UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GIULIA ROSSATTO ROCHA

APLICAÇÃO DE ALGORITMO GENÉTICO PARA ROTEIRIZAÇÃO E CARREGAMENTO DE VEÍCULO

BAURU
2023

GIULIA ROSSATTO ROCHA

APLICAÇÃO DE ALGORITMO GENÉTICO PARA ROTEIRIZAÇÃO E CARREGAMENTO DE VEÍCULO

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru, como requisito para obtenção do título de bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Márcia A. Zanoli Meira e Silva

BAURU
2023

R672a

Rocha, Giulia Rossatto

Aplicação de algoritmo genético para roteirização e carregamento de veículo / Giulia Rossatto Rocha. -- Bauru, 2023

57 p. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru

Orientadora: Márcia Aparecida Zanoli Meira e Silva

1. Meta-heurística. 2. Algoritmo Genético. 3. Roteamento de veículos. I. Título.

GIULIA ROSSATTO ROCHA

APLICAÇÃO DE ALGORITMO GENÉTICO PARA ROTEIRIZAÇÃO E CARREGAMENTO DE VEÍCULO

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru, como requisito para obtenção do título de bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Márcia A. Zanoli Meira e Silva

BANCA EXAMINADORA

Profa. Dra. Márcia A. Zanoli Meira e Silva
Orientadora
Faculdade de Ciências – Unesp/Bauru
Departamento de Computação

Profa. Dra. Simone das Graças Domingues Prado
Faculdade de Ciências – Unesp/Bauru
Departamento de Computação

Profa. Me. Juliana da Costa Feitosa
Faculdade de Ciências – Unesp/Bauru
Departamento de Computação

Bauru, 17 de janeiro de 2023

RESUMO

A globalização é responsável pelo surgimento de um maior número de clientes exigentes quanto à prazos e qualidade de entrega de mercadorias. Nesse sentido, a capacidade em atender às necessidades dos clientes, com qualidade e com baixo custo despendido é uma urgência no contexto de concorrência entre empresas de transporte, distribuição e coleta. A logística e a gestão apresentam-se como estratégias para realizar a organização e planejamento dos recursos empresariais de forma a maximizar a execução de pedidos. No entanto, no âmbito da gestão de transportes há dificuldades no planejamento e roteirização dos veículos envolvidos de modo a determinar o melhor percurso, com menor gasto de tempo e recursos operacionais. O presente trabalho teve por objetivo a construção de um software capaz de realizar a roteirização de um veículo que percorra a menor distância possível entre as localidades pré-definidas por um usuário, considerando uma possível limitação de carga do veículo e uma cidade inicial que servirá como depósito. Foram utilizados Algoritmos Genéticos que são meta-heurísticas baseadas no conceito de evolução dos seres vivos e no processo de seleção natural, visto que o Problema de Roteirização de Veículos Capacitados (PRVC) e que o Problema do Caixeiro Viajante (PCV) são chamados de problemas NP-difícil e, portanto, não são capazes de gerar uma solução ótima em tempo computacional viável com algoritmos polinomiais ou técnicas tradicionais da pesquisa operacional. O software desenvolvido consiste em uma aplicação web, desenvolvida em Python, com a utilização do *micro framework* Flask e do *framework* Bootstrap para estilização das páginas. Finalmente, o algoritmo desenvolvido foi submetido a diversos testes, alterando alguns parâmetros como o processo de seleção, cruzamento e mutação. Verificou-se que o algoritmo genético se apresenta como uma ótima alternativa para a solução do problema, pois permite a utilização de variedades de parâmetros, apresentando ótimos resultados em um tempo positivo. Por fim, verificou-se que o operador de mutação SM não apresentou bons resultados para obtenção da menor distância possível, enquanto os operadores que se destacaram foram os operadores de cruzamento OX e PMX e os operadores de mutação EM, SIM e DM, tanto em questão de tempo quanto em melhor solução obtida.

Palavras-Chaves: Meta-heurística. Algoritmo Genético. Roteamento de veículos.

ABSTRACT

Globalization is responsible for the emergence of a greater number of demanding customers in terms of deadlines and delivery quality. Therefore, the ability to meet customer needs, with quality and at a low cost, is urgent in the context of competition between transport, distribution and collection companies. Logistics and management are presented as strategies to carry out the organization and planning of business resources in order to maximize orders execution. However, in the context of transport management, there are difficulties in planning and routing the vehicles involved in order to determine the best route, with less time and less use of operational resources. The objective of this work was to build a software capable of routing a vehicle that travels the shortest possible distance between locations predefined by a user, considering a possible vehicle load limitation and an initial city that will serve as vehicle deposit. Genetic Algorithms were used, which are meta-heuristics based on the concept of evolution of living beings and on the process of natural selection, since the Capacitated Vehicle Routing Problem (CVRP) and the Traveling Salesman Problem (TSP) are called NP-hard problems, therefore, are not able to generate an optimal solution in computational time feasible with polynomial algorithms or traditional operations research techniques. The software presented is a web application, developed in Python, using the microframework Flask and the framework Bootstrap for page styling. Finally, the developed algorithm was submitted to several tests, changing some parameters such as the selection, crossover and mutation process. It was verified that the genetic algorithm presents itself as a great alternative for the problem solution, since it allows the use of a variety of parameters, presenting excellent results in a positive time. Finally, it was verified that the mutation operator SM did not present good results for obtaining the smallest possible distance, while the operators that stood out were the crossover operators OX and PMX and the mutation operators EM, SIM and DM, regarding the matter of time and the best solution obtained.

Keywords: Metaheuristics. Genetic Algorithm. Vehicle routing.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho propôs a elaboração de um software capaz de tratar o problema de roteirização de um veículo, com limitação de carga, responsável por atender a demanda de diversas cidades, de modo que o meio de transporte parta de uma cidade de origem e retorne para ela. Para isso, caso o veículo atinja sua capacidade máxima em um determinado ponto da rota, este deve retornar à cidade inicial para descarregamento, considerando a cidade de origem uma espécie de depósito para o roteiro. Dessa forma, o problema resolvido é uma variação do Problema do Caixeiro Viajante (PCV) e do Problema de Roteamento de Veículos Capacitados (PRVC). Considerando essas variações foram aplicados algoritmos genéticos com a finalidade de obter uma solução viável em um tempo computacional possível e aceitável.

A necessidade desta solução se deve ao fato de que a globalização promoveu o surgimento de clientes mais exigentes quanto a qualidade e prazos de entrega de produtos e, consequentemente, estimulou a competitividade do mercado de distribuição e coleta de mercadorias, juntamente com a necessidade de transporte. Nesse contexto, muitas empresas de transporte têm buscado investir em velocidade, flexibilidade, eficiência e pontualidade na entrega e/ou coleta de produtos, melhor aproveitamento da frota de veículos e planejamento de rotas mais adequadas de modo a diminuir tempo, distância, recursos operacionais e, consequentemente, gerar uma melhoria da imagem do empreendimento e satisfação de seus clientes (MELO; FILHO, 2001).

Nesse sentido, Melo e Filho (2001) apontam que sistemas de roteirização de veículos são capazes de produzir soluções para problemas de planejamento de rotas de transportes, geralmente utilizando métodos heurísticos, obtendo baixo tempo e esforço de processamento quando comparado a métodos manuais.

A classe de Problemas de Otimização Combinatória (POC) são complicações reais encontradas no cotidiano que envolvem dificuldades de alocação, roteamento e programação de recursos. Visando resolver este tipo de problema, torna-se crescente o desenvolvimento de algoritmos eficientes para essa finalidade. Dentre as dificuldades dos POC, tem-se o PCV que, dado um conjunto de nós interligados, busca encontrar o menor caminho e custo possível, visitando todos os nós viáveis e retornando ao ponto de origem (PRESTES, 2006).

O trabalho aqui apresentado teve como base o PCV, pois de acordo com Prestes (2006), trata-se de um problema NP-Difícil, ou seja, não é viável a utilização de algoritmos polinomiais para sua solução, visto que há um crescimento exponencial de soluções possíveis em relação

ao número de nós do conjunto do problema. Além disso, segundo Junior e Silva (2015), o PCV foi um dos primeiros problemas usados para aprimorar o problema de roteirização de veículos.

Nesse contexto, o PRVC, no qual este trabalho também se baseia, pode ser considerado uma generalização do PCV que busca determinar um roteiro viável de uma frota mínima de veículos, com limite de capacidade, para atender localidades distintas, na menor distância e custo possível. Cada nó deve ser visitado uma única vez e o veículo não pode ter seu limite de capacidade excedido. Ao final, a frota deve retornar à localidade de origem (NÉIA et al., 2013).

Para Junior e Silva (2015) é viável a utilização de heurísticas para problemas de roteirização visto que a busca de soluções exatas, em um tempo computacional válido, não é ideal para esse tipo de problema. Conforme explica Prestes (2006), as heurísticas consistem em algoritmos nos quais não se encontra a melhor solução para o problema (solução ótima), mas sim uma solução viável, em tempo aceitável para as necessidades da aplicação.

Como a aplicação de algoritmos polinomiais para resolver este tipo de problema não é viável, uma solução é a utilização de algoritmos genéticos (AGs). De acordo com Lacerda e Carvalho (1999), esses algoritmos são inspirados no mecanismo de evolução de populações de seres vivos e no princípio da seleção natural, que buscam pela melhor solução de um determinado problema.

Consistem em um algoritmo iterativo que começa com uma população inicial de cromossomos, ou seja, um conjunto de possíveis soluções para o problema. A cada iteração, os cromossomos são avaliados por uma função de aptidão pré-definida, que representa o quão viável o cromossomo é para a solução do problema. Em seguida ocorre o processo de seleção, através da escolha dos melhores indivíduos com base em sua aptidão para o cruzamento (*crossover)* gerando cromossomos filhos e os piores são descartados. Esse processo iterativo continua até que uma solução satisfatória seja identificada, de acordo com o critério de parada escolhido (LACERDA; CARVALHO, 1999).

Cada iteração do AG é caracterizada por uma etapa de seleção no qual alguns cromossomos são selecionados para gerar cromossomos filhos por meio de operadores de cruzamento e mutação. Há mais de uma forma de realizar a implementação dessas etapas, como a seleção por torneio no qual os cromossomos pais são selecionados aleatoriamente, ou a seleção por roleta no qual os pais são selecionados com probabilidade proporcional a sua aptidão. Além disso, há diversas variações de operadores de cruzamento e mutação que podem ser utilizados para alcançar os resultados desejados (LACERDA; CARVALHO, 1999). Considerando esses fatos, este trabalho buscou explorar estas diferentes variações a fim de estabelecer comparações.

Néia et al. (2013) propõe que para o PCV, um cromossomo pode ser representado como um vetor de genes que determinaria uma possível sequência de localidades a serem visitadas pelo veículo. Logo, cada gene correspondente a uma posição do vetor representa uma localidade específica a ser visitada pelo veículo.

Finalmente, é válido destacar que este trabalho consiste em uma melhoria do projeto de Iniciação Científica desenvolvido pela autora (Edital 4/2021 - PIBIC/Unesp - ID 4383). No trabalho atual acrescentou-se a implementação de mais um método de seleção, chamado seleção por torneio, enquanto o projeto de Iniciação Científica utilizou apenas da seleção por roleta. Também foram incluídos um operador de cruzamento, denominado cruzamento ordenado, e um método que seleciona uma técnica de cruzamento aleatória a cada iteração do algoritmo. Quanto aos operadores de mutação, além dos operadores já utilizados no projeto de Iniciação Científica (operadores de mutação por troca e inversão simples) foram implementados os operadores de mutação por deslocamento, inserção, inversão, embaralhamento e um método aleatório, que também seleciona um dos operadores de mutação aleatoriamente a cada iteração do AG, assim como o de cruzamento.

Além dos operadores adicionais implementados, também foi desenvolvida uma melhoria no método de utilização do elitismo, no qual o usuário é responsável por identificar a taxa de elitismo que deseja adicionar às populações. A interface gráfica com o usuário também foi aprimorada, de modo a fornecer uma experiência mais simples e eficiente na inserção de dados. E por fim, foram aplicadas algumas estratégias de comparações entre os operadores utilizados, visando obter uma análise mais eficaz das diferentes implementações dos algoritmos aplicados ao problema.

1.1 Objetivos Gerais

O objetivo geral deste trabalho foi desenvolver um software que realize a roteirização de um veículo com capacidade de carga limitada obtendo a menor rota possível entre as localidades fornecidas por um usuário. Para a solução deste problema foram utilizados algoritmos genéticos e suas diferentes variações de seleção e operadores (cruzamento e mutação), possibilitando realizar uma análise comparativa entre estes.

1.2 Objetivos Específicos

Os objetivos específicos desta pesquisa, foram:

- Realizar um levantamento bibliográfico sobre problemas de logística e transporte, sobre o PCV, o PRVC com limite de carga e sua solução por meio do uso de AG e seus diferentes operadores;
- Desenvolver o AG capaz de calcular a melhor rota de um veículo entre as localidades inseridas pelo usuário, respeitando seu limite de carga;
- Implementar e caracterizar dois métodos de seleção de indivíduos para reprodução no AG;
- Implementar e caracterizar diferentes métodos de cruzamento de indivíduos no AG;
- Implementar e caracterizar diferentes métodos de mutação de indivíduos no AG;
- Utilizar elitismo para o AG;
- Desenvolver uma aplicação responsável por realizar a integração do AG e a interface com o usuário;
- Realizar diferentes testes para identificar possíveis falhas e realizar correções na aplicação; e
- Pesquisar e elaborar técnicas de comparação entre os operadores do algoritmo de forma a identificar as mais adequadas ao problema.

1.3 Justificativa

Para Gomes et al. (2019), o transporte de cargas é um dos aspectos essenciais nos custos logísticos de empresas e estas precisam antecipar as tendências de mercado. Para atender clientes de forma rápida, eficiente e com qualidade, é justificável o uso de técnicas de pesquisa operacional para roteirização de veículos e uso de ferramentas computacionais para otimização desses processos para minimização de custos.

