada indivíduo depende exclusivamente da sua avaliação de aptidão no ambiente em que está inserido. Assim como no âmbito biológico, o método de seleção dos indivíduos é baseado na aptidão de cada um no ambiente, com intuito de sobrevivência e reprodução.

Segundo Mazzucco (1999), diversas experiências comprovam que a seleção natural é um fato incontestável. Alguns experimentos científicos realizado através da cultura de bactérias comprovam essa tese, onde as mesmas eram submetidas a doses de penicilina progressivas e, as mais resistentes proliferavam e se reproduziam, enquanto que as demais não sobreviviam, em contra partidas, passadas algumas gerações, as bactérias mais resistentes formavam decentes mais fortes contra a penicilina do que os antepassados. Os principais métodos utilizados na literatura para seleção da população são: O método do Cruzamento, Elitismo, Roleta e Torneio.

a) Método da Roleta

Desenvolvido por Holland (1975) na década de setenta, esse método tem por finalidade obter a probabilidade da seleção do indivíduo que realizará o

cruzamento, por meio do cálculo da função de fitness. Nesse modelo de seleção, a probabilidade de um indivíduo ser escolhido para reprodução é determinada pelo seu nível de aptidão em relação aos outros indivíduos. Isso é feito usando uma roleta de tamanho n, onde cada indivíduo tem uma chance proporcional ao seu nível de aptidão de ser escolhido para reprodução, e o valor da roleta é aleatoriamente selecionado a cada rodada.

b) Torneio

O procedimento conhecido como método do torneio consiste em selecionar um grupo de indivíduos aleatórios da população para competir entre si. Aquele que apresentar o melhor resultado em termos de aptidão (fitness) será o selecionado para sobreviver. Esse processo é repetido até que o número desejado de indivíduos seja alcançado para a reprodução. De acordo com Linden (2012), esse método não favorece significativamente nenhum indivíduo em particular, o que contribui para manter uma maior diversidade na população.

c) Elitismo

De acordo com Mitchell (1996), o elitismo é um modelo de seleção que se baseia na definição de um número determinado de indivíduos para reprodução, sendo selecionados aqueles com maior aptidão. Esse método tem como objetivo garantir que as conquistas mais significativas obtidas sejam preservadas nas próximas gerações.

2.3.4 Cruzamento

A reprodução de novos indivíduos é possível por meio do cruzamento genético, que combina partes de dois ou mais indivíduos e gera um novo cromossomo com características de ambos (YANG, 2014). Esse processo pode criar novos indivíduos mais aptos, mas a taxa de cruzamento é geralmente definida entre 70% e 100% da população, sem regras absolutas. Para evitar resultados inviáveis, é necessário um mapeamento das posições para respeitar as restrições do problema de roteirização de veículos. Cada zona eleitoral deve

ser visitada apenas uma vez, e portanto, não pode haver repetições no cromossomo dos filhos. Para lidar com essa restrição, uma sequência de zonas eleitorais é escolhida aleatoriamente do cromossomo do primeiro pai para formar um novo indivíduo, e as zonas eleitorais correspondentes ao outro pai são selecionadas na mesma ordem. Esse método preserva a ordem das cidades dos progenitores e envolve dois cortes aleatórios no início, onde cada parte é atribuída a um filho.

2.3.5 Mutação

Na aplicação de algoritmos genéticos, é importante considerar a ocorrência de mutação, ainda que seja um evento raro na natureza. Segundo Chambers (2000), o operador de mutação é fundamental para aumentar a diversidade genética na população, evitando que o algoritmo fique preso em mínimos locais. O operador de mutação introduz alterações aleatórias em um ou mais genes do cromossomo, de forma que a probabilidade de ocorrer essa mudança é determinada pela taxa de mutação estabelecida na aplicação.

De acordo com Goldberg (1989), o processo de mutação é um evento que modifica uma característica genética do cromossomo de forma aleatória, sendo essa mudança dependente de um fator de probabilidade geralmente pequeno. O objetivo da utilização desse operador é manter a variabilidade genética da população e evitar efeitos indesejáveis no algoritmo. Normalmente, o processo de mutação consiste na inversão ou substituição aleatória de algum gene na estrutura do cromossomo.

2.3.6 Geração da População

Após a aplicação dos operadores genéticos, é crucial determinar a próxima geração do algoritmo genético, o que implica na seleção dos indivíduos sobreviventes. Nesse contexto, é essencial estabelecer um operador que reintroduza a população no algoritmo, iniciando assim um novo ciclo. De acordo com Silva (2011), existem três métodos para essa reintrodução: o método puro, o método uniforme e o método do elitismo. No presente trabalho, optou-se pelo método uniforme.

O método uniforme, quando aplicado a um Algoritmo Genético para resolver um Problema do Caixeiro Viajante (PCV), implica em selecionar indivíduos da população de maneira equitativa, sem priorizar características específicas. Cada indivíduo tem uma probabilidade igual de ser escolhido para compor a próxima geração. Esse enfoque é particularmente útil no contexto do PCV, pois busca manter uma diversidade genética na população, evitando a convergência prematura para soluções subótimas.

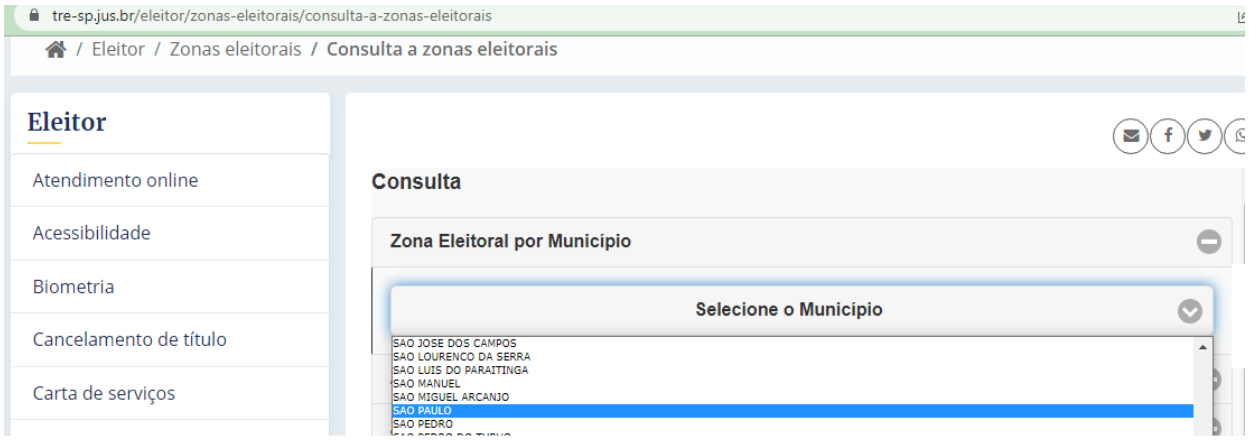
3. PROCEDIMENTOS METODOLÓGICOS

Neste capítulo, são expostas as características da pesquisa aplicada, juntamente com os parâmetros das ferramentas computacionais empregadas no estudo. Adicionalmente, são fornecidas informações sobre o processo de coleta de dados e sua implementação.

3.1 Recolhimento dos dados

Os dados utilizados nesta pesquisa foram extraídos do site do Tribunal Regional Eleitoral de São Paulo, restritos à base correspondente às zonas eleitorais do município em análise, totalizando 58 endereços distintos. A obtenção dessas informações ocorreu por meio da consulta à página virtual do TRE-SP, conforme ilustrado na Figura 5:

Figura 5: Recolhimento de dados no website do TRE-SP



Fonte: Adaptado de TRE-SP (2022)

Em seguida, após selecionar a opção desejada, nesse mesmo endereço eletrônico, foram geradas as relações de todas as zonas eleitorais existentes no município estudado. Na sequência, os dados específicos de cada uma das 58 zonas eleitorais foram processados no site do TRE-SP. A Figura 6 exemplifica o dado de uma das 58 zonas eleitorais registradas na plataforma:

Figura 6: Dados específicos da zonal eleitoral Bela Vista

1ª ZONA ELEITORAL
SÃO PAULO - BELA VISTA

	CARTÓRIO ELEITORAL
Endereço	Avenida Brigadeiro Luís Antônio, 453
Bairro	BELA VISTA
Cidade	SAO PAULO
CEP	01317-000
Localizada no Fórum	NÃO
E-mail	ze001@tre-sp.jus.br
Telefone	(0xx11) 31302701/31018683/
Fax	31058768
Whatsapp	11 31302701
Celular Institucional	
Horário de Atendimento	12:00 às 18:00
Chefe	CÍNTIA HIROMI NAKASAKO NAKASA
Juiz	

Fonte: Adaptado de TRE-SP (2022)

A partir dos dados extraídos do site do TRE-SP, fornecendo os endereços de todas as zonas eleitorais vinculadas nesse estudo, foi elaborada uma matriz de distância com o auxílio da ferramenta digital *Google Maps* relacionando todos os 58 postos coletados (mais o ponto de origem), totalizando assim um conjunto de 3481 pontos distintos utilizados neste estudo. A elaboração dessa matriz de distâncias será discutida nos capítulos subsequentes, além disso, através da mesma ferramenta digital (*Google Maps)*, foram extraídos os valores de latitude e longitude dos endereços de cada uma das zonas eleitorais analisadas, a tabela a seguir sintetiza todos os dados coletados e utilizados na pesquisa:

Tabela 1: Característica dos dados coletados

Dados	Abordagem	Característica	volume de dados
Zonas eleitorais	Qualitativo	Dados específicos referentes a todas as zonas eleitorais do município de SP, coletados na base de dados do TRE -SP	58
Latitude e Longitude dos endereços	Quantitativo	Longitude e latitude de todas a zonas eleitorais, gerada com o auxílio da ferramenta google maps	118
Matriz de distâncias	Quantitativo	Matriz que relaciona a distância entre todas a zonas eleitorais, gerada com o auxílio da ferramenta google maps	3481

Fonte: Autoria própria (2022)

3.2 Materiais e Ferramentas

Para a realização desse estudo foram utilizadas diferentes ferramentas computacionais, dentre elas podemos destacar a utilização da ferramenta *google maps®,* nessa ferramenta foram inseridos manualmente o endereço de todas as localizações da base de dados coletados, a fim de traçar as distâncias reais (quilômetros) entre todos os pontos analisados. Além disso, foi utilizada a ferramenta MS-Excel®, onde os dados coletados na ferramenta *google maps®,* foram relacionados em uma matriz de distâncias, e por intermédio dessa matriz foi desenvolvida uma solução computacional por intermédio do suplemento Solver do MS-Excel®, para gerar um sequenciamento de pontos com o objetivo de minimizar a distância total percorrida entre todas as localidades, em um trajeto que englobou todos os endereços das zonas eleitorais analisadas, com início e término na origem.