Assim, é válido ressaltar a utilização de meta-heurísticas, ou seja, métodos iterativos de inteligência computacional capazes de realizar busca de soluções adequadas para a resolução de um problema em tempo válido. Dentre eles, os AGs apresentam-se como métodos eficazes para sistemas de roteirização de veículos e problemas de distribuição, visto que são computacionalmente simples e suficientemente capazes de fornecer soluções otimizadas para o problema. Esses algoritmos são capazes de gerar rotas de menor custo considerando a carga do veículo utilizado (MALAQUIAS, 2006).

Como visto anteriormente, na escolha dos AGs considerou-se que o PCV e o PRVC são problemas NP-Difícil e, portanto, não são capazes de gerar uma solução ótima em tempo computacional válido. Por outro lado, as heurísticas e meta-heurísticas são capazes de fornecer uma solução válida em tempo computacional aceitável.

A escolha pelo desenvolvimento do próprio AG aplicado ao problema deste trabalho se deve à argumentação de Linden (2012) de que um AG só funciona corretamente se for atribuído a ele o máximo de particularidades do problema a ser resolvido, principalmente na função de aptidão quanto na codificação dos cromossomos e na escolha dos operadores genéticos. Caso contrário, AGs genéricos podem possuir desempenho inferior àqueles especificamente projetados para um problema.

1.4 Metodologia

Inicialmente, foram utilizadas técnicas de levantamento bibliográfico sobre a necessidade da roteirização de veículos com limite de carga e dos problemas de logística e transporte. Além disso, foram utilizadas técnicas de apreensão bibliográfica sobre a caracterização, utilização e implementação dos AGs como pilar principal para solução dos impasses do presente trabalho. Por fim, também foi realizada uma pesquisa exploratória a fim de aplicar técnicas de comparação entre os operadores utilizados no desenvolvimento do AG.

Esta pesquisa possui uma abordagem de desenvolvimento, pois seu objetivo principal consistiu na implementação de uma aplicação web, inicialmente local, que realiza a integração do AG desenvolvido com uma interface gráfica com o usuário.

A aplicação é dividida no *back-end* e no *front-end*. O *back-end* é composto pela implementação do AG na linguagem de programação Python, com a utilização do *micro-framework* Flask. O *front-end* corresponde ao desenvolvimento da interface gráfica com o usuário da aplicação. Para sua implementação utilizou-se o *framework* Bootstrap que permite a criação de interfaces gráficas com diferentes complexidades, e a estilização das páginas web foram criadas utilizando *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CSS) e *JavaScript* (JS).

Para a realização dos testes dos algoritmos implementados e exploração de possíveis erros de execução, foram selecionadas 20 cidades aleatoriamente. Para elaborar a matriz de distâncias entre essas cidades foi utilizado o Google Maps manualmente, obtendo uma matriz de distâncias de dimensão 20x20. Além disso, as demandas de cada cidade a serem atendidas pelo veículo foram escolhidas proporcionalmente à quantidade populacional de cada uma.

Por fim, para as comparações realizadas sobre os dados obtidos considerou-se o tempo; o melhor valor e a média dos resultados obtidos em todas as combinações de seleção, cruzamento e mutação; o efeito do elitismo sobre as populações; a convergência das combinações dos operadores utilizados; e o impacto do aumento e diminuição do limite de carga do veículo que realizará o roteiro programado.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados conceitos essenciais para a compreensão do trabalho. Nele serão abordados conceitos e estudos de logística e transporte, uso de Pesquisa Operacional e técnicas de otimização, o que são Problemas de Otimização Combinatória, especificamente o PCV e o PRVC, além de processos de otimização aplicados a estes problemas e dificuldades encontradas nas abordagens tradicionais. Além disso, serão introduzidas heurísticas e meta-heurísticas para resolução de conflitos encontrados nas abordagens da pesquisa operacional tradicional e, por fim, um detalhamento dos AGs e seu funcionamento como uma meta-heurística eficiente para solução do problema apresentado neste trabalho.

2.1 Logística e Transporte

Pacífico e Silva Júnior (2020) afirmam que a concorrência entre corporações e a exigência de novas estratégias para garantir a sobrevivência de uma empresa surgem em grande escala com a globalização. A globalização é responsável por exigir empresas dinâmicas, com capacidade de decisão mais ágil, precisa e menos complexa frente aos processos administrativos e operacionais. Nesse contexto, a logística apresenta-se como um importante papel para a sobrevivência das organizações.

Christopher (2018) aponta que a logística e a gestão, fortemente acopladas, são conceitos utilizados ao longo de toda a história da humanidade, influenciando em guerras através do planejamento de suprimentos e localização de exércitos, com grandes construções da história, entre outros. O autor também define:

> Logística é o processo de gestão estratégica da aquisição, movimentação e armazenagem de materiais, peças e estoques finais (e fluxos de informação relacionados) por meio da organização e seus canais de comercialização, de tal forma que as rentabilidades atual e futura sejam maximizadas através da execução de pedidos, visando ao custo-benefício. (CHRISTOPHER, 2018, p. 12).

Dessa forma, é possível afirmar que o atendimento dos clientes com a melhor relação custo-benefício é a maior missão da gestão logística. A posição de superioridade permanente de uma empresa em um ambiente globalizado e concorrente, no quesito de satisfação de clientes, pode ser alcançada pela melhor gestão de logística e da cadeia de suprimentos (CHRISTOPHER, 2018).

Nesse contexto, pode-se destacar que a logística de transportes representa um papel importante nas empresas visto que procura realizar seus processos de coleta ou distribuição

atendendo às necessidades de seus clientes, garantindo satisfazer suas exigências quanto ao local, horário e qualidade (FRANCISCO; GILBERTO, 2018). O transporte apresenta-se como um fator essencial na cadeia de suprimentos pois garante a distribuição de materiais e fornecimento de recursos produtivos para as corporações. Logo, quanto melhor a eficiência no transporte, menor o tempo em trânsito de produtos e melhor a qualidade do serviço aos consumidores finais (GOMES et al., 2019).

Nesse âmbito, a área de Pesquisa Operacional (PO) pode ser aplicada a problemas de gestão de transportes para identificar e especificar restrições e gargalos no sistema logístico, promovendo pontos de melhorias e otimização por meio de modelos matemáticos e formulações de algoritmos, auxiliando assim em uma melhor tomada de decisão em corporações empresariais. Na área de PO, estudos são realizados em problemas de roteamento de veículos de modo a obter as melhores rotas possíveis com o mínimo custo, tempo e distância percorrida pelo veículo. A tecnologia da informação e técnicas operacionais podem propiciar um processo logístico eficiente e de qualidade entre as etapas da cadeia de suprimento, como estoque, transporte e custo (FRANCISCO; GILBERTO, 2018).

2.2 Pesquisa Operacional e Técnicas de Otimização Combinatória

É notável que o campo de estudo da PO é uma ferramenta importante em todos os níveis de gestão, pois promove a eficiência e eficácia organizacional. Afinal, trata-se de um estudo tradicional que reúne diversas técnicas da modelagem matemática a fim de estruturar e solucionar problemas quantitativos que podem ser expressos matematicamente (GOLDBARG; LUNA, 2005).

Nesse contexto, problemas de estoque, transporte, planejamento, produção, coordenação, entre outros, são analisados, formulados e solucionados principalmente pelas áreas de Pesquisa Operacional, Ciência da Computação e Engenharia. Logo, para solução desses problemas são empregados técnicas de Otimização Combinatória e Programação Inteira. Quanto à Otimização Combinatória, trata-se de uma disciplina que analisa problemas que podem ser formulados como problemas de programação inteira e que também podem ser formulados por meio de análise combinatória. Estes são solucionados por meio de Algoritmos de Programação Inteira ou como problemas em grafos por meio de algoritmos específicos (GOLDBARG, E.; GOLDBARG, M.; LUNA, 2021).

Para Prestes (2006), os problemas de Otimização Combinatória consistem em atribuir valores a um conjunto de variáveis de decisão visando otimizar a função objetivo, minimizando-

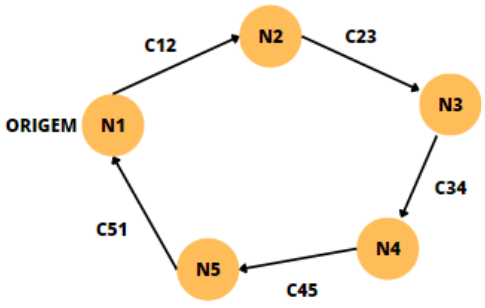
a ou maximizando-a. Para o autor, estes problemas têm sido utilizados em diversas situações reais, no entanto, grande parte desses pertencem à classe de problemas NP-árduos ou NP-difícil, ou seja, são problemas que raramente serão resolvidos com algoritmos polinomiais.

De acordo com Goldbarg e Luna (2005), dentre os problemas de otimização combinatória, há alguns problemas clássicos como o do caixeiro viajante, de corte, de empacotamento, de percurso de veículos, entre outros. Para resolução desses problemas, os autores alertam que a busca da otimalidade global é sempre o ideal a ser seguido, no entanto, visto algumas complexidades do mundo real, pode ser necessário o uso de técnicas heurísticas clássicas e meta-heurísticas, com complexidade numérica polinomial, para obtenção de boas soluções.

2.2.1 Problema do caixeiro viajante (PCV)

O PCV é considerado um dos problemas mais tradicionais de otimização e da programação matemática (GOLDBARG; LUNA, 2005). O problema é definido a partir de um grafo G = (N, E) tal que N = {1, ..., n} representa o conjunto de nós ou vértices e E = {1, ..., m} o conjunto de arestas de G. Os nós referem-se às localidades que devem ser visitadas, sejam elas cidades, endereços, entre outros. Logo, para interligação desses nós, existem arestas que ligam os vértices i e j, considerando um custo de deslocamento de c_{ij}. Dessa forma, o problema consiste em determinar o menor ciclo do grafo com o menor custo possível, determinado pela soma do custo de todas as arestas percorridas. Além disso, torna-se necessário que todos os vértices sejam visitados apenas uma vez e que inicie de um nó origem e retorne ao mesmo (PRESTES, 2006). Um exemplo de grafo do PCV pode ser visto na Figura 1, no qual existem cinco nós e cinco arestas e o caixeiro viajante parte do nó origem N1.

Figura 1 - Exemplificação em grafo do PCV



Fonte: Elaborada pela autora

Prestes (2006) explica que existem diversas abordagens para resolver todos os tipos de instâncias desse problema, visto que ele possui uma grande aplicabilidade em problemas reais. No entanto, essas abordagens podem ser divididas em duas grandes áreas: métodos exatos e métodos heurísticos.

Quanto à abordagem por métodos exatos, a forma mais intuitiva seria testar todas as combinações possíveis. No entanto, essa abordagem torna-se inviável pois o PCV consiste em um problema considerado NP-árduo, logo, as possibilidades de combinações crescem exponencialmente em relação ao número de nós do grafo, ou seja, em relação ao número de localidades que devem ser visitadas. Outras abordagens baseadas na Programação Inteira permitem obter ou provar a otimalidade de uma solução obtida em um tempo de execução finito, como por exemplo: *Branch and Bound*, *Branch and Cut*, *Branch and Price*, Programação Dinâmica e Relaxação Lagrangiana (PRESTES, 2006).

Dentre as vantagens no uso das abordagens exatas citadas anteriormente, tem-se que elas são capazes de provar que soluções ótimas podem ser obtidas, de apresentar informações essenciais sobre limites inferiores e superiores caso não consiga finalizar sua execução e de descartar espaços de buscas em que a solução ótima não pode ser encontrada. No entanto, como ponto negativo, esses métodos podem exigir alto custo computacional visto que o tempo de processamento cresce exponencialmente em relação ao aumento das instâncias do problema e, consequentemente, gera um alto consumo de memória, levando a interrupção do programa antes do previsto (PRESTES, 2006).

Quanto às abordagens por métodos heurísticos, elas surgem para solucionar as dificuldades na implementação de métodos exatos, possibilitando boas soluções aproximadas. Dentre elas, temos a Busca Tabu, *Simulated Annealing* (SA), *Greedy Randomized Adaptive Search Procedures* (GRASP), Algoritmos Genéticos, entre outros (PRESTES, 2006). Tais abordagens serão detalhadas posteriormente na seção 2.3.

2.2.2 Problema de roteamento de veículos capacitados (PRVC)

O Problema de Roteamento ou Roteirização de Veículos Capacitados (PRVC) consiste em identificar o melhor roteiro ou conjunto de roteiros de veículos, minimizando o custo total de transporte, de modo a atender a demanda de diversos clientes localizados em pontos distintos, garantindo que cada um seja visitado somente uma vez e que os veículos utilizados não excedam seus limites de capacidade. Além disso, trata-se de uma frota homogênea de veículos que partem do mesmo depósito e devem retornar a ele. Embora não seja objeto deste

estudo, quando são envolvidas variáveis referentes a janela de tempo, duração de roteiro e horários de atendimento, o problema é considerado um problema de roteamento e programação de veículos (NÉIA et al., 2013).

Segundo Néia et al. (2013), o problema de roteamento de veículos é uma generalização do PCV, pois consiste em utilizar m caixeiros para atender n localidades (clientes) e possui variações ao considerar outras variáveis, como capacidade de carregamento do veículo para atender clientes que possuem demandas a serem transportadas, duração máxima de viagem, entre outros.

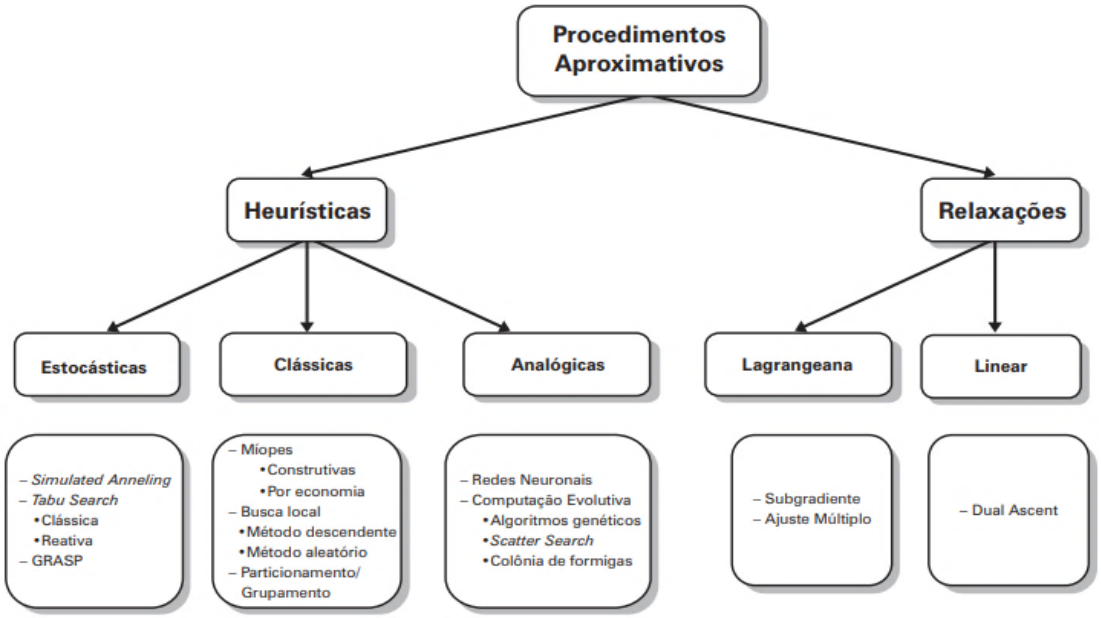
2.3 Heurísticas e Meta-Heurísticas

Conforme apontam Goldbarg e Luna (2005), problemas de programação linear inteira (PLI) são denominados NP-árduos. Tal característica é marcada pela dificuldade em encontrar soluções exatas devido ao grande número de combinações de soluções possíveis para serem processadas computacionalmente em um tempo possível. Por exemplo, seja um problema com n variáveis que desenvolverá, no mínimo $2^{\frac{n-1}{2}}$ nós. Supondo que o problema possua 201 variáveis, haveria em torno de 2^{100} combinações possíveis. Mesmo com as técnicas e avanços atuais, se um computador executasse 1,5 trilhões de nós por segundo, ele levaria 537 milhões de anos para decifrar todas as combinações existentes.

Nesse contexto, as heurísticas têm se tornado importantes alternativas para resolução de problemas de otimização combinatória. Também chamados de procedimentos aproximativos, os algoritmos heurísticos procuram explorar espaços de buscas em que não são garantidos resultados de sucesso e que utilizam esforço computacional coerente com as necessidades da aplicação de forma a "garantir a viabilidade ou a otimalidade da solução encontrada" (GOLDBARG; LUNA, 2005, p. 196). Esses algoritmos podem ser classificados em estocásticos, clássicos e analógicos, como mostra a Figura 2.

Quanto aos algoritmos meta-heurísticos, que envolvem as classificações estocásticas e analógicas e, portanto, os algoritmos genéticos, Goldbarg e Luna (2005) definem como algoritmos cujas estratégias não dependem do problema específico, elas são adaptadas ao caso em questão. Enquanto as heurísticas clássicas exploram todos os âmbitos do problema sem definir uma estratégia universal de solução.

Figura 2 - Classificação das abordagens aproximativas



Fonte: Goldbarg e Luna (2005)

Nesse contexto, o uso de heurísticas para encontrar soluções para problemas do caixeiro viajante consiste em elaborar um roteiro viável a partir de um conjunto inicial de nós que será modificado a cada iteração de acordo com um critério de escolha. A abordagem, no entanto, não garante a solução ótima ou o quão perto dela a solução viável se encontra (BENEVIDES, 2011).

2.4 Algoritmo Genético (AG)

Introduzido por John Holland em 1975 e popularizado por seu aluno David Goldberg, os AGs são técnicas meta-heurísticas de otimização e busca inspirados nos mecanismos de evolução e seleção natural de Charles Darwin (LACERDA; CARVALHO, 1999).

Os AGs são algoritmos evolucionários ou evolutivos, pois se baseiam no processo biológico da evolução natural, que consiste em fazer competir diversos indivíduos pelo processo de seleção natural, onde os mais aptos sobrevivem. Essa competição entre os indivíduos é o que determinam as soluções obtidas (LINDEN, 2012).

Linden (2012) indica que os AGs não são técnicas determinísticas e sim probabilísticas, logo, com os mesmos parâmetros utilizados e mesmos conjunto de dados é possível obter soluções diferentes a cada execução. Além disso, são algoritmos simples que necessitam apenas de informações relativas à adequabilidade da solução, permitindo que sejam extremamente aplicáveis a diversas situações do mundo real em que são necessários alocar recursos.

Esses algoritmos trabalham com grande número de pontos possíveis em um espaço de soluções, no entanto, não procuram em todos os pontos existentes. Nesse sentido, são muito empregados em PCVs e similares. Ao invés de procurar a solução em todo o conjunto de soluções possíveis (proporcional ao fatorial do número de nós do problema), o AG procura por uma proporção significativa das soluções possíveis de acordo com a definição dos parâmetros, caso contrário, seriam necessários dias para completar a execução do algoritmo e obtenção da solução final (LINDEN, 2012).

2.4.1 Funcionamento e terminologia básica

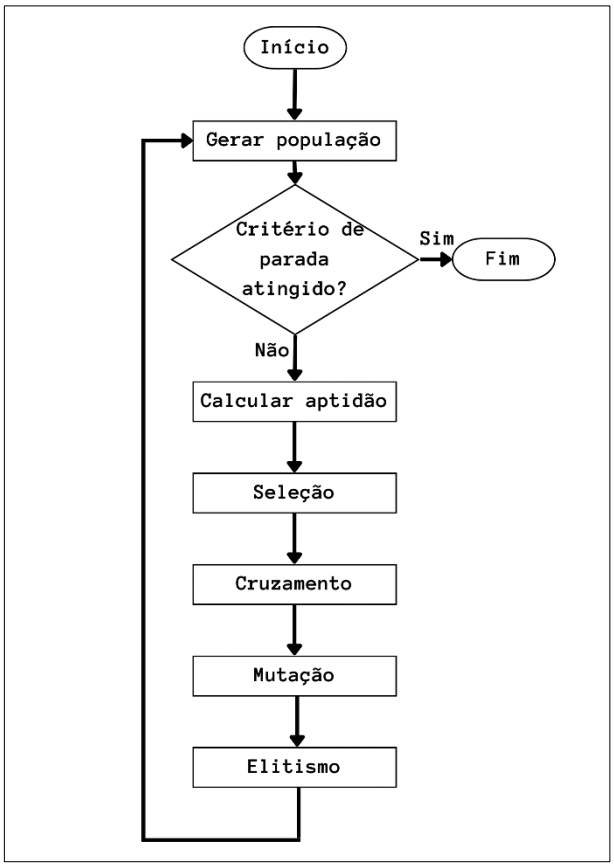
O AG processa populações de cromossomos ou indivíduos. Um cromossomo retrata uma possível solução para o problema e é representado por uma estrutura de dados, geralmente por um vetor ou estrutura de bits. Dessa forma, considerando que um cromossomo retrata uma solução possível, seu conjunto de todas as configurações possíveis representa o espaço de busca do problema: se o cromossomo exprime n parâmetros de uma função, n dimensões reflete o espaço de busca (LACERDA; CARVALHO, 1999).

O AG começa com a geração de uma população inicial de cromossomos ou indivíduos. Como se trata de um algoritmo iterativo, cada ciclo é chamado de geração e a cada geração a população passa por um processo evolutivo. Esse processo evolutivo avalia cada cromossomo de toda a população por meio de uma função de aptidão que representa o quão viável é essa solução para o problema. Os melhores indivíduos representam os indivíduos mais aptos e, portanto, aqueles que serão selecionados para a próxima geração. Esse processo é denominado de seleção, e esses cromossomos selecionados podem sofrer modificações em suas características por meio dos operadores de cruzamento ou *crossover*, do inglês, que geram novos indivíduos, e dos operadores de mutação que alteram a codificação genética dos cromossomos. Esse processo iterativo se repete até que uma solução satisfatória seja encontrada ou um critério de parada seja satisfeito (LACERDA; CARVALHO, 1999).

Segundo Lacerda e Carvalho (1999), o algoritmo consegue explorar diferentes e desconhecidos espaços de busca através dos operadores de *crossover* e mutação.

Esses passos, que consistem no funcionamento básico do AG, são visualizados no fluxograma apresentado na Figura 3. A etapa especificada chamada de Elitismo será apresentada na subseção 2.4.8.

Figura 3 - Fluxograma do Algoritmo Genético



Fonte: Elaborada pela autora

2.4.2 Representação cromossomial

Linden (2012) argumenta que a forma da representação do cromossomo é essencial para o funcionamento do AG e varia de acordo com a formulação do problema a ser resolvido. Essa representação corresponde a tradução da informação do problema em uma maneira possível de ser tratada pelos recursos computacionais. Como Lacerda e Carvalho (1999) representam cada cromossomo por uma estrutura de dados (um vetor, por exemplo), Linden (2012) explica que cada pedaço indivisível dessa representação é denominado gene, portanto, cada posição do vetor é um gene.

2.4.3 População inicial

A população inicial deve ser obtida da maneira mais simples possível, portanto, realizando uma escolha aleatória independente para geração de cada indivíduo da população. Dessa forma, é possível obter uma boa distribuição das soluções no espaço de busca, embora

não garanta exatamente que todos os espaços de busca sejam atingidos, visto que a população tem um tamanho limitado (LINDEN, 2012).

2.4.4 Função de aptidão

A função de aptidão consiste em uma forma de avaliação dos cromossomos, ou seja, verificar quão viável eles são para a solução do problema. Ela é atribuída a cada indivíduo auxiliando no processo de seleção do AG, separando os indivíduos mais aptos para serem transferidos para o próximo ciclo. Desta forma, ele utiliza os valores dos genes de cada cromossomo para produzir um valor numérico que representa a qualidade deste indivíduo (LINDEN, 2012). Lacerda e Carvalho (1999) argumentam que há várias alternativas para definir a aptidão e a mais simples iguala a aptidão ao valor da função objetivo.

2.4.5 Seleção

O processo de seleção é inspirado no processo de seleção natural biológico e consiste em selecionar os melhores cromossomos da população da geração atual para gerar cromossomos filhos por meio dos operadores de *crossover* e mutação. A seleção dos cromossomos pais é realizada de acordo com o valor de sua aptidão (LACERDA; CARVALHO, 1999).

Linden (2012) afirma, no entanto, que indivíduos menos aptos também podem ser capazes de gerar descendentes, mesmo que os mais aptos sejam mais privilegiados. Isso se deve ao fato de que os pais com piores aptidões também podem ter características genéticas importantes que devem ser passadas para as próximas gerações e que podem não estar presentes em outros indivíduos. É importante que esse processo ocorra de forma justa para evitar um processo chamado de convergência genética, no qual a população tende a ser composta de indivíduos muito semelhantes e sem diversidade genética para que a evolução possa prosseguir de forma satisfatória.

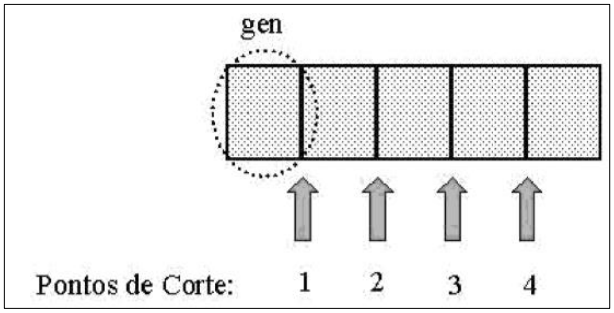
Um método muito utilizado é o método da roleta, no qual calcula-se a aptidão acumulada de todos os indivíduos e em seguida é gerado um número aleatório que se encontre no intervalo que abrange toda a soma de aptidão acumulada. Esse valor seleciona o primeiro cromossomo que possui a aptidão acumulada maior que o valor obtido aleatoriamente. Logo, uma cópia deste indivíduo é alocada à nova população. Além desse método, existem outras

formas de seleção que podem ser implementadas, como o método do torneio (LACERDA; CARVALHO, 1999).

2.4.6 Operador de cruzamento

Conforme Linden (2012) explica, operadores de *crossover* são aplicados a cromossomos resultantes do processo de seleção de forma a gerar cromossomos filhos. O autor explica que a operação se assemelha ao processo que ocorre na reprodução sexuada dos seres vivos durante a formação dos cromossomos. Podem existir operações de *crossover* complexas aplicadas a diferentes problemas. Um exemplo simples é o *crossover* de um ponto, no qual é escolhido um ponto de corte entre dois cromossomos pais, ou seja, uma posição entre dois genes de um cromossomo. A Figura 4 apresenta um exemplo de todas as opções de corte em um cromossomo.