Outra ferramenta utilizada na pesquisa, foi a IDE (Integrated Development Environment) Pycharm, estruturada na linguagem de programação Python. Nessa ferramenta foi implementado um código baseado na meta-heurística do algoritmo evolucionário, que será apresentado nas seções subsequentes, onde foi processada a matriz de distâncias gerada com o auxílio da ferramenta *google maps®,* com o objetivo de traçar rotas que minimizam o percurso entre os pontos dentro da matriz supracitada.

Além disso, também por intermédio da ferramenta *google maps®,* que forneceu as coordenas (latitudes e longitude) de todos os pontos, foi possível utilizar o mesmo código para gerar um gráfico de dispersão entre os postos de distribuição atribuídos pelo algoritmo ao longo do trajeto, a fim de demostrar um panorama visual das rotas que foram aferidas no processo.

O dispositivo eletrônico utilizado no desenvolvimento dessa pesquisa foi um notebook da marca Samsung, modelo RV411 com processador Intel Inside Core i3 e 4gb de memória RAM.

3.2.1 Matriz de distâncias

Para a realização dessa pesquisa acadêmica, utilizou-se a ferramenta Google Maps para formar uma matriz de distâncias entre os endereços das zonas eleitorais referentes a este estudo, respeitando os seguintes critérios:

- Deve ser sempre considera a menor distância entre dois endereços distintos, caso exista mais de uma possibilidade de realizar o mesmo trajeto.

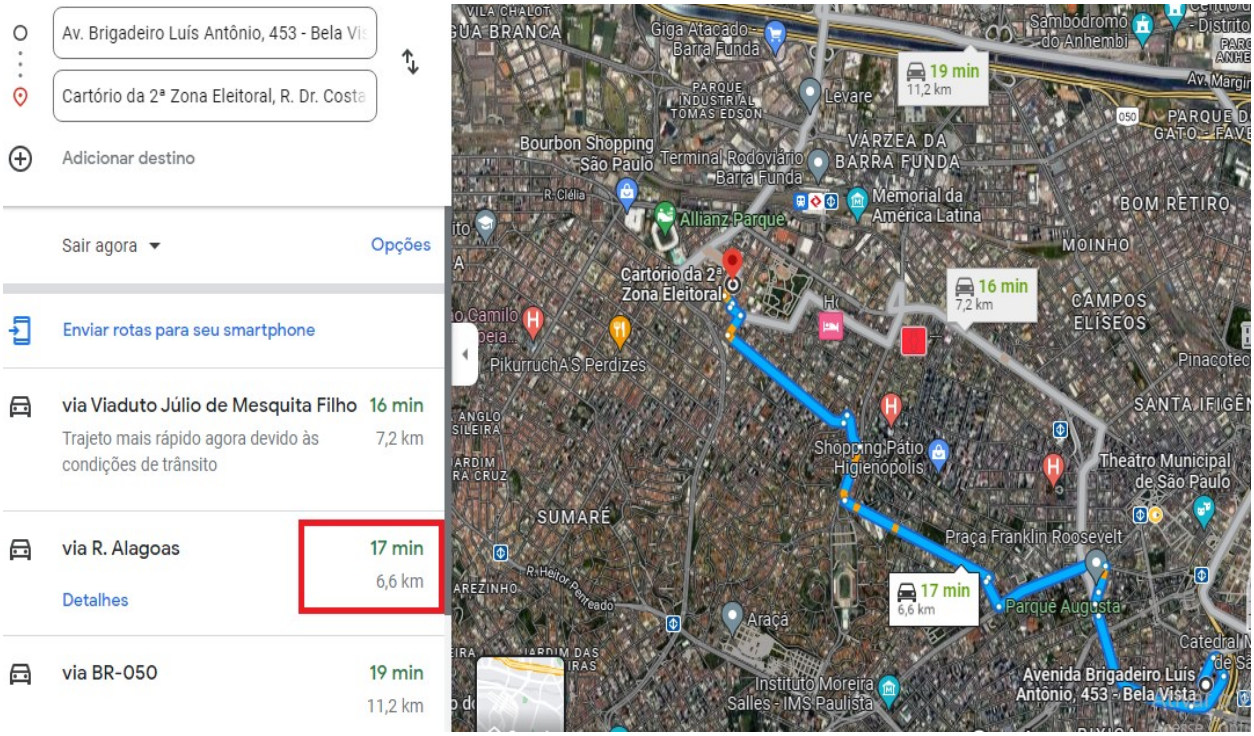
- Em todos os casos é utilizado a opção de rotas entre veículos, considerando a direção de sentido do trânsito em cada via.

- Deve-se considerar todas as possibilidades de distância existentes entre cada um dos endereços que será mapiado, exceto a distância entre o mesmo local, que será nula

Dessa forma, para cada ponto fixo da matriz (contendo o endereço da Z.E) foi traçado a distância entre todos os outros pontos adjacentes a este ponto, através da ferramenta Google Maps, até concluir a primeira linha, repetido o procedimento em todas as 59 linhas. Para cada linha preenchida *(i)* é gerada uma respetiva coluna *(j*), formando assim uma matriz quadrada *i x j*, onde *i x j* representa os endereços de cada um dos pontos dessa matriz.

A Figura 7 a seguir, ilustra o procedimento de montagem da matriz de distâncias. Nesse exemplo foi inserido o endereço da 1ª zona eleitoral SÃO PAULO – BELA VISTA como ponto fixo, e como ponto de destino foi inserido o endereço da 2ª zona eleitoral SÃO PAULO – PERDIZES

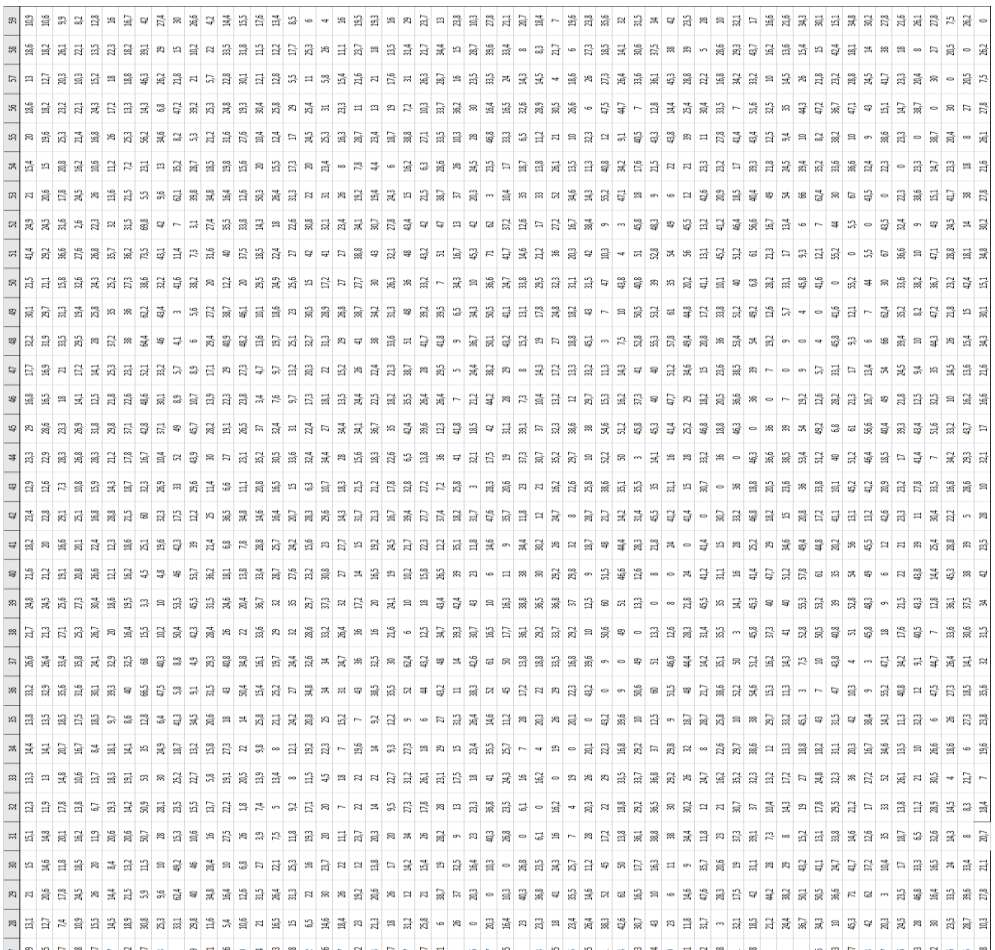
Figura 7: Rota de percurso definida pelo Google Maps



Fonte: Autoria própria (2022)

Nesse caso, a menor distância entre os dois endereços é 6,6 km, sendo assim esse valor é inserido dentro da matriz, o procedimento se repete para os demais pontos, até concluir todas as combinações. Totalizando 3841 pontos inseridos em uma matriz quadrada de 59x59, com origem no endereço do tribunal regional eleitoral de São Paulo, ilustrada na Figura 8 a seguir:

Figura 8: Matriz de distâncias

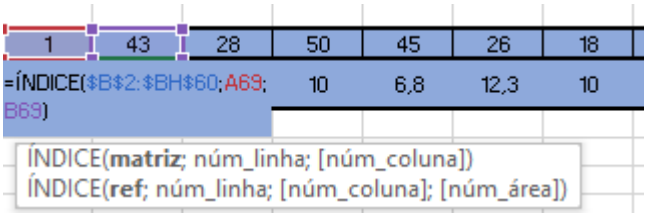


Fonte: Autoria própria (2022)

3.2.2 Software Microsoft Excel

Para formular a situação do problema elencado na pesquisa, através do MS-Excel®, primeiramente foi inserida a matriz de distâncias dentro da planilha do software, em seguida foi selecionada uma célula vazia para ser aplicada a função INDICE, essa função retorna um valor de referência em relação ao valor de dentro de uma tabela ou intervalo pré-selecionado, que no caso seria a matriz de distância, sendo assim, a função retornará o valor de referência contido no índice da matriz, como demostrado na Figura 9 abaixo:

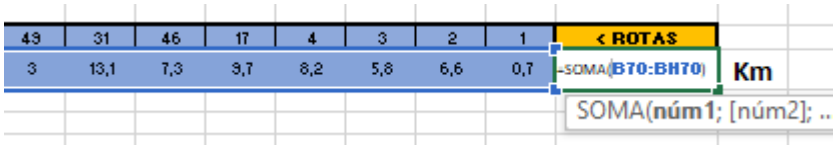
Figura 9: Aplicação da função ÍNDICE no MS-Excel



Fonte: Autoria própria (2022)

Em seguida foi aplicada a função SOMA, para somar todos valores retornados na função INDICE, conforme a Figura 10 abaixo:

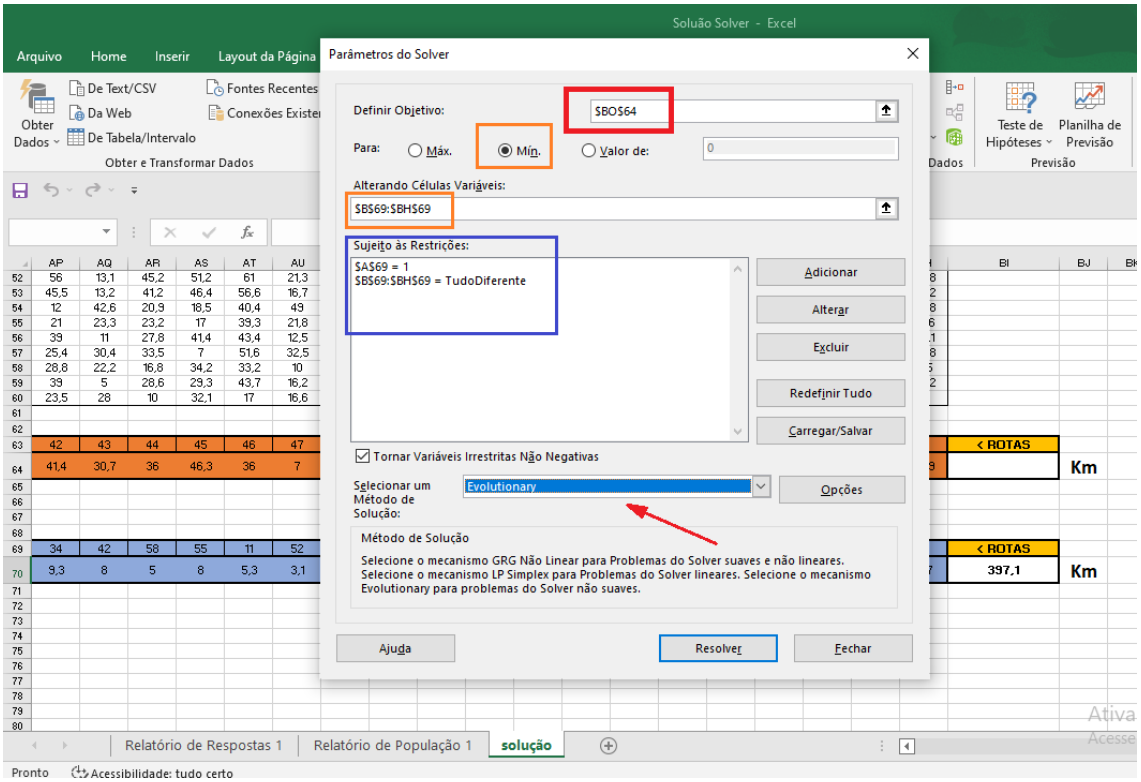
Figura 10: Soma dos índices



Fonte: Autoria própria (2022)

Na sequência foi aplicado o suplemento solver do MS-Excel. O objetivo foi minimizar a soma dos valores dos índices da matriz, alterando as posições de cada um dos índices, para que assim fosse definido o sequenciamento de rota que minimiza o percurso. Os parâmetros aplicados no solver são apresentados na Figura 11 abaixo:

Figura 11: Parâmetros aplicados pelo solver



Fonte: Autoria própria (2022)

Para garantir que todos os pontos da rota fossem visitados uma única vez, foi adicionada uma restrição para que todos os indicies das células variáveis fossem todos diferentes. Além disso outra restrição foi atribuída para garantir que o primeiro índice seja sempre igual a 1, isso assegura que a rota (solução) comece sempre pela origem da matriz. Após definidos os parâmetros, foi selecionado o método de resolução como sendo o Evolutionary. Pôr fim, foi executada a solução para o processamento dos resultados, que serão apresentados na seção subsequente.

3.2.3 Linguagem de Programação Python

Foi escolhido como método para resolução do problema, a aplicação de uma meta-heurística baseada em algoritmos genéticos por meio de programação em linguagem python, através da IDE (Integrated Development Environment) Pycharm. A IDE Pycharm é umas das principais interface dentro da linguagem python, possuindo diversas bibliotecas, que são um conjunto de módulos e

funções práticas e pré-estabelecidas, que reduzem o uso de códigos dentro da programação.

Para iniciar a modelagem do algoritmo proposto, em princípio foi necessário atribuir a formulação geral de um algoritmo genético, de acordo com a estrutura demonstrada no pseudocódigo a baixo:

```
Início Algoritmo Genético Gerações = 1
Gerações Max = ‘’Número de gerações a serem executadas’’
Gerar População Inicial
Enquanto Gerações <= Gerações Max faça
        Cálculo Do Fitness da População
        Seleção
        Cruzamento
        Mutação
        Geração da Nova População
Cálculo Do Fitness da Nova População
Gerações = Gerações + 1
Fim Enquanto
Imprimir melhor individuo
Fim Algoritmo Genético
```

Para iniciar o código dentro da IDE, primeiramente foi necessário importar dentro da biblioteca, algumas funções fundamentais para o funcionamento do algoritmo. Elas são a função *randint* para a geração de números aleatórios na seleção dos indivíduos, função *math* para a realização de atributos de operações matemáticas, e a função *pyplot*. *plt* para gerar a representação gráfica do conjunto de pontos vinculados as coordenadas de cada endereço da rota final, em seguida foi realizado o processamento de leitura das bases de dados, tanto das coordenas dos pontos, como da matriz de distâncias, que se encontram em arquivo no formato. csv, por fim é retornado as variáveis: matriz, coordenadas, conforme o script abaixo:

Figure 12: script I (processamento da base de dados)

```
from random import randint
import math
from matplotlib import pyplot as plt

def le_dados(arq_instancia, qtd_cidades):
    # Abre o arquivo para leitura
    with open(arq_instancia, 'r') as f:

        # Lê todas as linhas
        linhas = f.readlines()

        #Apaga a primeira linha do csv
        del linhas[0]

        # Lê a matriz ignorando o primeiro elemento
        matriz = list()
        lista_aux = list()
        for a in range(0, qtd_cidades+1):
            for i in range(1, qtd_cidades+2):
                lista_aux.append(float(linhas[a].strip().split(';')[i]))
            matriz.append(lista_aux[:])
            lista_aux.clear()
        linhas.clear()

       #Lê as coordenadas
   with open("coordenadas.csv", 'r') as f:
        linhas = f.readlines()
        del linhas[0]
        coordenadas = list()
        coordenada = list()
        for a in range(0, qtd_cidades + 1):
            for i in range(0, 2):
                coordenada.append(float(linhas[a].strip().split(';')[i]))
            coordenadas.append(coordenada[:])
            coordenada.clear()
        linhas.clear()

    # Retorna os valores lidos
    return matriz, coordenadas
```

Fonte: Autoria própria (2022).

Para iniciarmos a estrutura do algoritmo genético é necessário definirmos a nossa população. A população é o conjunto de indivíduos que estão sendo cogitados como solução e que serão usados para criar o novo conjunto de indivíduos para análise. O indivíduo, também chamado de cromossomo, é uma possível solução para um dado problema. Cada indivíduo é um conjunto de parâmetros, cuja representação depende do domínio. Sendo assim, a solução inicial do algoritmo é formulada da seguinte maneira:

Figure 13: script II (solução inicial)

```
def sol_inicial(populacao, qtd_cidades):

    lista = list()
    matriz_populacao = list()
    cidades_visitadas = 0

    for a in range(0,populacao):
        lista.append(0)
        cidades_visitadas += 1
        while (cidades_visitadas != qtd_cidades+1):
            num_sorteado = randint(1,qtd_cidades)
            if num_sorteado not in lista:
                lista.append(num_sorteado)
                cidades_visitadas += 1
        matriz_populacao.append(lista[:])
        lista.clear()
        cidades_visitadas = 0

    return matriz_populacao
```

Fonte: Autoria própria (2022)

Após a geração da população inicial, é necessário realizar a avaliação de aptidão. A função de Aptidão ou Fitness mede o grau de aptidão de cada indivíduo da população. O grau de aptidão sugere a qualidade da solução do indivíduo. A ideia nesse caso foi calcular todas as distancias entre as Z.E e organizar o vetor solução em ordem decrescente:

Figure 14: script III (função de Aptidão)

```
#calcula as distâncias, arruma o vetor e consequentemente
a matriz em ordem decrescente

def fitness(matriz_populacao, mat_distancias, qtd_cidades, populacao):

    lista_aux = list()
    vetor_dist = list()
    matriz_dist_populacao = list()

 #calcula todas as distâncias
 for a in range(0, populacao):
     fit = 0
     for i in range(0, qtd_cidades):
         lista_aux.append(mat_distancias[matriz_populacao[a][i]][matriz_populacao[a]
         fit += mat_distancias[matriz_populacao[a][i]][matriz_populacao[a][i+1]]
     lista_aux.append(mat_distancias[matriz_populacao[a][qtd_cidades]][matriz_popula
     fit += mat_distancias[matriz_populacao[a][qtd_cidades]][matriz_populacao[a][0]]
     matriz_dist_populacao.append(lista_aux[:])
     lista_aux.clear()
     vetor_dist.append(fit)
```

Fonte: Autoria própria (2022).

Dada a formulação da função de Fitness é necessário agora aplicar os operadores lógicos (Genéticos): Seleção, Cruzamento e Mutação. Para realizar a seleção é necessário considerar o genótipo do indivíduo, que é a sequência de genes. No caso de um problema de roteirização, cada gene é uma cidade (número de vértices no grafo). Exemplo: [0, 2, 1, 4, 3] é um genótipo de indivíduo para o problema de roteirização com 5 cidades, no caso da pesquisa realizada, as cidades representam os endereços de cada uma das Z.E. Essa seleção é feita de forma randômica, entre um número 0 a 10000, seguindo o critério do método da roleta.

Figure 15: script IV (Seleção)

```
#seleciona 1 num aleatorios entre 0 e 10000
 para ser o pail
num1 = randint(0, 10000)
indice_pail = 0
for aux in range(0, populacao):
    if num1 < vet_decisao[aux]:
        indice_pail = aux


# seleciona o pai2 sem ser o pail
indice_pai2 = populacao - 1
while True:

    num1 = randint(0, 10000)
    for num in range(0, populacao):
        if num1 < vet_decisao[num]:
            indice_pai2 = num
            break

    if indice_pail != indice_pai2:
        break
```

Fonte: Autoria própria (2022).