Figura 4 - Exemplos de pontos de corte



Fonte: Linden (2012)

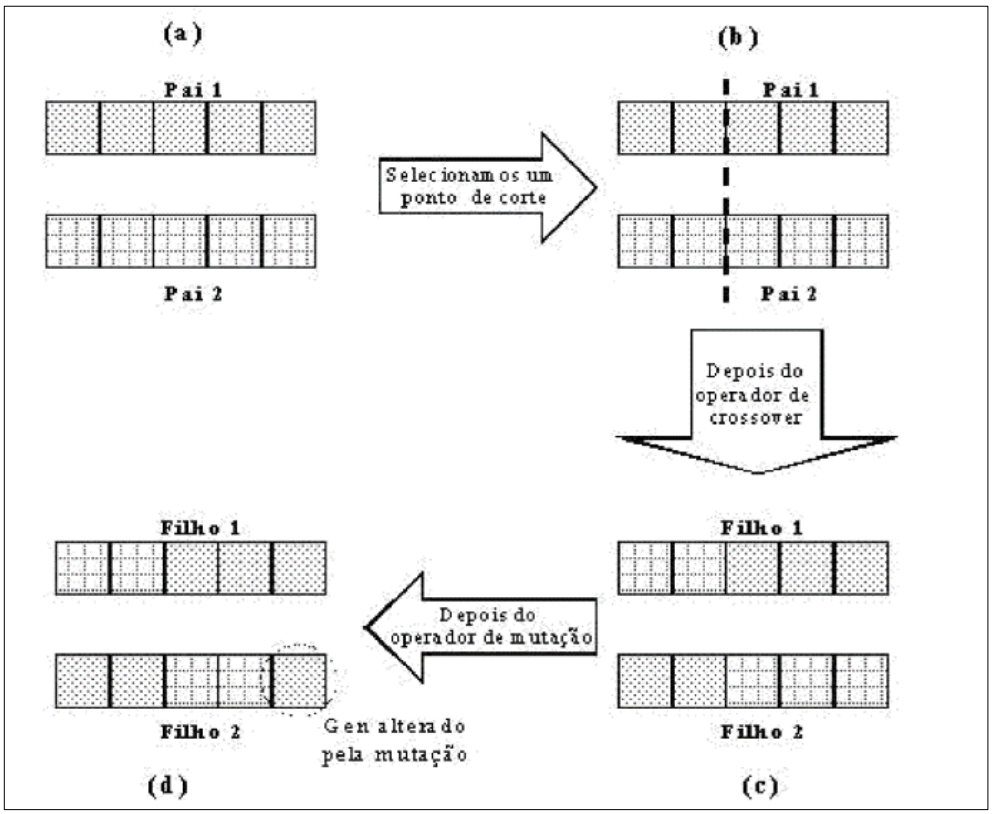
Escolhido o mesmo ponto de corte para os dois cromossomos, os pais são separados em duas partes, uma à esquerda e outra à direita. Assim, o primeiro filho é formado pela junção da parte esquerda do primeiro pai com a parte direita do segundo pai, enquanto o segundo filho é formado com a parte esquerda do segundo pai e com a parte direita do primeiro. O processo é exemplificado na Figura 5 entre as etapas (b) e (c).

2.4.7 Operador de mutação

Segundo Lacerda e Carvalho (1999), o operador de mutação é aplicado logo após o processo de cruzamento com uma certa probabilidade de acontecer, alterando a informação contida em um ou mais genes de um cromossomo. A mutação é responsável por aumentar a diversidade dos cromossomos na população. No entanto, segundo os autores, a mutação deve

ocorrer a uma taxa pequena, visto que ela destrói a informação contida na população, mas que assegure a ocorrência de diversidade. A Figura 5 exemplifica o resultado da mutação entre os passos (c) e (d).

Figura 5 - Exemplo da operação de crossover de um ponto e mutação



Fonte: Linden (2012)

2.4.8 Elitismo

Dependendo do tipo de implementação do processo de seleção, principalmente se for aleatório ou probabilístico, e dos processos de *crossover* e mutação, que modificam as estruturas dos cromossomos, os melhores indivíduos podem ser perdidos de uma geração para a outra. Nesse sentido, o elitismo consiste em transferir o melhor cromossomo da geração atual para a próxima. Esse processo tende a garantir uma convergência mais rápida do AG, garantindo a busca pela solução de forma mais rápida, ou seja, em menor número de ciclos (LACERDA; CARVALHO, 1999).

3 SOFTWARE DESENVOLVIDO

O software construído consiste em uma aplicação web, inicialmente local, dividido no *back-end* onde é realizado todo o processamento do AG com a linguagem de programação Python e toda a construção e coordenação das rotas necessárias para a aplicação web com o *micro framework* Flask. O *front-end* consiste na construção da interface gráfica com o usuário, desenvolvido em Bootstrap, HTML, CSS e JS.

3.1 Funcionamento

A seguir será explicado o funcionamento básico do software, a ordem de execução das telas, assim como as demandas de entrada do usuário para o funcionamento correto do sistema.

3.1.1 Planilha com dados das cidades e suas distâncias

Antes de apresentar as telas utilizadas para o software, o programa exige a utilização de uma planilha no formato XLSX. Nesta planilha são necessárias duas páginas, uma denominada “Lista das Cidades” e outra denominada “Matriz de Distancias”. A primeira página, correspondente a “Lista das Cidades”, deve conter a numeração desejada das cidades, o nome delas e a quantidade de carga que cada uma demandará. A Figura 6 apresenta um exemplo de como a primeira página deve ser preenchida.

Figura 6 - Página “Lista das Cidades” da planilha utilizada para testes do software

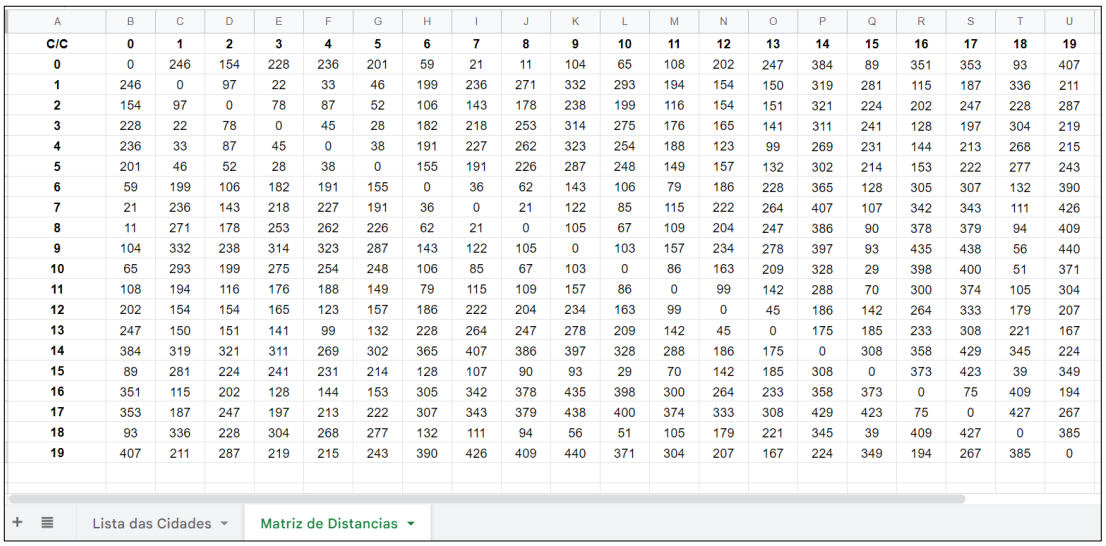
	A	B	C	D
1	**N**	**Cidade**	**Carga**	
2	0	Sorocaba	50	
3	1	Bauru	40	
4	2	Botucatu	30	
5	3	Agudos	22	
6	4	Pederneiras	19	
7	5	Lençois Paulista	18	
8	6	Tatuí	15	
9	7	Araçoiaba da Serra	10	
10	8	Votorantim	12	
11	9	São Paulo	80	
12	10	Indaiatuba	34	
13	11	Piracicaba	28	
14	12	São Carlos	36	
15	13	Araraquara	20	
16	14	Franca	15	
17	15	Campinas	65	
18	16	Marília	35	
19	17	Assis	23	
20	18	Jundiaí	13	
21	19	São José do Rio Preto	39	

Lista das Cidades | Matriz de Distancias

Fonte: Elaborada pela autora

Para armazenar o percurso percorrido entre todas essas cidades, deve ser informado a matriz de distâncias na página “Matriz de Distancias”, com a numeração especificada na página “Lista das Cidades”, como mostra a Figura 7.

Figura 7 - Página “Matriz de Distancias” da planilha utilizada para testes do software



C/C	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	246	154	228	236	201	59	21	11	104	65	108	202	247	384	89	351	353	93	407
1	246	0	97	22	33	46	199	236	271	332	293	194	154	150	319	281	115	187	336	211
2	154	97	0	78	87	52	106	143	178	238	199	116	154	151	321	224	202	247	228	287
3	228	22	78	0	45	28	182	218	253	314	275	176	165	141	311	241	128	197	304	219
4	236	33	87	45	0	38	191	227	262	323	254	188	123	99	269	231	144	213	268	215
5	201	46	52	28	38	0	155	191	226	287	248	149	157	132	302	214	153	222	277	243
6	59	199	106	182	191	155	0	36	62	143	106	79	186	228	365	128	305	307	132	390
7	21	236	143	218	227	191	36	0	21	122	85	115	222	264	407	107	342	343	111	426
8	11	271	178	253	262	226	62	21	0	105	67	109	204	247	386	90	378	379	94	409
9	104	332	238	314	323	287	143	122	105	0	103	157	234	278	397	93	435	438	56	440
10	65	293	199	275	254	248	106	85	67	103	0	86	163	209	328	29	398	400	51	371
11	108	194	116	176	188	149	79	115	109	157	86	0	99	142	288	70	300	374	105	304
12	202	154	154	165	123	157	186	222	204	234	163	99	0	45	186	142	264	333	179	207
13	247	150	151	141	99	132	228	264	247	278	209	142	45	0	175	185	233	308	221	167
14	384	319	321	311	269	302	365	407	386	397	328	288	186	175	0	308	358	429	345	224
15	89	281	224	241	231	214	128	107	90	93	29	70	142	185	308	0	373	423	39	349
16	351	115	202	128	144	153	305	342	378	435	398	300	264	233	358	373	0	75	409	194
17	353	187	247	197	213	222	307	343	379	438	400	374	333	308	429	423	75	0	427	267
18	93	336	228	304	268	277	132	111	94	56	51	105	179	221	345	39	409	427	0	385
19	407	211	287	219	215	243	390	426	409	440	371	304	207	167	224	349	194	267	385	0

Fonte: Elaborada pela autora

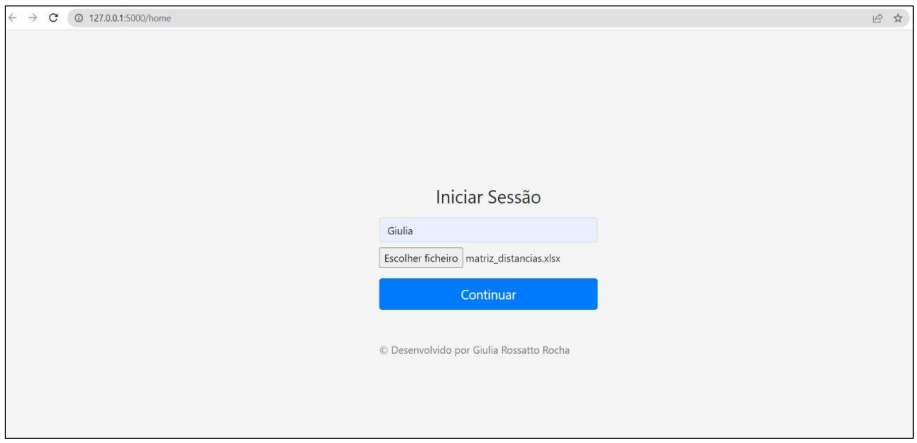
De acordo com as Figuras 6 e 7, por exemplo, a distância entre Sorocaba (cidade com índice 0) e Bauru (cidade com índice 1) é de 246 km. Ressalta-se que tais dados foram preenchidos manualmente de acordo com os valores aproximados obtidos pelo Google Maps, em quilômetros, com pontos aleatórios entre quaisquer duas cidades.

A utilização de dados já processados em detrimento de dados gerados durante a execução do algoritmo deve-se ao fato de que algumas interfaces de programação de aplicação (APIs - *Application Programming Interfaces*, do inglês) demandam um tempo significativo para obter respostas de requisições para cálculo de distâncias entre duas cidades. Esse tempo incrementa drasticamente no tempo de execução final do AG e, consequentemente, em um atraso de resposta para o usuário.

3.1.2 Telas e rotas da aplicação

Uma vez delimitada a planilha contendo informações sobre as cidades, suas cargas e sua respectiva matriz de distâncias, o usuário está apto a utilizar a aplicação. Primeiramente, o usuário é direcionado para a página inicial apresentada na Figura 8. Nela, o usuário deverá inserir um nome ou apelido para registrar a sessão e a planilha elaborada.

Figura 8 - Página inicial do software para registro de sessão

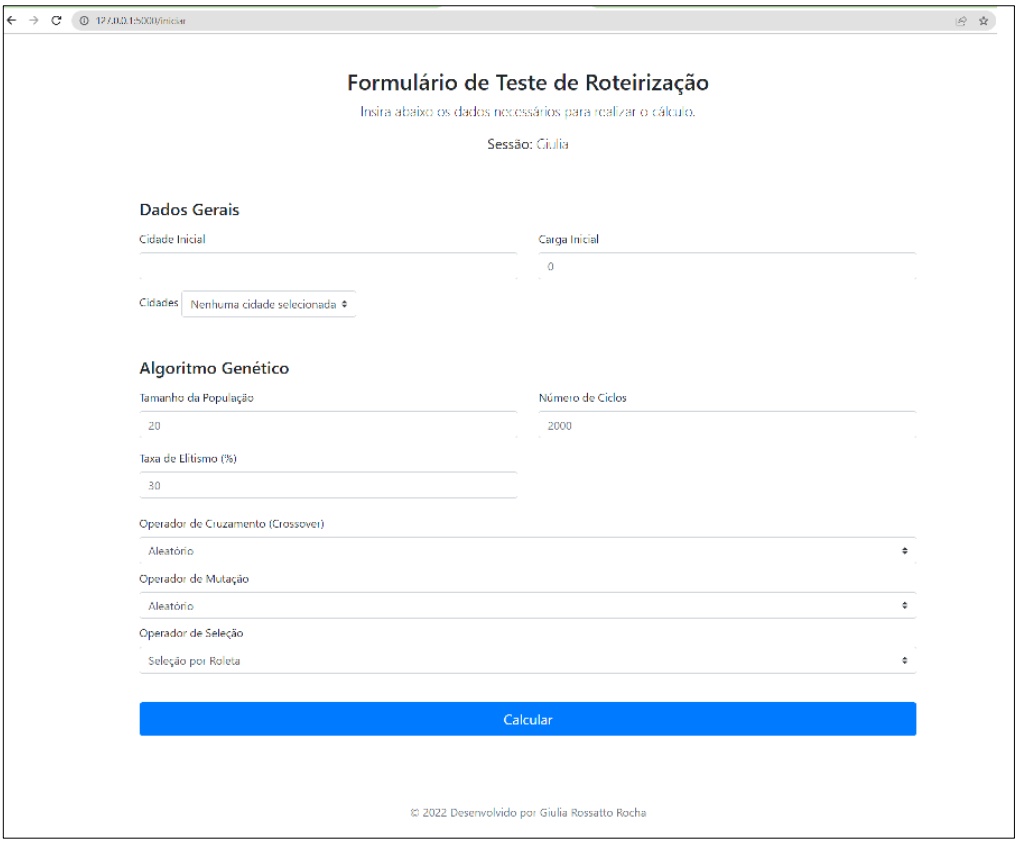


Fonte: Elaborada pela autora

O software é responsável por obter todas as cidades inseridas na planilha e apresentá-las ao usuário na tela seguinte, possibilitando a escolha das cidades que ele gostaria de realizar a roteirização.

A página apresentada na Figura 9 mostra o formulário de teste no qual o usuário deve inserir parâmetros necessários para o funcionamento do AG.

Figura 9 - Formulário de teste de roteirização do software



Fonte: Elaborada pela autora

Esses parâmetros necessários são:

- Cidade Inicial: cidade em que o veículo inicia seu percurso e que servirá como depósito;
- Carga Inicial: representa a capacidade máxima que o veículo pode assumir;
- Cidades: cidades que irão compor o percurso realizado pelo veículo;
- Tamanho da População: representa o número de indivíduos que compõe a população de cada geração;
- Número de ciclos: representa o número de iterações ou gerações do AG;
- Taxa de elitismo: representa a porcentagem dos melhores indivíduos que serão preservados para a próxima geração;
- Operador de Cruzamento (*Crossover*): método de cruzamento dos cromossomos pais para gerar cromossomos filhos. Possui as seguintes opções: Aleatório, *Crossover* CX, *Crossover* PMX e *Crossover* OX;
- Operador de Mutação: método de mutação aplicado sobre os cromossomos filhos. Possui as seguintes opções: Aleatório, Mutação EM, Mutação SIM, Mutação DM, Mutação ISM, Mutação IVM e Mutação SM; e
- Operador de Seleção: método no qual os cromossomos serão selecionados para passar pelos processos de *crossover* e mutação. Possui as seguintes opções: Seleção por Roleta e Seleção por Torneio.

A Figura 10 mostra que o usuário é capaz de selecionar as cidades informadas na planilha da página inicial (Figura 8). O usuário pode selecionar tanto algumas como todas as cidades apresentadas e deve identificar a cidade (campo “Cidade Inicial”) na qual o veículo iniciará o seu percurso e que servirá como depósito para descarga do veículo durante o trajeto.

É válido destacar que as opções apresentadas para cruzamento, mutação e seleção serão explicadas na seção 3.2.

Figura 10 - Preenchimento dos campos “Cidade Inicial” e “Cidades”

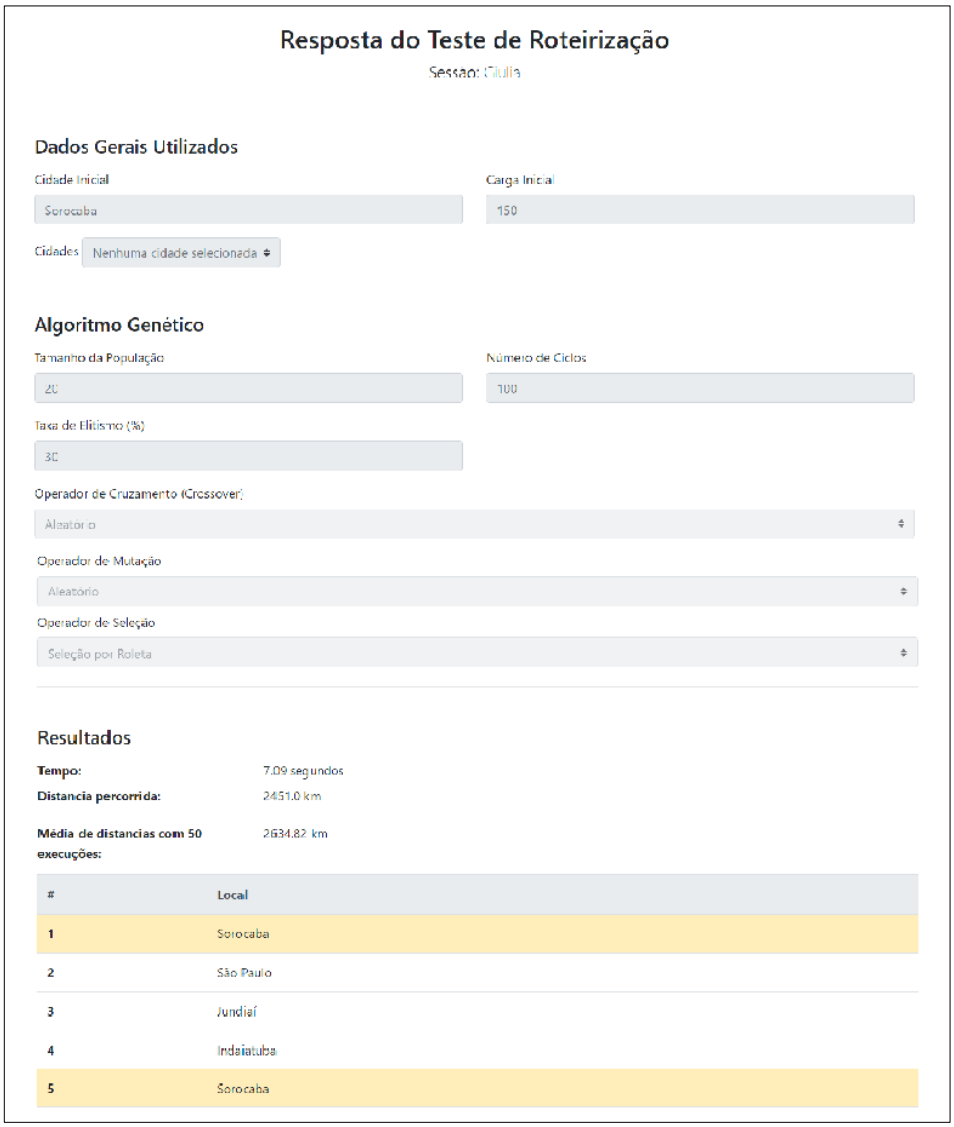
Fonte: Elaborada pela autora

Ao preencher todos os dados, o usuário deve apertar o botão “Calcular” para que o AG inicie seu processamento e direcione a resposta para a página de resultados apresentada na Figura 11.

O exemplo da Figura 11 considerou a cidade de Sorocaba como a inicial e a carga máxima do veículo 150 kg. Para a execução do AG considerou-se uma população de 20 indivíduos, 100 ciclos e taxa de elitismo de 30%. Tanto o operador de cruzamento quanto o de mutação foram considerados do tipo aleatório e o método de seleção utilizado foi por Roleta.

Podem ser realizados quantos testes de roteirização forem necessários. O resultado é apresentado após 50 execuções seguidas do AG com os mesmos parâmetros. O resultado mostra parte das cidades percorridas destacando a volta do veículo à cidade depósito para descarga (em amarelo, como apresenta a Figura 11), o tempo necessário para executar todo o algoritmo, a melhor distância percorrida dentre todas as 50 execuções do AG e a média da distância de todas as execuções.

Figura 11 - Parte do resultado de um teste de roteirização do software



Fonte: Elaborada pela autora

3.2 Algoritmo Genético Aplicado ao Problema

Para o AG aplicado ao problema deste trabalho, a codificação dos genes dos cromossomos ou indivíduos foram realizados por meio de um vetor em que cada posição indica uma cidade a ser visitada, a qual é representada por um inteiro. Para identificar esses inteiros é utilizada a planilha fornecida pelo usuário. Nela é possível obter o nome da cidade e o inteiro que a representa.

Como o PCV e o PRVC envolvem determinar uma sequência de visitação de localidades, é natural que cada gene corresponda a uma localidade a ser visitada e que esses valores não se repetem ao longo do vetor. Caso contrário, o veículo visitará a mesma cidade (sem ser a cidade depósito) mais de uma vez.

Supondo que se deseja visitar as cidades Bauru, Agudos e Botucatu, as quais são representadas, respectivamente, pelos inteiros 1, 2 e 3, por exemplo, o roteiro Agudos-Bauru-Botucatu será representado pelo vetor [2, 1, 3]. Portanto, é de extrema importância que a ordenação entre os valores seja mantida. No entanto, o algoritmo desenvolvido reconhece que os vetores [2, 1, 3], [3, 2, 1] e [1, 3, 2] são iguais, pois a ordem final é a mesma, visto que a cidade inicial é representada por um inteiro e não pela primeira posição do vetor.

A seguir são apresentados os parâmetros e os operadores utilizados para o desenvolvimento do algoritmo. A escolha dos operadores foi baseada em Malaquias (2006), visto que devem preservar a condição de que cada cidade deve ser visitada pelo veículo apenas uma vez. Portanto, alguns operadores usuais não poderiam ser utilizados.

3.2.1 População inicial

A população inicial é gerada aleatoriamente, respeitando as cidades inseridas pelo usuário, o tamanho da população e a regra de que cada cromossomo deve abranger todas as cidades exigidas, sem repetir a mesma ao longo da rota, com exceção da cidade depósito.

3.2.2 Função de custo

O objetivo do problema em questão é encontrar uma solução que minimize esta função de custo dada pela distância total percorrida pelo veículo ao visitar cada cidade da rota. Como adendo, podem existir retornos do veículo à cidade inicial, portanto, a função custo também deve considerar a distância da cidade em que o veículo se encontra até a cidade inicial e, posteriormente, a distância da cidade de origem ao próximo local do roteiro. Isso ocorre caso o veículo não possa atender a demanda da cidade seguinte a ser alcançada de acordo com o roteiro.

3.2.3 Função de aptidão

A função de aptidão consiste em um cálculo aplicado a cada indivíduo, para avaliar sua viabilidade de solução para o problema. Visto que o objetivo é minimizar a função de custo, quanto menor a distância total percorrida pelo veículo em uma rota, melhor o valor de aptidão do indivíduo. Portanto, a função de aptidão é a função de custo, dada pela soma total das distâncias, considerando os retornos realizados à origem.

Convém salientar que nesta implementação, a melhor solução é aquela que possui o menor valor de aptidão, visto que o objetivo é a minimização da função custo e a aptidão é dada por ela.

3.2.4 Seleção

Este trabalho implementou dois modelos de seleção: seleção por roleta, apresentada por Lima et al. (2017) e a seleção por torneio, apresentada por Lacerda e Carvalho (1999).

A seleção por roleta, também apresentada por Lacerda e Carvalho (1999) foi adaptada de acordo com o modelo de Lima et al. (2017) para respeitar os valores dados pela aptidão de cada indivíduo. Como o melhor indivíduo é aquele que possui menor aptidão, deve ser dada alguma forma de privilegiar os cromossomos com menores aptidões. Na seleção por roleta, apresentada por Lima et al. (2017), cada indivíduo recebe "bilhetes" de acordo com a sua aptidão. Indivíduos com maiores aptidões representam indivíduos que percorreram maiores distâncias e, consequentemente, indivíduos com menores aptidões são aqueles que percorreram menores distâncias, ou seja, são as melhores soluções. Portanto, tem-se que:

1. O cromossomo que possui o maior valor de aptidão corresponde à pior solução, recebendo apenas dois bilhetes (F_1 recebe 2 bilhetes, sendo F_1 o valor de aptidão do cromossomo);
2. O cromossomo x válido recebe $F_1 - F_x + 2$ bilhetes;
3. Realiza-se o sorteio de um número entre 1 e o total de bilhetes distribuídos; e
4. O indivíduo que possuir o bilhete sorteado será selecionado para realizar a operação de *crossover*.

Além da seleção por roleta, também foi implementada a seleção por torneio, apresentada por Lacerda e Carvalho (1999), no qual são escolhidos aleatoriamente n cromossomos, de probabilidade iguais, e o melhor entre eles é escolhido para o processo de cruzamento e mutação. Para o presente trabalho, foi adotado $n = 1$ para evitar a convergência prematura como ocorre na seleção por roleta, conforme explicado por Lacerda e Carvalho (1999).

Para ambos os métodos de seleção, em cada ciclo evolutivo são selecionados N indivíduos, ou seja, são realizados N sorteios, em que N corresponde ao tamanho populacional fornecido pelo usuário para passar pelos operadores de cruzamento e mutação. Foi optado por selecionar toda a população visando garantir que novos cromossomos sejam gerados, visto que

o processo de elitismo, que será explicado na subseção 3.2.8, já garante a transferência dos melhores cromossomos da geração atual para a seguinte.