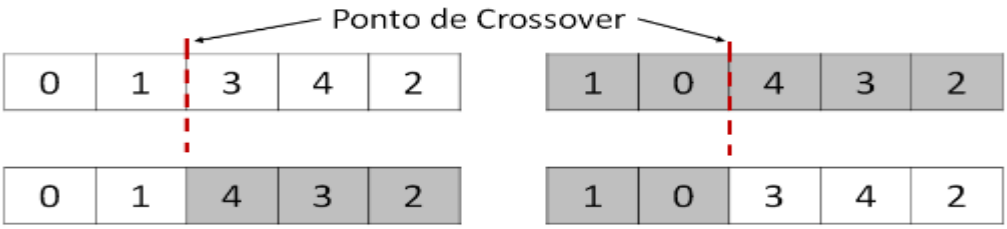
O cruzamento também conhecido por Crossover, realiza a recombinação de características dos pais, permitindo que as próximas gerações herdem essas características, a fim de melhorar a solução a cada nova recombinação, o script abaixo demonstra como é realizada a iteração do cruzamento.

Figure 16: script V (cruzamento)

```
def cruzamento(matriz_populacao, vetor_dist, matriz_dist_populacao, populacao, mat_distancias):

    #encontra o vetor de 1-Sdistribuição normal para selecionar dois pais.
    #qualquer vetor da população pode ser selecionado mas a probabilidade
    é maior quanto menor a distância

    vet_dist_inverso = list()
    #encontra o somatório da distribuição normal e o inverso das distâncias
    Sdistribuicao = 0
    for aux in range(0, populacao):
        vet_dist_inverso.append(1/vetor_dist[aux])
        Sdistribuicao += vet_dist_inverso[aux]

    #encontra os vetores de distribuição normal (dist/Sdists) e transforma no acumulado
    vet_decisao = list()
    for aux in range(0, populacao):
        if (aux == 0):
            vet_decisao.append((vet_dist_inverso[aux]/Sdistribuicao)*10000)

        else:
            vet_decisao.append(((vet_dist_inverso[aux] / Sdistribuicao) *10000)+ vet_decisao[aux-1])


    #seleciona 1 num aleatorios entre 0 e 10000 para ser o pai1

    num1 = randint(0, 10000)
    indice_pai1 = 0
    for aux in range(0, populacao):
        if num1 < vet_decisao[aux]:
            indice_pai1 = aux
```

Fonte: Autoria própria (2022).

Após o primeiro cruzamento, escolhe-se dois indivíduos e troca-se trechos dos cromossomos entre eles. Cortando o cromossomo e recombinando-o, como exemplificado no esquema a baixo.



Essa troca, permite que o algoritmo avalie novas possibilidades a fim de realocar cada um dos pares que gerem melhores soluções. A troca se repete, toda vez que é encontrada uma combinação que otimiza o resultado.

Figure 17: script VI (Geração do primeiro descendente)

```
# seleciona o pai2 sem ser o pai1
indice_pai2 = populacao - 1
while True:

    num1 = randint(0, 10000)
    for num in range(0, populacao):
        if num1 < vet_decisao[num]:
            indice_pai2 = num
            break

    if indice_pai1 != indice_pai2:
        break

#Geração de descendentes. Primeiro seleciona randomicamente onde vai corta
os vetores pai

vetor_pai1 = list()
vetor_pai2 = list()
vetor_pai1.append(matriz_populacao[indice_pai1][:])
vetor_pai2.append(matriz_populacao[indice_pai2][:])
vetor_filho1 = list()
vetor_filho2 = list()
corte = randint(1,qtd_cidades-1)

#Preenche os vetores filhos
vetor_filho1.extend(vetor_pai2[0][0:corte])
vetor_filho1.extend(vetor_pai1[0][corte:qtd_cidades+1])
vetor_filho2.extend(vetor_pai1[0][0:corte])
vetor_filho2.extend(vetor_pai2[0][corte:qtd_cidades+1])

#Arruma caso tenha alguma cidade repetida no vetor filho 1...
vetor_indices_repetidos = list()
vetor_substituicoes = list()
del vetor_filho1[vetor_filho1.index(0)
vetor_filho1.insert(0,0)

for aux in range(0, qtd_cidades+1):
    if aux not in vetor_filho1:
        vetor_substituicoes.append(aux)
```

Fonte: Autoria própria (2022).

Após a geração dos decentes através do cruzamento, é feito o cálculo total dos vetores filhos, para que assim seja feita uma comparação com o pior elemento da população, caso o vetor resulte uma solução inferior ao pior elemento da população, esse vetor é substituído entre os 50% da população menos apta. Essa substituição garante que os próximos vetores filhos, que serão gerados em novos cruzamentos, resultem em melhores descendentes para a população, sendo assim, a cada nova interação processada, os novos vetores

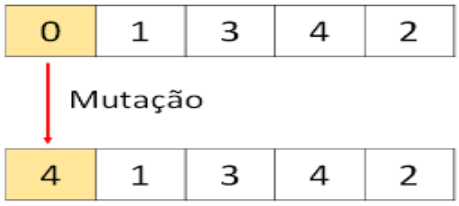
serão realocados para as poluções menos aptas, conferindo melhores soluções ao algoritmo, conforme o script abaixo:

Figure 18: script VII (Geração do segundo descendente)

```
#calcula a distância total dos vetores filhos e vê se é melhor
que o pior elemento da população.se for substitui entre o 50% da população menos apta.

dist_filho1 = 0
dist_filho2 = 0
vetor_dist_filho1 = list()
vetor_dist_filho2 = list()
for aux in range(0, qtd_cidades):
    vetor_dist_filho1.append(mat_distancias[vetor_filho1[aux]][vetor_filho1[aux+1]])
    vetor_dist_filho2.append(mat_distancias[vetor_filho2[aux]][vetor_filho2[aux + 1]])
    dist_filho1 += mat_distancias[vetor_filho1[aux]][vetor_filho1[aux+1]]
    dist_filho2 += mat_distancias[vetor_filho2[aux]][vetor_filho2[aux + 1]]

vetor_dist_filho1.append(mat_distancias[vetor_filho1[qtd_cidades]][vetor_filho1[0]])
vetor_dist_filho2.append(mat_distancias[vetor_filho2[qtd_cidades]][vetor_filho2[0]])
dist_filho1 += mat_distancias[vetor_filho1[qtd_cidades]][vetor_filho1[0]]
dist_filho2 += mat_distancias[vetor_filho2[qtd_cidades]][vetor_filho2[0]]


substituicao1 = -9999
substituicao2 = -9999
if (dist_filho1 < vetor_dist[populacao-1]) or (dist_filho2 < vetor_dist[populacao-1]):
    # recalcula a distribuição onde os maiores valores tem mais probabilidade de sair.
    Sdistribuicao = 0
    for aux in range(0, populacao):
        Sdistribuicao += vetor_dist[aux]

    # encontra os vetores de distribuição normal (dist/Sdists) e transforma no acumulado
    vet_decisao.clear()
    for aux in range(0, populacao):
        if (aux == 0):
            vet_decisao.append((vetor_dist[aux] / Sdistribuicao) * 10000)
```

Fonte: Autoria própria (2022).

Os operadores de mutação, são efetuados alterando-se o valor de um gene de um indivíduo sorteado aleatoriamente com uma determinada probabilidade. No caso do algoritmo desenvolvido essa probabilidade é fixada em uma taxa de mutação, que corresponde a 2%, o intuito da utilização desse operador é manter a diversidade genética e prevenir efeitos negativos no algoritmo



Para este vetor escolhido é feita uma repartição. É selecionado um valor inicial e depois o tamanho do trecho, para realizar a substituição. É necessário que a quantidade de elementos restantes %2, seja = 0, para a repartição do vetor posteriormente, conforme o script a seguir

Figure 19: script VIII (Mutação)

```
def mutacao(matriz_populacao, vetor_dist, matriz_dist_populacao, populacao, mat_distancias):
    # encontra o vetor de 1-Sdistribuição normal para selecionar um indivíduo.
    # qualquer vetor da população pode ser selecionado mas a probabilidade é maior quanto menor
      a distância percorrida.
    vet_dist_inverso = list()
    # encontra o somatório da distribuição normal e o inverso das distâncias
    Sdistribuicao = 0
    for aux in range(0, populacao):
        vet_dist_inverso.append(1 / vetor_dist[aux])
        Sdistribuicao += vet_dist_inverso[aux]

    # encontra os vetores de distribuição normal (dist/Sdists) e transforma no acumulado
    vet_decisao = list()
    for aux in range(0, populacao):
        if (aux == 0):
            vet_decisao.append((vet_dist_inverso[aux] / Sdistribuicao) * 10000)

        else:
            vet_decisao.append(((vet_dist_inverso[aux] / Sdistribuicao) * 10000) + vet_decisao[aux-1])

    # seleciona 1 num aleatorios entre 0 e 10000 para ser o indice

    num1 = randint(0, 10000)
    indice = 0
    for aux in range(0, populacao):
        if num1 < vet_decisao[aux]:
            indice = aux

    #Para este vetor escolhido é feita uma repartição. É selecionado um valor inicial e depois o tamanho do trecho

    while True: #é necessário q a quantidade de elementos restantes %2 seja = 0 para
                a repartição do vetor posteriormente

       ponto_inicial = randint(0, qtd_cidades)
       if (qtd_cidades+1-ponto_inicial)%2 == 0:
           ponto_meio = int(((qtd_cidades+1-ponto_inicial)/2) + ponto_inicial)
```

Fonte: Autoria própria (2022).

Ao término do processamento dos operadores genéticos, o algoritmo guardará as informações que satisfazem a solução do problema. Para cada iteração processada é realizada a montagem do vetor (população), realizasse então o cálculo da distância total desse vetor solução, depois é feita a comparação da distância antiga com a nova distância. Caso a nova distância otimize o resultado esse vetor é anexado no vetor original. Ao fim desse processo é retornada a variável *melhor população.*

Em seguida, é feita a geração da solução gráfica. Para a plotagem do gráfico foi utilizada a biblioteca pyplot, nesse caso para cada ponto x e y no plano

cartesiano foram indexados uma lista, após isso, foi realizado um laço de repetição entre a variável cidade em relação a matriz população. A cada iteração desse loop, a posição relativa de cada ponto na matriz solução foi anexada utilizando a função *append*. Os parâmetros utilizados no algoritmo foram: Tamanho da população: 175, número de interação: 30000, quantidade de cidades: 58

Figure 20: script IX (Melhor população)

```
    #monta o novo vetor

    novo_vetor = list()
    vetor_original_populacao = list()
    vetor_aux = list()
    vetor_original_populacao = (matriz_populacao[indice])


    novo_vetor = vetor_original_populacao[0:ponto_inicial]
    vetor_aux = vetor_original_populacao[ponto_inicial: ponto_meio]
    novo_vetor.extend(vetor_original_populacao[ponto_meio:qtd_cidades+1])
    novo_vetor.extend(vetor_aux)
    del novo_vetor[novo_vetor.index(0)]
    novo_vetor.insert(0, 0)

    #Calcula a nova distância

    dist_nova = 0
    vetor_dist_nova = list()
    for aux in range(0, qtd_cidades):
        vetor_dist_nova.append(mat_distancias[novo_vetor[aux]][novo_vetor[aux + 1]])
        dist_nova += mat_distancias[novo_vetor[aux]][novo_vetor[aux + 1]]

    vetor_dist_nova.append(mat_distancias[novo_vetor[qtd_cidades]][novo_vetor[0]])
    dist_nova += mat_distancias[novo_vetor[qtd_cidades]][novo_vetor[0]]

    #Compara a distância nova com a antiga e se for melhor substitui no vetor original

    if dist_nova < vetor_dist[indice]:
        matriz_dist_populacao.insert(indice, vetor_dist_nova)
        matriz_populacao.insert(indice, novo_vetor)
        vetor_dist.insert(indice, dist_nova)
        del matriz_populacao[indice+1]
        del matriz_dist_populacao[indice+1]
        del vetor_dist[indice+1]

def salva_solucao(arq_rel, melhor_populacao, dist_populacao, dist):
```

Fonte: Autoria própria (2022).

Por fim o algoritmo retorna o vetor com a melhor rota encontrada, logo após é realizada a plotagem do gráfico que define essa rota, conforme script a baixo:

Figure 21: script X (Representação gráfica)

```
# Plotagem do gráfico
xs = list()
ys = list()
for cidade in matriz_populacao[0]:
    xs.append(coordenadas[cidade][0])
    ys.append(coordenadas[cidade][1])

#Label gráfico
count = 0
ax.plot(xs, ys, 'ro--')

for x, y in zip(xs, ys):
    label = matriz_populacao[0][count]

    plt.annotate(label,  # this is the text
                 (x, y),  # these are the coordinates to position the label
                 textcoords="offset points",  # how to position the text
                 xytext=(0, 10),  # distance from text to points (x,y)
                 ha='center')  # horizontal alignment can be left, right or center
    count += 1

plt.show()
```

Fonte: Autoria própria (2022).

4. RESULTADOS

A seguir são apresentados os resultados obtidos através dos métodos aplicados de roteirização, inerentes a essa pesquisa.

Para a solução através do algoritmo genético, estruturado dentro da linguagem de programação em Python, foram atribuídos como resultados, o seguinte sequenciamento de rota:

Quadro 5: sequenciamento de rota (algoritmo genético)

ROTA								
0	1	3	11	58	17	2	13	40
DISTÂNCIA								
0.7	4.0	5.5	4.2	6.0	4.4	6.0	7.8	9.0
ROTA								
29	7	8	52	39	28	38	23	37
DISTÂNCIA								
11.5	8.0	9.6	6.0	6.0	10.0	10.0	6.0	3.0
ROTA								
43	55	34	24	20	21	53	6	5
DISTÂNCIA								
7.0	6.0	6.0	7.0	4.6	4.4	7.2	6.2	10.0
ROTA								
12	27	42	49	44	25	18	56	32
DISTÂNCIA								
5.4	3.0	10.1	6.8	12.3	17.5	5.8	4.0	8.0

ROTA								
30	41	54	10	50	35	26	9	48
DISTÂNCIA								
11.8	11.0	5.3	7.3	10.3	11.0	5.0	3.0	5.7
ROTA								
46	47	36	51	22	0			
DISTÂNCIA						TOTAL		
9.0	7.5	3.0	2.8	8.8		411.4		

Fonte: Autoria própria (2022)

Como parâmetro definido dentro da linguagem de programação em Phyton, a origem da rota teve início no ponto zero. No quadro a cima, além da definição do sequenciamento da rota estabelecida através da solução do algoritmo implementado, também foram atribuídas as distâncias percorridas entre cada um dos pontos de parada, além da distância total transitada para concluir o trajeto, totalizando um percurso final de 411,4 km.

Após a definição do sequenciamento de rota atribuído pelo algoritmo, foi possível traçar um roteiro de distribuição das urnas eletrônicas dentro do município de São Paulo, através do endereço de cada uma de suas respectivas zonas eleitorais, conforme o quadro abaixo:

Quadro 6: Roteiro de distribuição (solução algoritmo genético)

ROTA	ZONAS ELEITORAIS	ENDEREÇO
0	TRE-SP	Rua Francisca Miquelina, 123
1	SÃO PAULO - BELA VISTA	Avenida Brigadeiro Luís Antônio, 453
3	SÃO PAULO - SANTA IFIGÊNIA	Rua ANTONIO CORUJA, 99/109
11	SÃO PAULO - SANTANA	Avenida Leôncio de Magalhães, 357 363 - andar 2

58	SÃO PAULO - LAUZANE PAULISTA	Avenida Santa Inês, 864
17	SÃO PAULO - CASA VERDE	Avenida Casa Verde, 1819
2	SÃO PAULO - PERDIZES	Rua Doutor Costa Junior, 509
13	SÃO PAULO - PINHEIROS	Rua Ferreira de Araújo, 538
40	SÃO PAULO - RIO PEQUENO	Avenida Corifeu de Azevedo Marques, 1.140
29	SÃO PAULO - MORUMBI	Rua Ibiapaba, 422
7	SÃO PAULO - VALO VELHO	Avenida Ellis Maas, 875 877
8	SÃO PAULO - SANTO AMARO	Rua Tenente Cel. Carlos da Silva Araújo, 355 -
52	SÃO PAULO - JARDIM SÃO LUÍS	Rua AMÉRICO FALCÃO, 251
39	SÃO PAULO - CAPÃO REDONDO	Estrada de Itapecerica, 2.720
28	SÃO PAULO - CAMPO LIMPO	Rua Américo Falcão, 251 257
38	SÃO PAULO - PIRAPORINHA	Rua Professor Barroso do Amaral, 32
23	SÃO PAULO - CAPELA DO SOCORRO	Avenida Atlântica, 1551
37	SÃO PAULO - GRAJAÚ	Rua Antônio Carlos Tacconi, 45
43	SÃO PAULO - PARELHEIROS	Avenida Pedro Roschel Gottzfritz, 210
55	SÃO PAULO - JAÇANÃ	Avenida Paulo Lincoln do Valle Pontin, 94
34	SÃO PAULO - BRASILÂNDIA	Rua Bonifácio Cubas, 567
24	SÃO PAULO - VILA JACUÍ	Rua Coronel Manuel Feliciano de Souza, 134 A
20	SÃO PAULO - PEDREIRA	Avenida Nossa Senhora do Sabará, 4.051
21	SÃO PAULO - CIDADE ADEMAR	Avenida Cupecê, 1147
53	SÃO PAULO - JABAQUARA	Avenida Engenheiro Armando de Arruda Pereira, 2.917

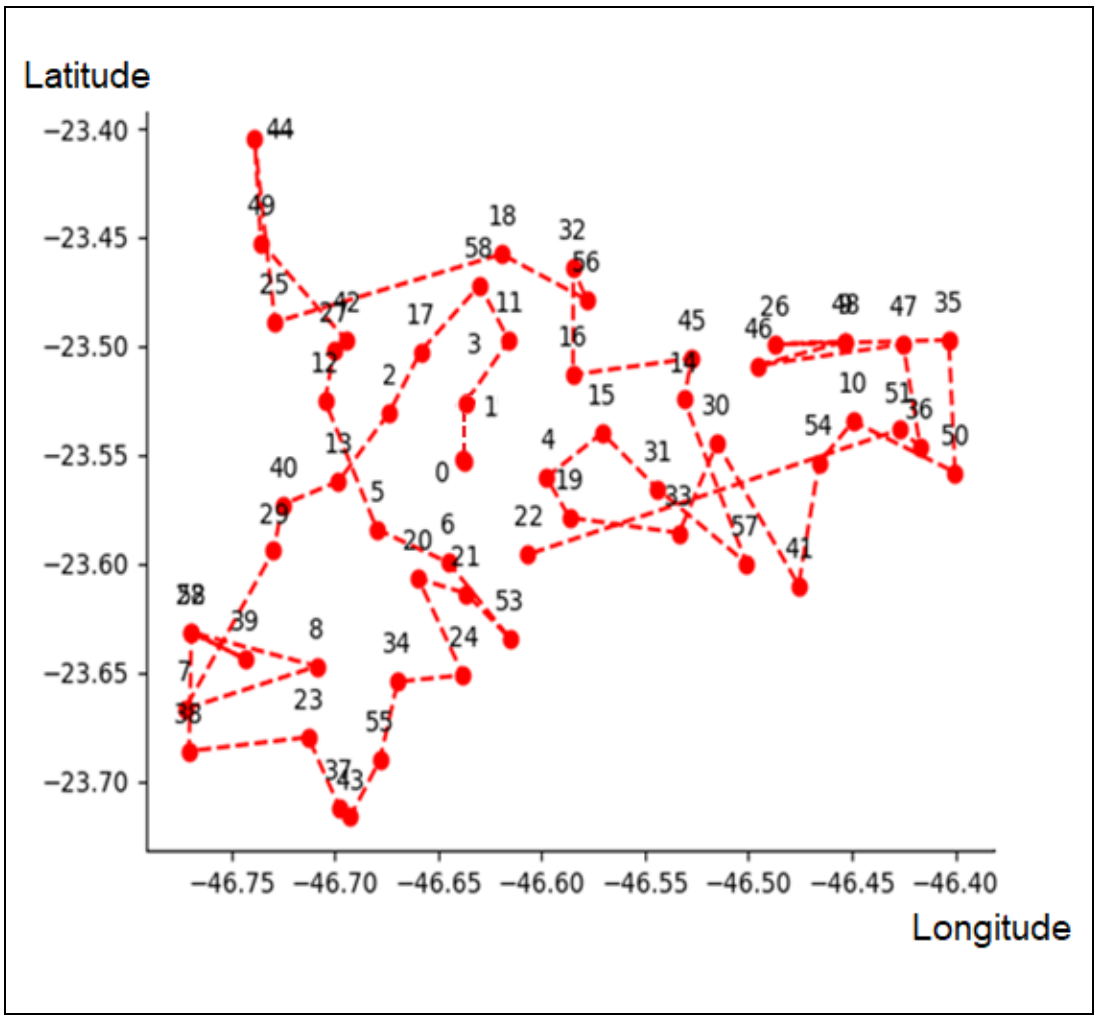
6	SÃO PAULO - INDIANÓPOLIS	Rua Clodomiro Amazonas, 41
5	SÃO PAULO - SAÚDE	Rua Itapiru, 174 -
12	SÃO PAULO - CURSINO	Avenida DO CURSINO, 3821
27	SÃO PAULO - VILA MARIANA	Rua Loefgreen, 2007
42	SÃO PAULO - JARDIM PAULISTA	Rua Clodomiro Amazonas, 41
44	SÃO PAULO - NOSSA SENHORA DO Ó	Avenida Paula Ferreira, 60
25	SÃO PAULO – BRASILÂNDIA	Rua Bonifácio Cubas, 567
18	SÃO PAULO – JARAGUÁ	Estrada de Taipas, 412
56	SÃO PAULO – PERUS	Rua DEMIFONTE, 208
32	SÃO PAULO – PIRITUBA	Avenida Raimundo Pereira de
16	SÃO PAULO - TUCURUVI	Rua Maria Amália Lopes de Azevedo, 657
45	SÃO PAULO - VILA SABRINA	Avenida Roland Garros, 1013
14	SÃO PAULO – JAÇANÃ	Avenida Paulo Lincoln do Valle
57	SÃO PAULO - VILA MARIA	Rua ARARITAGUABA, 936 938
31	SÃO PAULO – CANGAÍBA	Avenida Cangaíba, 1.158
15	SÃO PAULO - PENHA DE FRANÇA	Rua Jorge Augusto, 258
4	SÃO PAULO - TEOTÔNIO VILELA	Avenida Arquiteto Vilanova Artigas, 1815
19	SÃO PAULO - VILA FORMOSA	Rua Cristovão Girão, 132
33	SÃO PAULO - TATUAPÉ	Praça Santa Terezinha, 43/47