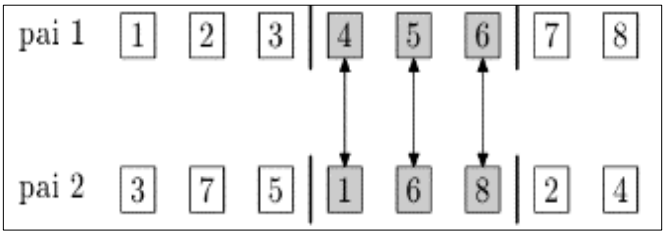
3.2.5 Operadores de cruzamento

Para o cruzamento, os indivíduos resultantes do processo de seleção foram cruzados aos pares, realizando $N/2$ cruzamentos, onde N corresponde ao tamanho populacional indicado pelo usuário. Cada cruzamento originou dois novos indivíduos, portanto N indivíduos.

Foram implementados quatro tipos de cruzamentos: operador de cruzamento de mapeamento parcial (*partially-mapped crossover* – PMX), operador de cruzamento em ciclo (*cycle crossover* – CX), operador de cruzamento ordenado (*order crossover* – OX) apresentado por Malaquias (2006) e cruzamento aleatório, que seleciona um dos tipos de cruzamentos citados anteriormente a cada ciclo de execução do AG.

O operador PMX preserva as informações referente à ordem e posição das rotas dos pais para os cromossomos filhos. Para ambos os pais deve ser selecionado o mesmo ponto de corte, como mostra a Figura 12.

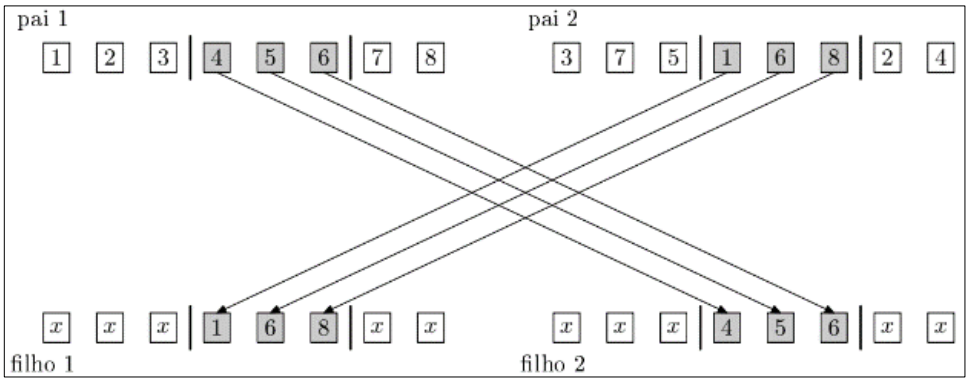
Figura 12 - Cromossomos pais iniciais com o mesmo ponto de corte para o PMX



Fonte: Malaquias (2006)

A sequência obtida entre os pontos de corte é denominada seção de mapeamento. Cada seção de mapeamento é copiada para um filho, tal que a seção de mapeamento do pai 1 é copiada para o filho 2, enquanto a seção do pai 2 é copiada para o filho 1, como mostra a Figura 13.

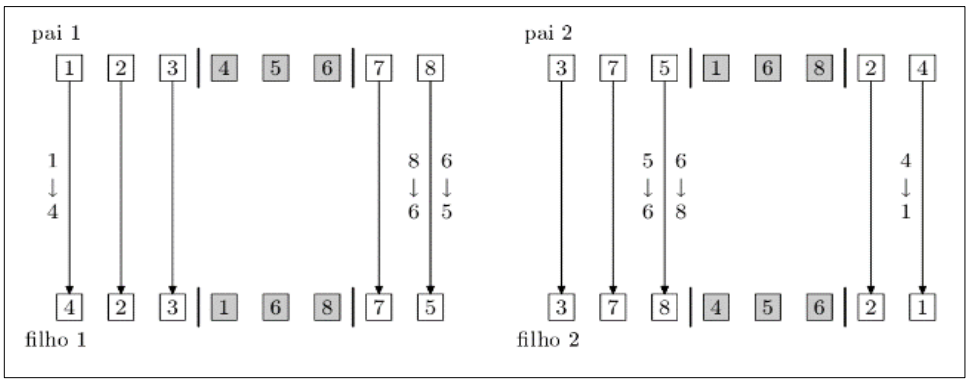
Figura 13 - Troca de seção de mapeamento entre os cromossomos pais para os cromossomos filhos no PMX



Fonte: Malaquias (2006)

Em seguida, o pai 1 precisa dar continuidade realizando a cópia dos seus elementos restantes para o filho 1, por meio de um processo de mapeamento, evitando os valores já existentes que fora preenchido pelo pai 2. No exemplo, o mapeamento inicia no primeiro elemento do pai 1, que seria a cidade 1. Como a cidade 1 já existe no filho 1, ela não pode ser copiada. Como substituto deve ser utilizado o mapeamento definido de acordo com a posição do pai. A cidade 1 que não pode ser copiada corresponde a posição 3 do filho 1 (considerando o primeiro elemento na posição do vetor como índice 0). A posição 3 no pai 1 corresponde à cidade 4. Como o valor 4 ainda não existe no filho 1, o valor é copiado. O processo segue com o mapeamento, e da mesma forma com o pai 2 e o filho 2, como mostra a Figura 14.

Figura 14 - Resultado do PMX

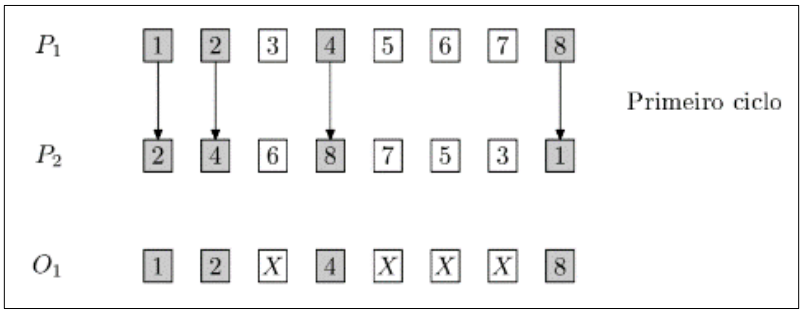


Fonte: Malaquias (2006)

O operador CX utiliza o mapeamento dos elementos de ambos os pais para compor um filho por meio de ciclos. Na Figura 15, dada como exemplo, escolhe-se o primeiro elemento do pai 1 ou do pai 2. Escolhido o primeiro elemento do pai 1 (ou seja, índice 0), deve ser mapeado o valor correspondente ao índice da cidade escolhida do pai 1. Portanto, a cidade 2 do pai 2 corresponde ao índice 0 escolhido para início do algoritmo. Para dar continuidade ao ciclo,

deve-se procurar no pai 1 o índice correspondente a cidade 2. Como a cidade 2 corresponde ao índice 1 no pai 1, um novo valor é escolhido. O ciclo termina quando for encontrado no pai 2 um valor já incluído no pai 1. Assim, são selecionados todos os valores do ciclo do pai 1 para o filho.

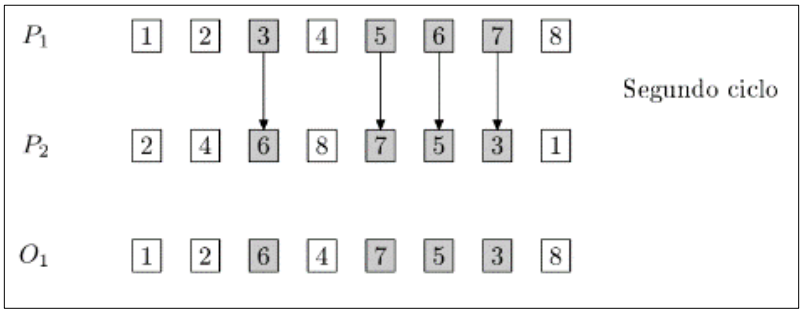
Figura 15 - Primeiro ciclo do CX



Fonte: Malaquias (2006)

Para o próximo ciclo é escolhido o índice no qual o filho ainda não está preenchido, no caso o índice 2. Ao final desse ciclo, os valores a serem copiados para o filho devem vir do cromossomo pai 2, como mostra a Figura 16. Esse processo é alternado até que o filho todo seja preenchido. Para o presente trabalho, o segundo filho é formado iniciando do pai 2, ao invés do pai 1.

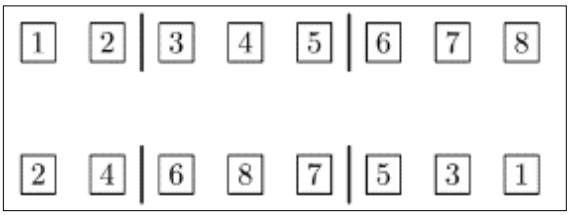
Figura 16 - Segundo ciclo do CX



Fonte: Malaquias (2006)

Por fim, o operador OX prioriza a ordenação das cidades e não suas posições, no qual um filho é formado de acordo com o sub-roteiro de um dos pais, preservando a ordem das cidades. Para exemplificar o algoritmo, suponha dois cromossomos pais, como mostra a Figura 17.

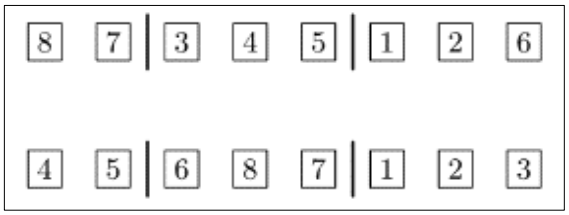
Figura 17 - Cromossomos pais com dois pontos de cortes para o operador OX



Fonte: Malaquias (2006)

Para cada filho são copiados os valores existentes entre os dois pontos de cortes, ou seja, o pai 1 é copiado no filho 1 e o pai 2 no filho 2. Em seguida, os valores restantes do filho 1 são preenchidos a partir da sequência do segundo corte do pai 2, omitindo as cidades já presentes. O processo se repete para o filho 2 com o pai 1. O resultado é apresentado na Figura 18.

Figura 18 - Exemplos de filhos gerados pelo operador OX



Fonte: Malaquias (2006)

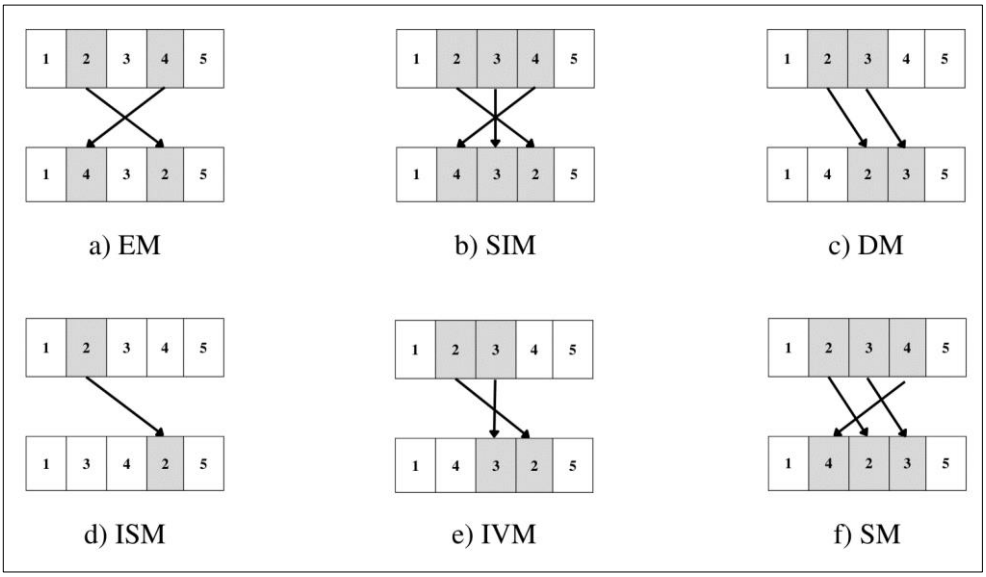
3.2.6 Operadores de mutação

Os operadores de mutação são responsáveis por modificar os indivíduos filhos e permitir novas combinações de soluções. Foram utilizados os seguintes operadores de mutação, apresentados por Malaquias (2006):

1. Mutação por troca (*exchange mutation* – EM): representado na ilustração (a) da Figura 19, seleciona aleatoriamente duas cidades e troca suas posições;
2. Mutação por inversão simples (*simple inversion mutation* – SIM): representado na ilustração (b) da Figura 19, seleciona aleatoriamente um trecho do cromossomo e inverte a ordem das cidades;
3. Mutação por deslocamento (*displacement mutation* – DM): representado na ilustração (c) da Figura 19, seleciona aleatoriamente um trecho do cromossomo, remove esse trecho do cromossomo e o insere em outro ponto de corte;
4. Mutação por inserção (*insertion mutation* - ISM): representado na Figura 19(d), seleciona aleatoriamente uma cidade do cromossomo e a transfere para outra posição;

5. Mutação por inversão (*inversion mutation* - IVM): representado na Figura 19(e), seleciona aleatoriamente um trecho do cromossomo, remove-o do cromossomo, inverte-o e insere em outro ponto de corte;
6. Mutação por embaralhamento (*scramble mutation* - SM): representado na Figura 19(f), seleciona aleatoriamente um trecho do cromossomo e o embaralha; e
7. Mutação aleatória: seleciona uma dentre seis as mutações apresentadas para cada ciclo de execução do AG.

Figura 19 - Operadores de mutações implementados



Fonte: Elaborada pela autora

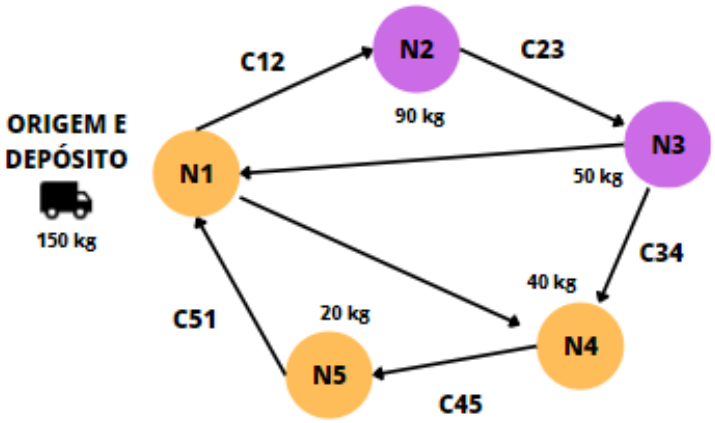
3.2.7 Restrição de carga

Para lidar com a limitação de carga do veículo e das demandas de cada cidade foi estipulado que, caso o veículo esteja na cidade X e deseja ir para a cidade Y, o algoritmo verifica se o veículo possui capacidade suficiente para atender a demanda exigida pela cidade Y. Caso não possua, ele sairá da cidade X em direção à cidade inicial, descarregará a carga atual e retomará o percurso partindo para a cidade Y, sendo calculada a distância entre a cidade X e a origem e entre a origem e a cidade Y.

A Figura 20 exemplifica o modelo de tratamento da carga do veículo adotado no software. De acordo com a Figura 20, o veículo partirá do nó de origem N1 em direção ao nó N2. Em N2 ele receberá 90kg de carga, restando, assim, 60kg para atender o próximo nó (cliente). Em N3 o veículo ainda consegue receber a demanda, visto que ocupará mais 50kg. Dessa forma, o veículo possui 140kg de carga ocupada e 10kg restantes. Como o veículo só

possui 10kg restantes, ele não é capaz de atender o nó N4. Assim, é necessário retornar à cidade depósito (N1) para realizar seu descarregamento e, posteriormente, prosseguir sua rota a partir de N4. A partir do nó N4, ele é capaz de atender todas as cidades restantes, visto que somarão o total de 60kg, até chegar ao seu destino em N1.

Figura 20 - Exemplificação em grafo do modelo de tratamento da restrição de carga do veículo adotado no software



Fonte: Elaborada pela autora

O veículo nem sempre aproveitará sua capacidade máxima, no entanto, evitará frustrações de clientes que poderiam ser geradas caso o veículo preenchesse sua capacidade máxima em determinada localidade, mas a demanda da cidade atual em que o veículo se encontra não tivesse sido totalmente atendida, precisando retornar a ela novamente após sua descarga na cidade inicial. Dessa forma, cada cidade será totalmente atendida durante a passagem do veículo.

3.2.8 Elitismo

Ao final do processo de cruzamento e mutação, tem-se N indivíduos selecionados para compor a nova população que fará parte do próximo ciclo evolutivo. No entanto, para garantir a seleção dos melhores indivíduos é realizado o processo de elitismo, no qual é removido uma porcentagem desses N indivíduos que serão substituídos pelos melhores indivíduos da população que o originou. Essa porcentagem é fornecida pelo usuário. Logo, quanto maior a porcentagem, maior o número de indivíduos substituídos para compor a população da geração.

3.3 Diagrama de Classes do Algoritmo Genético Implementado

Para a modelagem da solução, utilizou-se a linguagem de modelagem UML criando um diagrama de classes para organizar a implementação realizada do AG e facilitar o entendimento do fluxo de funcionamento do código. O diagrama apresenta-se na Figura 21 e é composto de quatro classes: *AlgoritmoGenetico, Ambiente, Populacao e Cromossomo*.

Figura 21 - Diagrama de classes UML do algoritmo implementado

Fonte: Elaborada pela autora

A classe *AlgoritmoGenetico* corresponde a classe principal que rege o funcionamento das demais. Ela é responsável por iniciar o AG e coordenar todos os passos principais para o funcionamento do AG, como iniciar a população, realizar a iteração de cada geração, iniciar o processo de seleção, cruzamento, mutação e verificar a condição de parada. Nela há alguns atributos, dentre eles tem-se:

- *crossover* e *mutacao*: representam o *crossover* e a mutação, respectivamente, que serão utilizados para todo o AG;

- *opcoes_crossover* e *opcoes_mutacao*: representam todas as opções possíveis de *crossover* e mutação que foram implementadas no código;
- *crossover_aleatorio* e *mutacao_aleatoria*: representam se o usuário escolheu crossover e/ou mutação aleatória. Caso seja verdadeiro, a cada ciclo é sorteado um valor de *opcoes_crossover* e *opcoes_mutacao*; e
- *cidades*, *cidade_inicial*, *mastriz_dist*, *restricoes_carga*, *carga_veiculo*, *tamanho_populacao*, *numero_ciclos*, *taxa_elitismo*: valores obtidos pelos dados fornecidos pelo usuário.

A classe descrita anteriormente necessita de uma classe *Ambiente* que será necessária para realizar todos os processos de *crossover*, mutação, elitismo e seleção já descritos anteriormente. Seus atributos são derivados da classe *AlgoritmoGenetico* e não é possível sua existência sem ele.

Também é necessária a existência da classe *Populacao* que é instanciada a cada ciclo do AG para geração da população. Nela são descritos métodos de alteração e modificação da população e seus indivíduos, como criar a população inicial aleatória com o método *cria_populacao_inicial()*. Ao instanciar um objeto de *Populacao*, deve ser passado uma lista de valores que representarão todos os indivíduos da população instanciada para o método *atribui_individuos()*. Para cada indivíduo passado será chamado o método *cria_individuo()* demandando o uso de *adiciona_individuo()* que instancia a classe *Cromossomo*, atribui seu valor aos genes da classe e o adiciona ao atributo *individuos* da classe *Populacao*.

Além dos métodos descritos, outros métodos importantes na classe *Populacao* são *elimina_individuo()* utilizado durante o processo de elitismo para remover os piores indivíduos da população e o método *distribui_bilhetes()* utilizado para a seleção por roleta.

Por fim, a classe *Populacao* exige a existência da classe *Cromossomo* que constitui a formação de cada um de seus indivíduos. Seus principais atributos são *genes* que representa a codificação do vetor para o problema e *aptidao* que é dada pela função custo já descrita anteriormente. Além disso, atributos e métodos referentes aos bilhetes são utilizados no método de seleção por roleta na classe *Ambiente*.

4 RESULTADOS

Para apresentação dos resultados foram escolhidas aleatoriamente 20 cidades para elaboração de uma planilha com uma matriz 20x20 contendo as distâncias calculadas entre as cidades, a partir de pontos de partidas aleatórios entre cada uma. A planilha elaborada para os testes pode ser verificada na Figura 7 apresentada na seção 3.1.1. A distância entre duas cidades, em quilômetros (km), foi calculada utilizando a ferramenta Google Maps.

Para cada cidade atribui-se um valor fictício, representando a demanda de carga exigida em cada uma delas, respeitando seu tamanho populacional. Desta forma as cidades mais populosas possuem maiores demandas de carga, enquanto cidades menos populosas exigem menos. As cargas utilizadas para os testes estão dispostas na Figura 6 (seção 3.1.1).

As Tabelas e gráficos deste capítulo utilizaram todas as 20 cidades, tendo a cidade de Sorocaba como cidade inicial e depósito. Além disso, os parâmetros utilizados no AG foram: 50 indivíduos por população, 150 unidades de limite de carga do veículo, um total de 1.000 ciclos evolutivos e uma taxa de elitismo de 20%.

Foram realizadas 56 comparações, nas quais foram combinadas um tipo de seleção (por roleta ou torneio), um tipo de cruzamento (aleatório, CX, PMX ou OX) e um tipo de cruzamento (aleatório, EM, SIM, DM, ISM, IVM ou SM) para cada combinação. O algoritmo foi executado 10 vezes a fim de obter o tempo total com essas dez execuções, a melhor distância obtida dentre as dez e a média das distâncias considerando todas as execuções para cada arranjo. Além disso, comparou-se a convergência dos resultados de cada combinação e a influência do elitismo e do limite de carga dos veículos nos resultados.

4.1 Análise de Tempo de Cada Combinação

O tempo é um parâmetro muito importante para medir a aplicabilidade do AG. Como ele surge para solucionar o problema de tempo para problemas NP-árduos como o PCV e PRVC, produzindo uma solução viável em um tempo computacional válido, obter um retorno relativamente rápido torna-se importante para medir a efetividade do algoritmo. Além disso, retornos rápidos consistem em clientes que exigem rápidas soluções mais satisfeitos.

Dessa forma, a Tabela 1 e a Tabela 2 apresentam os intervalos de tempo, medidos em segundos, para obter a melhor solução em 10 execuções seguidas do AG desenvolvido com seleção por roleta e seleção por torneio, respectivamente.

Tabela 1 - Tempo (em segundos) obtido nas combinações entre operadores de cruzamento e mutação, após 10 execuções do algoritmo genético, utilizando seleção por roleta

Seleção por Roleta				
	Cruzamento aleatório	**CX**	**PMX**	**OX**
Mutação aleatória	36,71	**89,47**	22,63	22,37
EM	26,99	34,52	21,87	19,74
SIM	24,36	81,93	23,44	21,02
DM	24,66	74,06	23,82	21,85
ISM	24,15	82,48	21,50	**19,45**
IVM	25,54	40,31	24,07	21,84
SM	26,72	75,23	24,15	23,27

Fonte: Elaborado pela autora

Tabela 2 - Tempo (em segundos) obtido nas combinações entre operadores de cruzamento e mutação, após 10 execuções do algoritmo genético, utilizando seleção por torneio

Seleção por Torneio				
	Cruzamento aleatório	**CX**	**PMX**	**OX**
Mutação aleatória	69,38	43,50	23,22	20,77
EM	25,63	**81,11**	22,29	20,88
SIM	23,40	47,90	22,92	20,57
DM	22,76	38,19	22,07	20,89
ISM	23,10	60,21	21,28	**19,59**
IVM	23,99	33,79	24,01	21,14
SM	26,90	37,82	24,04	22,39

Fonte: Elaborado pela autora

Ao analisar o melhor tempo, Tabelas 1 e 2 com destaque em verde, observa-se que a combinação OX e ISM apresentou melhores resultados, independente da seleção utilizada, obtendo em torno de 19 segundos. No entanto, a utilização do cruzamento CX foi a pior (destaque em vermelho nas Tabelas 1 e 2) em quesito de desempenho quando combinada com a mutação aleatória e a seleção por roleta, como mostra a Tabela 1, e quando combinada com a mutação EM e a seleção por torneio (Tabela 2), com 89,47 e 81,11 segundos, respectivamente.

Além disso, é válido destacar os desempenhos obtidos dos cruzamentos PMX e OX, em ambas as Tabelas, com qualquer mutação e seleção utilizada. O cruzamento aleatório também apresentou bons resultados, com exceção da sua combinação com a mutação aleatória nas duas Tabelas. Ademais, é possível notar que o cruzamento CX apresentou o pior desempenho de tempo em relação aos outros tipos de cruzamento, em ambos os casos. Portanto, a escolha da seleção não influenciou no desempenho de tempo.

4.2 Análise da Melhor Solução

Nas Tabelas 3 e 4, apresentadas na sequência, é possível verificar a melhor distância obtida, em quilômetros, pelo AG após 10 execuções das combinações de cruzamento e mutação utilizando seleção por roleta e por torneio, respectivamente. Pelos valores obtidos em ambas as tabelas, o melhor resultado em todas as execuções foi de 2.451 km, ou seja, o melhor percurso obtido que atinge todas as 20 cidades marca o valor de 2.451 km.

Na Tabela 3 verifica-se que todos os cruzamentos atingem, pelo menos com alguma combinação de mutação, o melhor valor gerado pelo AG. Isso também se aplica a todas as mutações. Algumas exceções como os cruzamentos CX e OX apresentaram quantidade maior de valores superiores a 2.451 km quando combinados com algumas mutações, mas nada significativo para ser discutido.

Tabela 3 - Melhor distância (em km) obtida nas combinações entre operadores de cruzamento e mutação, após 10 execuções do algoritmo genético, utilizando seleção por roleta

Seleção por Roleta				
	Cruzamento aleatório	**CX**	**PMX**	**OX**
Mutação aleatória	2.451,0	2.451,0	2.451,0	2.479,0
EM	2.451,0	2.479,0	2.479,0	2.451,0
SIM	2.493,0	2.467,0	2.451,0	2.451,0
DM	2.451,0	2.451,0	2.451,0	2.479,0
ISM	2.451,0	2.451,0	2.451,0	2.479,0
IVM	2.451,0	2.467,0	2.451,0	2.451,0
SM	2.479,0	2.451,0	2.479,0	2.479,0

Fonte: Elaborado pela autora

Tabela 4 - Melhor distância (em km) obtida em 10 execuções do algoritmo genético utilizando vários cruzamentos e o método de seleção por torneio

Seleção por Torneio				
	Cruzamento aleatório	**CX**	**PMX**	**OX**
Mutação aleatória	2.451,0	2.479,0	2.451,0	2.465,0
EM	2.451,0	2.479,0	2.451,0	2.465,0
SIM	2.451,0	2.451,0	2.451,0	2.451,0
DM	2.451,0	2.451,0	2.451,0	2.451,0
ISM	2.479,0	2.451,0	2.451,0	2.451,0
IVM	2.451,0	2.451,0	2.451,0	2.452,0
SM	2.451,0	2.486,0	2.479,0	2.451,0

Fonte: Elaborado pela autora

O mesmo ocorre na Tabela 4 em que todos os cruzamentos atingem, pelo menos com alguma combinação de mutação, o melhor valor gerado pelo AG descrito no início do capítulo. Isso também se aplica a todas as mutações. Ressalta-se que o operador PMX e o operador SIM, atingiram a marca do melhor valor gerado pelo AG em todas as suas combinações.

As Tabelas anteriores mostram que, para 10 execuções, pelo menos uma foi capaz de gerar os valores apresentados. Além disso, é possível notar que a seleção por torneio apresentou melhores resultados que a seleção por roleta nesse aspecto.