30	SÃO PAULO - MOÓCA	Rua Madre de Deus, 427
41	SÃO PAULO - VILA PRUDENTE	Avenida Paes de Barros , 3237
54	SÃO PAULO – SAPOPEMBA	Avenida Sapopemba, 6214
10	SÃO PAULO - VILA MATILDE	Rua Fernão Albernaz, 400
50	SÃO PAULO - SÃO MATEUS	Rua Elísio Ferreira, 506
35	SÃO PAULO - PARQUE DO CARMO	Avenida LIDER, 3114
26	SÃO PAULO – ITAQUERA	Rua Paulo Lopes Leão, 166
9	SÃO PAULO - CIDADE TIRADENTES	Rua Àlvaro da Costa, 28
48	SÃO PAULO - ITAIM PAULISTA	Rua Monte Camberela, 342
33	SÃO PAULO - ERMELINO MATARAZZO	Avenida Boturussu, 874
16	SÃO PAULO - SÃO MIGUEL PAULISTA	Rua Coronel Manuel Feliciano de Souza, 134 A
46	SÃO PAULO - VILA JACUÍ	Rua Coronel Manuel Feliciano de Souza, 134 A
47	SÃO PAULO - PONTE RASA	Avenida São Miguel, 3906
36	SÃO PAULO - JARDIM HELENA	Avenida Cocá, 633
51	SÃO PAULO – GUAIANASES	Rua Serra do Mar, 180
22	SÃO PAULO - CONJUNTO JOSÉ BONIFÁCIO	Estrada Itaquera-Guaianases, 2120
0	TRE-SP	Rua Francisca Miquelina, 123

Fonte: Autoria própria (2022).

Além disso, o algoritmo processou a plotagem de um gráfico de dispersão para cada ponto de parada atribuído à rota final, relacionando-os às suas respectivas coordenadas geográficas (latitude e longitude). Esse gráfico proporciona uma representação visual do trajeto gerado pelo algoritmo, conforme ilustrado na Figura 22.:

Figura 22: Rota definida em coordenas geográficas.

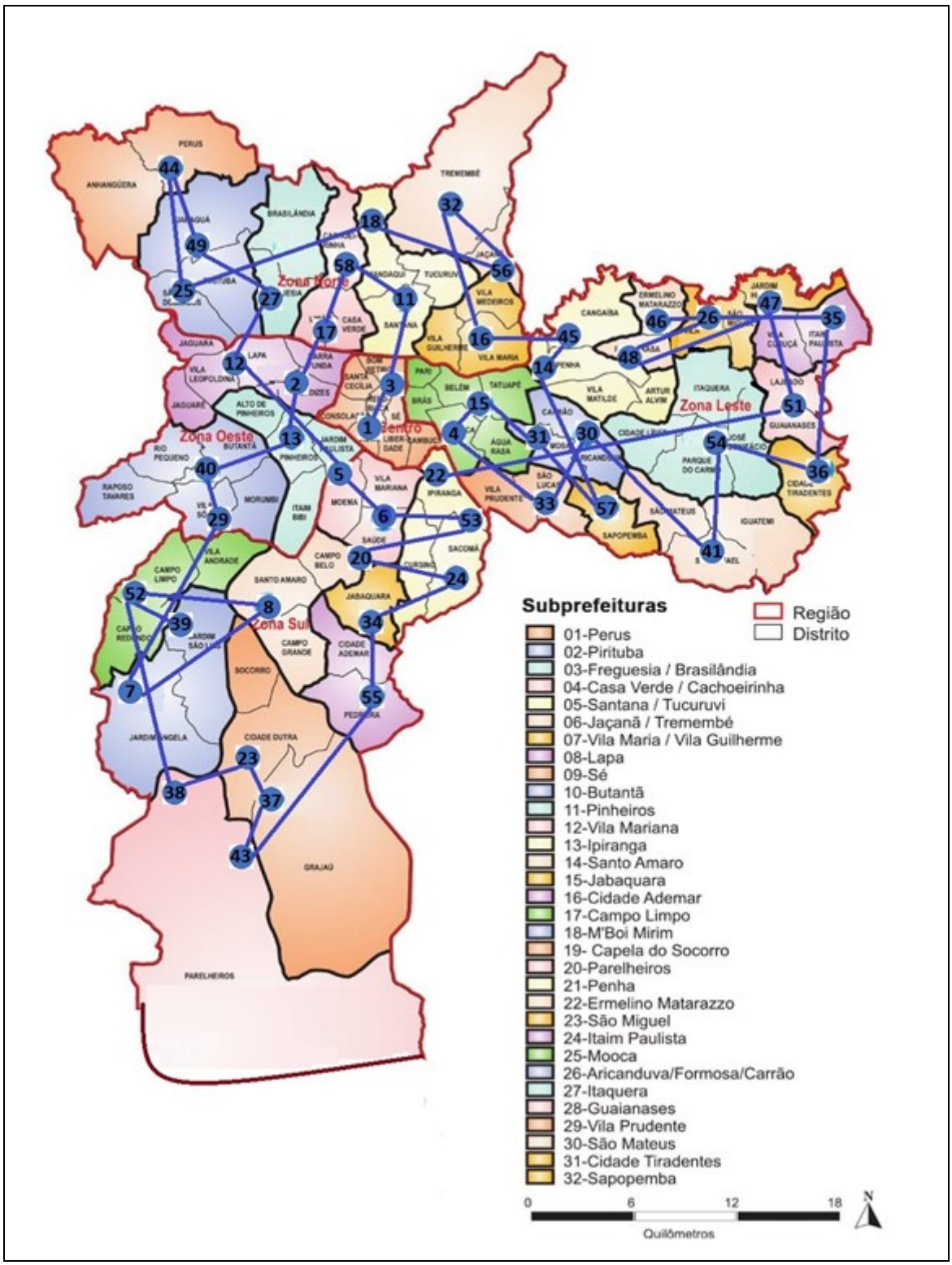


Fonte: Autoria própria (2022).

Com base no roteiro delineado no Quadro 6, que associa a distribuição das urnas eletrônicas às suas respectivas zonas eleitorais, foi elaborada a demarcação física da rota de distribuição final, incluindo todos os pontos de parada. Essa demarcação teve como referência o mapa de desenvolvimento

urbano do município de São Paulo, disponibilizado no website da Secretaria Municipal de Desenvolvimento Urbano, conforme a ilustração abaixo:

Figura 23: Demarcação da rota final de distribuição



Fonte: Adaptado pelo autor (2022).

Para a solução através do software *MS-Excel*, utilizando o complemento *solver,* foram atribuídos como resultados, o seguinte sequenciamento de rota:

Quadro 7: Sequenciamento de rota (MS-Excel)

ROTA								
1	43	28	50	45	26	18	12	5
DISTÂNCIA								
12,9	3	10	6,8	12,3	10	5,3	8,7	3,6
ROTA								
20	59	19	33	57	27	47	15	16
DISTÂNCIA								
16	4	4,5	4	16	5	4,7	6,9	5
ROTA								
32	14	25	35	9	56	38	44	24
DISTÂNCIA								
1,8	14,8	6	6,4	6,8	7	3	6,5	10
ROTA								
39	8	53	29	40	30	41	13	6
DISTÂNCIA								
3,3	5,5	3	6	11	9	6,8	10	7
ROTA								
21	7	22	54	23	34	42	58	55
DISTÂNCIA								
3,1	2,8	4,4	6	9,3	8	5	8	5,3
ROTA								
11	52	37	51	48	36	10	49	31
DISTÂNCIA								
3,1	3	4	9,3	3	5,8	3	13,1	7,3
ROTA								
46	17	4	3	2	1			

DISTÂNCIA						TOTAL		
9,7	8,2	5,8	6,6	0,7		379,1		

Fonte: Autoria própria (2022).

Como parâmetro definido dentro do software MS-Excel, a origem da rota teve seu início determinado no ponto 1. No quadro a cima, assim como no sequenciamento de rota definido pelo algoritmo genético implementado e discutido anteriormente, também foram atribuídas as distâncias percorridas entre cada um dos pontos de parada, além da distância total realizada para concluir o trajeto, que foi de 397,1 km.

Além disso, também foi elaborado o roteiro de distribuição das urnas eletrônicas em suas respectivas zonas eleitorais, tendo como base o sequenciamento de rota gerado pela aplicação do método de roteirização atribuído pela solução do software MS-Excel, conforme do quadro abaixo:

Quadro 8: Roteiro de distribuição (solução software MS-Excel)

ID	ROTEIRO/ ZONAS ELEITORAIS	ENDEREÇO
-	TRE-SP	Rua Francisca Miquelina, 123
376	SÃO PAULO – BRASILÂNDIA	Rua Bonifácio Cubas, 567
327	SÃO PAULO - NOSSA SENHORA DO Ó	Avenida Paula Ferreira, 60
403	SÃO PAULO – JARAGUÁ	Estrada de Taipas, 412
389	SÃO PAULO – PERUS	Rua DEMIFONTE, 208
325	SÃO PAULO – PIRITUBA	Avenida Raimundo Pereira de Magalhães, 4586
255	SÃO PAULO - CASA VERDE	Avenida Casa Verde, 1819
249	SÃO PAULO – SANTANA	Avenida Leôncio de Magalhães, 357 363 - andar superior
4	SÃO PAULO – MOÓCA	Rua Madre de Deus, 427

257	SÃO PAULO - VILA PRUDENTE	Avenida Paes de Barros , 3237
422	SÃO PAULO - LAUZANE PAULISTA	Avenida Santa Inês, 864
256	SÃO PAULO – TUCURUVI	Rua Maria Amália Lopes de Azevedo, 657
349	SÃO PAULO – JAÇANÃ	Avenida Paulo Lincoln do Valle Pontin, 94
420	SÃO PAULO - VILA SABRINA	Avenida Roland Garros, 1013
326	SÃO PAULO - ERMELINO MATARAZZO	Avenida Boturussu, 874
392	SÃO PAULO - PONTE RASA	Avenida São Miguel, 3906
252	SÃO PAULO - PENHA DE FRANÇA	Rua Jorge Augusto, 258
253	SÃO PAULO – TATUAPÉ	Praça Santa Terezinha, 43/47
348	SÃO PAULO - VILA FORMOSA	Rua Cristovão Girão, 132
251	SÃO PAULO – PINHEIROS	Rua Ferreira de Araújo, 538
320	SÃO PAULO – JABAQUARA	Avenida Engenheiro Armando de Arruda Pereira, 2.917
351	SÃO PAULO - CIDADE ADEMAR	Avenida Cupecê, 1147
246	SÃO PAULO - SANTO AMARO	Rua Tenente Cel. Carlos da Silva Araújo, 355 -
418	SÃO PAULO – PEDREIRA	Avenida Nossa Senhora do Sabará, 4.051
371	SÃO PAULO – GRAJAÚ	Rua Antônio Carlos Tacconi, 45
381	SÃO PAULO – PARELHEIROS	Avenida Pedro Roschel Gottzfritz, 210
280	SÃO PAULO - CAPELA DO SOCORRO	Avenida Atlântica, 1551
372	SÃO PAULO – PIRAPORINHA	Rua Professor Barroso do Amaral, 32
20	SÃO PAULO - VALO VELHO	Avenida Ellis Maas, 875 877
408	SÃO PAULO - JARDIM SÃO LUÍS	Rua AMÉRICO FALCÃO, 251

328	SÃO PAULO - CAMPO LIMPO	Rua Américo Falcão, 251 257
373	SÃO PAULO - CAPÃO REDONDO	Estrada de Itapecerica, 2.720
346	SÃO PAULO – MORUMBI	Rua Ibiapaba, 422
374	SÃO PAULO - RIO PEQUENO	Avenida Corifeu de Azevedo Marques, 1.140
250	SÃO PAULO – LAPA	Rua Coriolano, 1978
5	SÃO PAULO - JARDIM PAULISTA	Rua Clodomiro Amazonas, 41
258	SÃO PAULO – INDIANÓPOLIS	Rua Clodomiro Amazonas, 41
6	SÃO PAULO - VILA MARIANA	Rua Loefgreen, 2007
259	SÃO PAULO – SAÚDE	Rua Itapiru, 174 -
413	SÃO PAULO – CURSINO	Avenida DO CURSINO, 3821
260	SÃO PAULO – IPIRANGA	Rua Bom Pastor, 2204
350	SÃO PAULO – SAPOPEMBA	Avenida Sapopemba, 6214
375	SÃO PAULO - SÃO MATEUS	Rua Elísio Ferreira, 506
421	SÃO PAULO - TEOTÔNIO VILELA	Avenida Arquiteto Vilanova Artigas, 1815
417	SÃO PAULO - PARQUE DO CARMO	Avenida LIDER, 3114
248	SÃO PAULO – ITAQUERA	Rua Paulo Lopes Leão, 166
405	SÃO PAULO - CONJUNTO JOSÉ BONIFÁCIO	Estrada Itaquera-Guaianases, 2120
353	SÃO PAULO – GUAIANASES	Rua Serra do Mar, 180
404	SÃO PAULO - CIDADE TIRADENTES	Rua Àlvarez da Costa, 28
397	SÃO PAULO - JARDIM HELENA	Avenida Cocá, 633
352	SÃO PAULO - ITAIM PAULISTA	Rua Monte Camberela, 342
247	SÃO PAULO - SÃO MIGUEL PAULISTA	Rua Coronel Manuel Feliciano de Souza, 134 A
398	SÃO PAULO - VILA JACUÍ	Rua Coronel Manuel Feliciano de Souza, 134 A
347	SÃO PAULO - VILA MATILDE	Rua Fernão Albernaz, 400

390	SÃO PAULO – CANGAÍBA	Avenida Cangaíba, 1.158
254	SÃO PAULO - VILA MARIA	Rua ARARITAGUABA, 936 938
18	SÃO PAULO – JARAGUÁ	Estrada de Taipas, 412
48	SÃO PAULO - ITAIM PAULISTA	Rua Monte Camberela, 342
3	SÃO PAULO - SANTA IFIGÊNIA	Rua ANTONIO CORUJA, 99/109
2	SÃO PAULO - PERDIZES	Rua Doutor Costa Junior, 509
1	SÃO PAULO - BELA VISTA	Avenida Brigadeiro Luís Antônio, 453
0	TRE-SP	Rua Francisca Miquelina, 123

Fonte: Autoria própria (2022).

Sendo assim, no cenário apresentado podemos observar que ambos os métodos aplicados, tanto o algoritmo genético implementado, quanto a resolução através do software *MS-Excel* com o complemento *solver*, obtiveram resultados finais bastante compatíveis entre si. Além do tempo total de processamento para gerar a solução final, que também foram bem próximos. A tabela abaixo sintetiza a comparação dos resultados entre os dois métodos de roteirização aplicados na pesquisa.

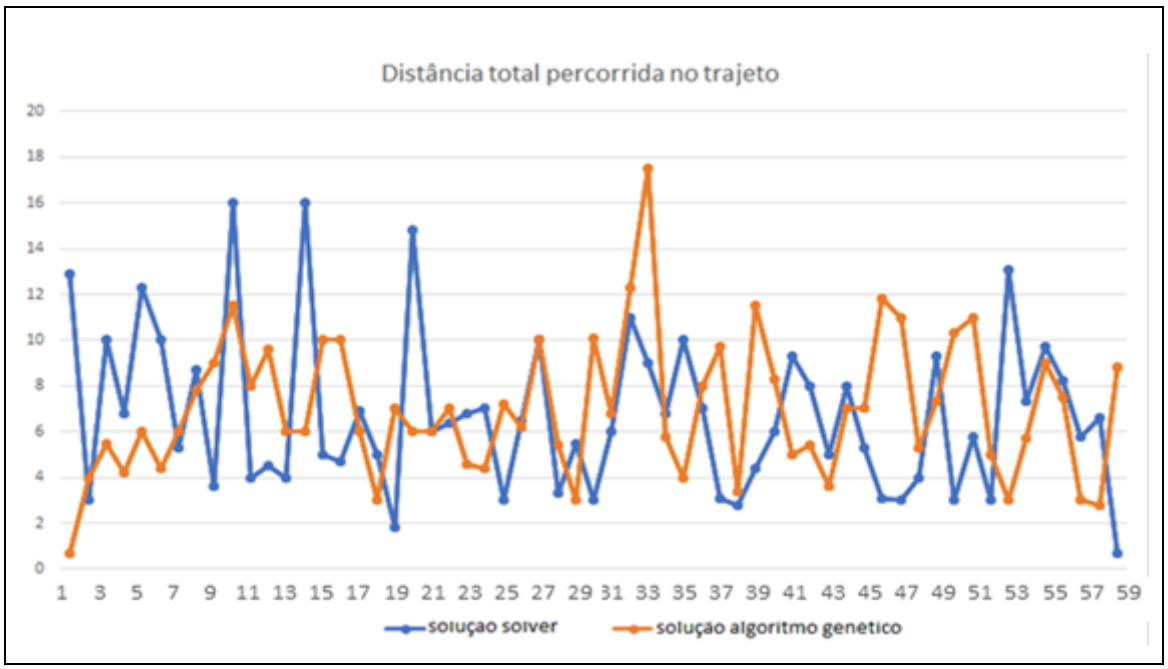
Tabela 5: Comparativo entre os métodos de roteirização aplicados

	Algoritmo Genético	MS-Excel/Solver	GAP (%)
DISTÂNCIA TOTAL PERCORRIDA (Km)	411,4	379,1	8,5
TEMPO DE PROCESSAMENTO (Min)	12 min 23s	10 min 38s	14,13

Fonte: Autoria própria (2022).

Também foi executado através da plotagem de um gráfico, um teste comparativo entre os dois métodos de roteirização aplicados, indicando a distância em quilometragem percorrida entre todos os pontos de paradas até concluir o trajeto integralmente, exercidos por ambos os métodos de resolução, conforme a Figura 24 abaixo:

Figura 24: Comparação entre as soluções algoritmo genético e MS-Excel (Solver)



Fonte: Autoria própria (2022).

Para testar a eficácia do algoritmo proposto, foi utilizada a base de dados do TSPLIB, uma coleção de instâncias do TSP com diferentes características, sendo uma das principais bases acadêmicas de referência para problemas NP-hard. Neste caso, foi empregada a instância hk48 como parâmetro de teste comparativo para o algoritmo.

A instancia hk48 possui 48 pontos distintos (cidades), sendo que a sua solução ótima catalogada no TSPLIB é de 11461 metros. O algoritmo desenvolvido nessa pesquisa, foi processado para resolver essa mesma instancia (hk48) e obteve uma solução de 13387 metros, com um GAP de 16,8%, a tabela abaixo sintetiza os resultados:

Tabela 6: Desempenho do Algoritmo Genético proposto

DISTÂNCIA TOTAL	Algoritmo Genético proposto	Solução TSPLIB instancia hk48	GAP (%)
	13387	11461	16,8

Fonte: Autoria própria (2022).

5. CONSIDERAÇÕES FINAIS

Com a crescente utilização da tecnologia nas eleições, a logística de distribuição de urnas eletrônicas se tornou um processo fundamental para o sucesso do processo eleitoral. No entanto, a roteirização dos veículos que fazem a distribuição das urnas ainda é um desafio para muitos municípios, principalmente nas grandes capitais, como no caso da cidade de São Paulo.

O objetivo deste trabalho foi propor um algoritmo genético para a roteirização de veículos na distribuição de urnas eletrônicas destinadas às zonas eleitorais do município de São Paulo, visando minimizar a distância total percorrida pelos veículos e aumentar a eficiência do processo.

A implementação do algoritmo genético possibilitou a simulação de um conjunto de rotas que minimizasse a distância total percorrida para atender todos os pontos de distribuição necessários. A ferramenta computacional MS-Excel também foi utilizada para efeito de comparação das soluções encontradas pelo algoritmo.

No caso do algoritmo desenvolvido a partir da população inicial, o AG aplicou os operadores de seleção, cruzamento e mutação para gerar novas soluções candidatas. No processo de seleção, os indivíduos mais aptos, ou seja, aqueles que apresentaram as rotas com menor distância, tiveram maior chance de serem selecionados para a próxima geração. No cruzamento, os indivíduos selecionados são combinados para gerar novas soluções, e na mutação, pequenas alterações foram introduzidas nas soluções geradas para aumentar a diversidade da população.

Esse processo de seleção, cruzamento e mutação foi repetido diversas vezes, gerando novas populações a cada iteração, até que uma condição de parada fosse atingida. Essa condição pode ser um número máximo de gerações, uma convergência dos melhores indivíduos ou um tempo máximo de execução, nesse caso o critério de parada adotado foi a convergência de melhores soluções.

Para o modelo TSP, o AG implementado se mostrou uma técnica eficiente para encontrar soluções próximas da ótima em um tempo razoável, especialmente para o problema proposto nesse estudo que envolveu diversas variáveis. É importante destacar que, assim como outras técnicas de otimização,

o AG não garante a obtenção da melhor solução possível, mas sim uma solução próxima da ótima em um tempo razoável.

Com base nos resultados obtidos, podemos concluir que o algoritmo genético proposto neste trabalho apresentou resultados satisfatórios na roteirização de veículos para a distribuição de urnas eletrônicas destinadas às zonas eleitorais do município de São Paulo. Embora o algoritmo tenha apresentado uma distância total percorrida um pouco maior em comparação com a solução encontrada pelo Microsoft Excel, o gap encontrado é relativamente pequeno, de apenas 8,5%, a distância total percorrida na rota final foi de 411,4 metros, enquanto a solução encontrada através do software Microsoft Excel foi de 379,1 metros, o que indica uma eficiência muito próxima das soluções encontradas pelo software de planilhas. O algoritmo proposto também obteve um gap 16,8 % com relação a solução ótima da instancia hk48 catalogado no TSPLIB

Além disso, o tempo de processamento computacional do algoritmo genético foi bastante satisfatório, levando 12 minutos e 23 segundos para encontrar uma solução, em contrapartida o software Microsoft Excel obteve melhor desempenho, onde foram necessários 10 minutos e 38 segundos para a geração da solução. O gap encontrado neste caso foi de 14,13%, o que também é considerada uma diferença relativamente pequena.

Portanto, podemos afirmar que o algoritmo genético proposto neste trabalho se mostrou uma alternativa viável e eficiente para a roteirização de veículos na distribuição de urnas eletrônicas destinadas às zonas eleitorais do município de São Paulo. Com os resultados obtidos, espera-se que este trabalho possa contribuir para a melhoria dos processos de logística e distribuição de urnas eletrônicas em futuras eleições, reduzindo custos e aumentando a eficiência do processo.

Cabe destacar que este trabalho apresenta limitações, como a coleta de dados limitada apenas ao município de São Paulo e a falta de validação em situações reais de eleições. Por isso, sugere-se a realização de estudos futuros para aprimorar o algoritmo proposto e testá-lo em outros contextos.

REFERÊNCIAS

BODIN, L.D. (1990), **Twenty Years of Routing and Scheduling, Operations** Research, 38, 4, 520- 524.

BODIN, L.D.; B. Golden; A. Assad e M. Ball (1983) **Routing andscheduling of vehicles and crews: The state of theart. Computers and Operations** Research, vol.10, n.2

Censo da democracia: Brasil tem 147,9 milhões de eleitores aptos a votar nas Eleições 2020. Disponível em: https://www.tse.jus.br/imprensa/noticias-tse/2020/Agosto/brasil-tem-147-9-milhoes-de-eleitores-aptos-a-votar-nas-eleicoes-2020. Acesso em: 02 maio. 2022.

CHAMBERS, L. **The Practical Handbook of Genetic Algorithms: Applications.** 2a. ed. CRC Press: Boca Ratón, FL, 2000

.

COOPER, D. R.; SCHINDLER, P.S. **Metodologia de Pesquisa em Administração**. 10. ed. Porto Alegre: Bookman, 2011

CUNHA, C. B. **Aspectos Práticos da Aplicação de Modelos de Roteirização de Veículos a Problemas Reais**. Transportes, v. 8, n.2, 2000. pp. 51-74.

CUNHA, C. B. **Uma Contribuição para o Problema de Roteirização de Veículos com Restrições Operacionais**. São Paulo, 1997. 222 p. Tese de Doutorado em Engenharia de Transportes – Escola Politécnica, Universidade de São Paulo, 1997.

DANTZIG, G. B.; RAMSER, R.H. **The Truck Dispatching Problem.** Management Science, v. 6, n. 1, p. 80-91, 1959.

DAS, R.; ALVES, G.; AVOSANI, C. **A importância da roteirização no nível de serviço: um estudo na rga operações logísticas e locação ltda**. [s.l: s.n.]. Disponível em: http://www.abepro.org.br/biblioteca/tn_sto_226_319_29604.pdf Acesso em: 14 maio. 2022.

Discrete and Combinatorial Optimization. Disponível em: http://comopt.ifi.uni-heidelberg.de/ Acesso em: 29 maio. 2023.

DORIGO, M.; GAMBARDELLA, L. M. Ant Colony System: A Cooperative **Evolutionary Computation**, v. 1, n.1, p. 53-66, 1997

FOULDS, L. R. **Combinatorial Optimization for Undergraduates. Springer-Verlag,** New York, 1984, p. 114.

GLOVER, F. "**Tabu Search — Part II", ORSA Journal on Computing**, 1990 2: 1, 4-32.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. Addison-Wesley: New York, 1989

.

GOLDEN, B., Ball, M. e Bodin, L. (1981), **Current and future research directions in network optimization, Computers & Operations** Research, 8, 2, 71-81.

GUEDES, Allison da Costa Batista; LEITE, Jéssica Neiva de Figueiredo; ALOISE, Dario José**. Um algoritmo genético com infecção viral para o problema do caixeiro viajante**. Revista Publica, Natal, v. 1, n. 1, p. 16-24, 23 out 2009. Disponível em: https://periodicos.ufrn.br/publica/article/view/125/121 Acesso em: 02 maio. 2022.

GUTIN, G., PUNNEN, A. P. **Combinatorial Optimization: The Traveling Salesman Problem and Its Variations**. Vol. 12. Springer. 2007

HOLLAND, J.H. (1992). **The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance**, Francisco J. Varela, Paul Bourgine, editors. Toward a Practice of Autonomous Systems: proceedings of the first European conference on Artificial Life (1992). MIT Press

KRETZSCHMAR, Luiz Antonio; NUNES, Luiz Fernando; BENEVIDES, Paula Francis. **Analise de resultados para construção de rota para o problema do caixeiro viajante**. In: CONGRESSO DE MATEMÁTICA APLICADA E COMPUTACIONAL - SUDESTE, 34. 2013. 217-222 p. Disponível em: http://www.sbmac.org.br/cmacs/cmac-se/2013/trabalhos/PDF/4374.pd Acesso em: 03 maio. 2022

LINDEN, R. **Algoritmos genéticos**. 3ª ed. Ciência Moderna: Rio de Janeiro, 2012.

LOBO, F. G**. The parameter-less genetic algorithm: Rational and automated parameter selection for simplified genetic algorithm operation**. (Dissertação de Doutorado), Universidade Nova de Lisboa, Lisboa, 2000.

LOESCH, Cláudio, HEIN, Neslon. **Pesquisa Operacional - fundamentos e modelos**. Saraiva, 2008

LOURENÇO, H. R., Martin, O. C., & Stutzle, T. (2003). **Iterated local search. In F. Glover & G. A. Kochenberger (Eds.), Handbook of metaheuristics (pp. 321-353).** Boston: Springer US

MAZZUCCO, JUNIOR, J. **Uma abordagem híbrida do problema da programação da produção através dos algoritmos genéticos e Simulated Annealing** – Tese de Doutorado, Florianópolis: UFSC, 1999.

MCCOMBES, S. **Descriptive Research Design | Definition, Methods and Examples**. Disponível em: https://www.scribbr.com/methodology/descriptive-research/#:~:text=Descriptive%20research%20aims%20to%20accurately . Acesso em: 28 novembro. 2023.

MITCHELL, M. An **Introduction to Genetic Algorithms.** MIT Press: Cambridge, MA, 1996.

NOVAES, A. G. **Logística e gerenciamento da cadeia de distribuição: estratégia, operação e avaliação**. 3. ed. Rio de Janeiro: Elsevier, 2007.

OLIVER, I. M.; SMITH, D. J.; HOLLAND, J. R. C. **A study of permutation crossover operators on the traveling salesman problem. In**: Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application. Hillsdale, NJ, USA: L. Erlbaum Associates Inc.,p. 224–230, 1987.

OSMAN, I. H.; LAPORTE, G. **Metaheuristics: A Bibliography. Annals of Operations Research**, v. 63, n. 5, p. 513-628, 1996.

PRADO BELFIORE, P. et al. **Problema de roteirização de veículos com frota heterogênea: revisão da literatura**. [s.l: s.n.]. Disponível em: http://din.uem.br/sbpo/sbpo2006/pdf/arq0205.pdf Acesso em: 13 maio. 2022.

REEVES, C.R. **Modern Heuristic Techniques for Combinatorial Problems.** John Wiley & Sons. Inc. New York, NY, 1993

SCHWAAB, Cassio dos Santos. **Uma revisão teórica sobre o problema do caixeiro viajante e suas aplicações**. Orientadora: Elizangela Dias Pereira. 2019 Trabalho de Conclusão de Curso (Licenciatura em Matemática) - Universidade Federal do Pampa, 2019

SILVA, F. A. V. **Um Algoritmo genético para o problema de roteamento de veículos com janela de tempo aplicado na distribuição de serviços de telecomunicação**, Universidade Federal do Ceara, Programa de pós-graduação em Logística e Pesquisa Operacional (Dissertação de Mestrado). Fortaleza, 2016.

SILVA, Wederson Adriano Lourenço da. A**lgoritmo Genético para Resolução do Problema do Caixeiro Viajante**. Barbacena, MG, 2010

SOLIANI, R. D.; INNOCENTINI, M. D. M.; CARMO, M. C. Collaborative logistics and eco-efficiency indicators: an analysis of soy and fertilizer transportation in the ports of Santos and Paranaguá. **Independent Journal of Management &**

Production.2020. Disponível em: https://doi.org/10.14807/ijmp.v11i5.1303 Acesso em: 23 maio. 2022.

TAILLARD, É.D. (1999), **A heuristic column generation method for the heterogeneous fleet VRP, RAIRO Recherche Opérationnelle**, 33, 1, 2-10.

TALBI, El Ghazali (2009). **Metaheuristics: From Design to Implementation. isbn**: Vol. 1. WILEY. 2009

TANOMARU, J. **Motivação, fundamentos e aplicações de algoritmos genéticos**. In: Anais do II Congresso Brasileiro de Redes Neurais, 1995. **telecomunicação**, Universidade Federal do Ceara, Programa de pós-graduação em Logística e Pesquisa Operacional (Dissertação de Mestrado). Fortaleza, 2011.

YANG, X.-S. Genetic algorithms (Capítulo de livro). In: **Nature-Inspired Optimization Algorithms**. Oxford: Elsevier, p. 77 – 87, 2014.

YIN, R. K. **Estudo de caso: planejamento e métodos.** 4. ed. Porto Alegre: Bookman, 2010.