No entanto, esse não é o melhor parâmetro para ser apresentado para apontar a melhor combinação, visto que se trata de um algoritmo iterativo e probabilístico que a cada execução pode gerar um resultado diferente. Por exemplo, se de 10 execuções, 9 forem ruins e 1 boa, o algoritmo apresentará essa melhor solução.

4.3 Análise da Solução Média

Nas Tabelas 5 e 6, dadas a seguir, é possível verificar a média das distâncias obtidas, em quilômetros, pelo AG após 10 execuções das combinações de cruzamento e mutação utilizando seleção por roleta e por torneio, respectivamente.

Tabela 5 - Distância média (em km) obtida em 10 execuções do algoritmo genético utilizando seleção por roleta

Seleção por Roleta				
	Cruzamento aleatório	**CX**	**PMX**	**OX**
Mutação aleatória	2.513,4	**2.487,2**	2.498,0	2.510,0
EM	2.539,6	2.546,4	2.542,0	2.547,9
SIM	2.536,2	2.557,8	2.506,1	**2.489,2**
DM	2.502,5	2.509,6	2.513,4	2.523,0
ISM	2.541,7	2.556,5	2.517,6	2.568,1
IVM	2.499,6	2.529,3	2.500,3	2.501,2
SM	**2.635,6**	2.561,7	**2.573,0**	**2.575,2**

Fonte: Elaborado pela autora

Na Tabela 5 verifica-se que a combinação CX e mutação aleatória apresentou o melhor resultado, enquanto a combinação do cruzamento aleatório com SM gerou a pior solução. Além disso, é possível notar que a combinação SM foi a mutação que gerou os piores resultados dentre todas as mutações utilizadas.

Tabela 6 - Distância média (em km) obtida em 10 execuções do algoritmo genético utilizando seleção por torneio

Seleção por Torneio				
	Cruzamento aleatório	**CX**	**PMX**	**OX**
Mutação aleatória	2.513,4	2.546,8	2.496,1	2.523,9
EM	2.561,5	**2.621,4**	2.502,4	**2.499,2**
SIM	2.518,8	2.507,9	2.505,6	2.520,0
DM	2.541,5	2.559,0	**2.474,6**	2.581,3
ISM	2.544,3	2.523,4	2.561,8	2.502,3
IVM	2.542,7	2.545,9	2.525,4	2.538,9
SM	2.583,8	**2.686,0**	**2.654,2**	2.578,5

Fonte: Elaborado pela autora

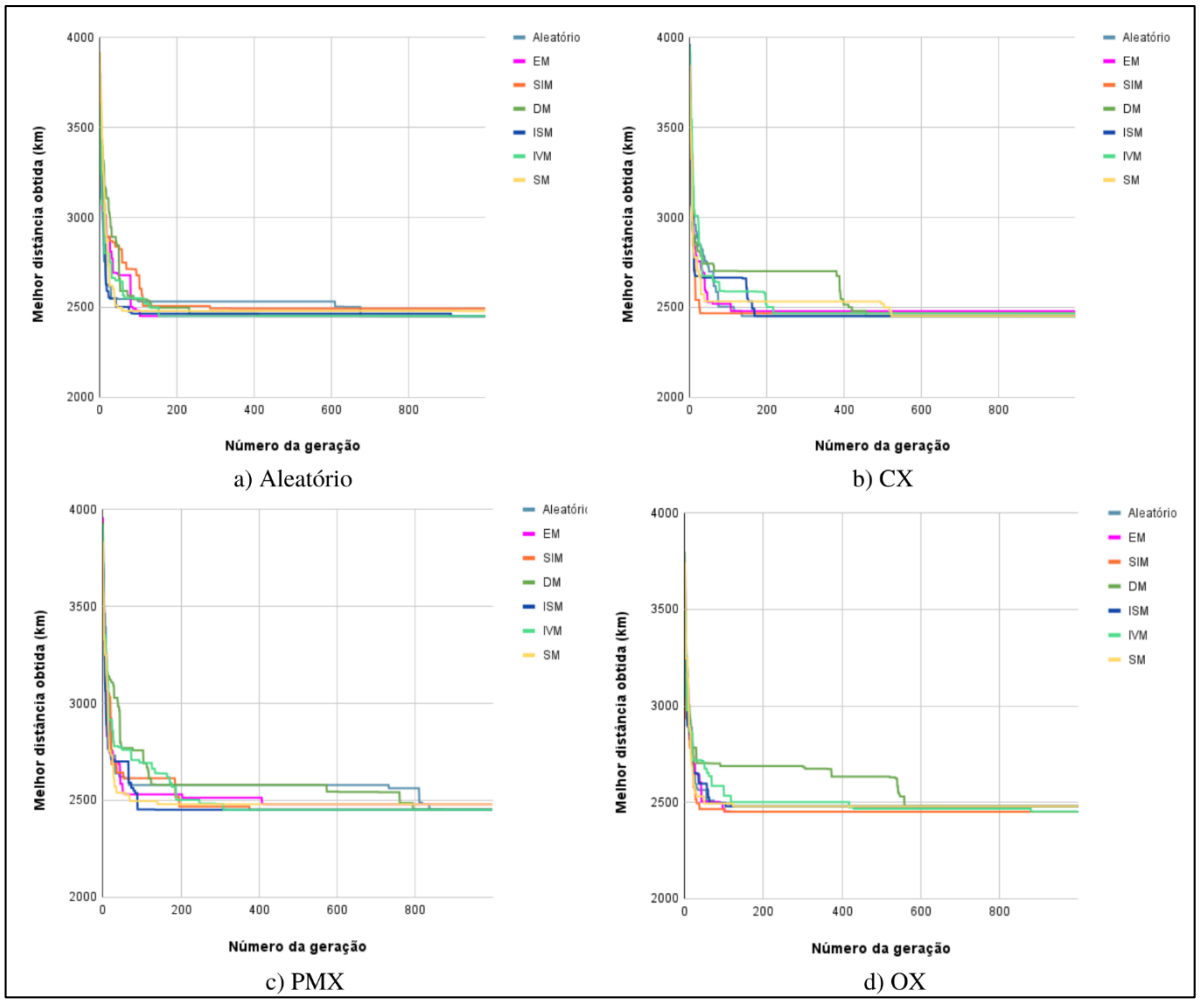
Na Tabela 6 verifica-se que a combinação PMX com DM apresentou o melhor resultado, enquanto a combinação CX com SM gerou a pior solução. Além disso, é válido destacar que tanto o cruzamento CX quanto a mutação SM apresentaram os piores resultados. Por outro lado, a melhor combinação apresentada foi a do cruzamento PMX com a mutação DM.

Observando as Tabelas 5 e 6, é possível afirmar que o operador de mutação SM foi o operador que gerou os piores resultados, independente do cruzamento e da seleção. Por outro lado, o operador de cruzamento OX obteve bons resultados em ambos os tipos de seleção.

4.4 Convergência do Algoritmo

Outro ponto a ser observado é a convergência do AG conforme a alteração dos operadores genéticos. A convergência representa o quão rápido o algoritmo encontra a melhor solução naquele período de execução. Portanto, para a plotagem de cada gráfico foi utilizado a melhor distância obtida a cada geração de cada combinação, considerando como solução final de cada uma as soluções apresentadas nas Tabelas 1 e 2.

Figura 22 - Melhor distância por número de geração para a seleção por roleta considerando cada combinação de cruzamento e mutação



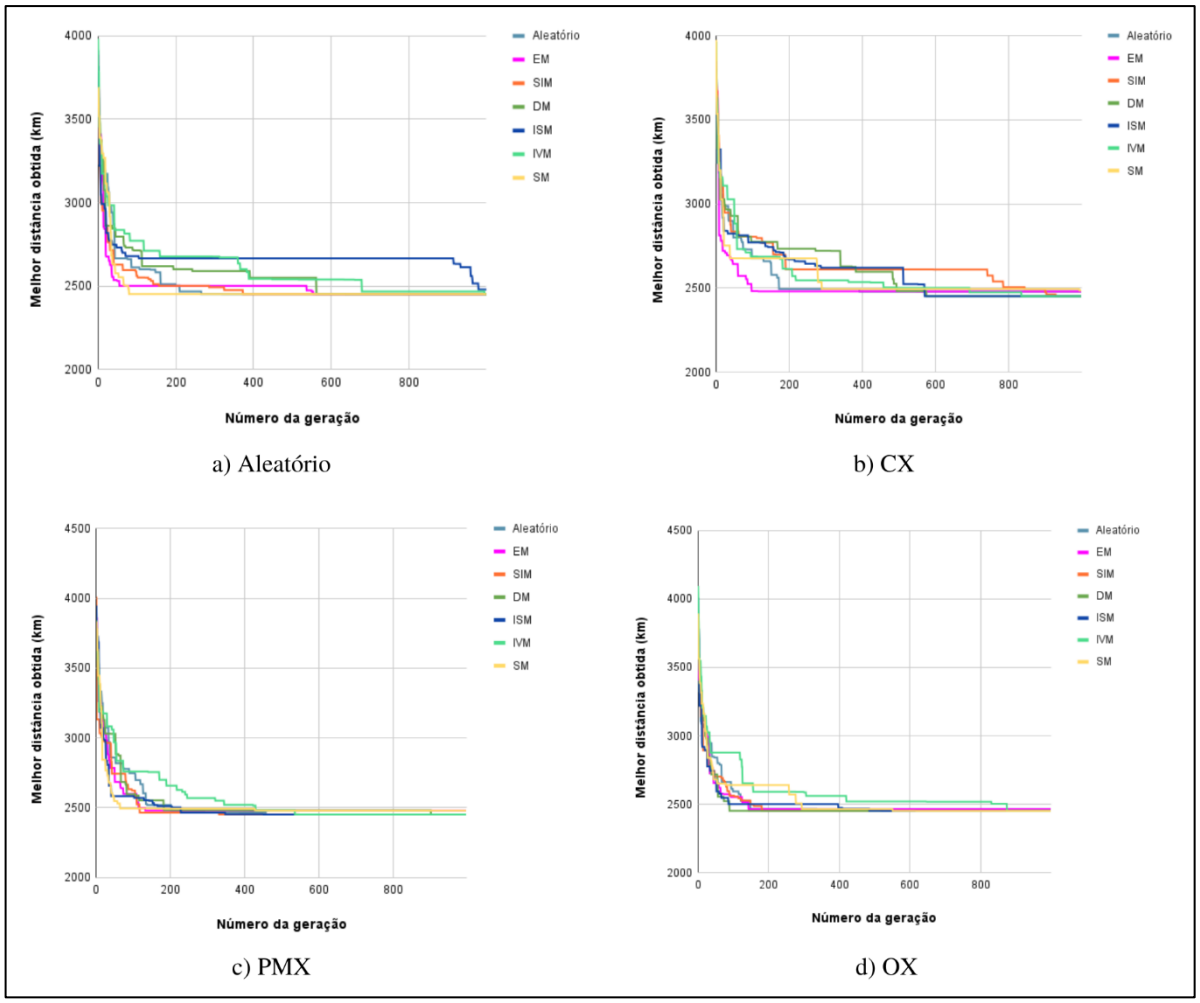
Fonte: Elaborado pela autora

Na Figura 22 verifica-se que todas as mutações combinadas com os cruzamentos apresentaram uma convergência rápida, principalmente os operadores EM e ISM. O operador que apresentou certa divergência foi o operador de mutação DM que obteve altos valores ainda na metade da execução do algoritmo, convergindo nas últimas gerações para a solução final.

Por fim, o gráfico apresentado em Figura 22(a) pelo operador de cruzamento aleatório representa a melhor convergência entre os demais.

Pela Figura 23 pode-se verificar uma convergência mais lenta com os operadores de cruzamento aleatório e CX. Mais uma vez, o operador de mutação EM se destaca pela sua rápida convergência. Por fim, os gráficos apresentados em (c) e (d) pelos operadores de cruzamento PMX e OX, respectivamente, apresentam as melhores convergências entre os demais.

Figura 23 - Melhor distância por número de geração para a seleção por torneio para cada combinação de cruzamento e mutação

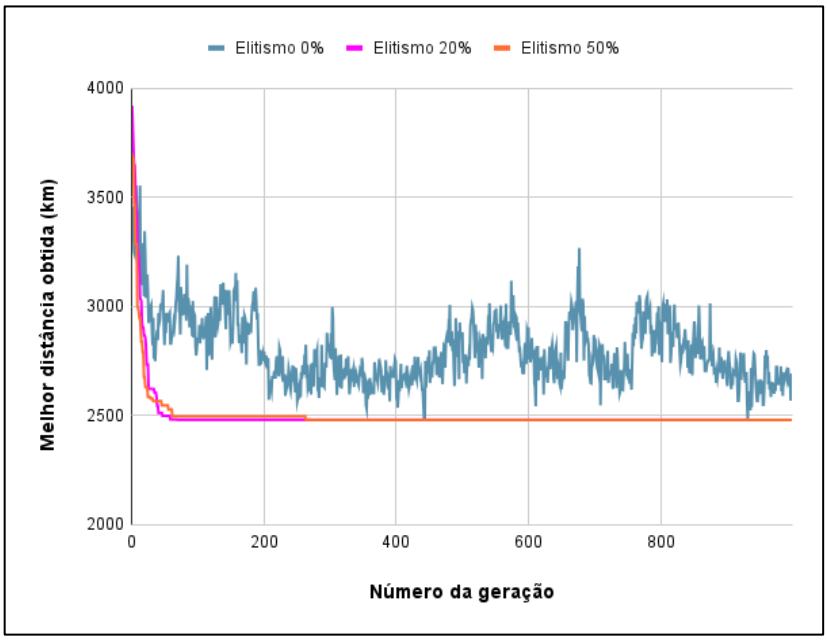


Fonte: Elaborado pela autora

4.5 Influência da Utilização de Elitismo

Para verificar a atuação da taxa de elitismo sobre as populações em cada ciclo, foi utilizado a seleção por roleta, com os operadores CX e mutação aleatória que representa a melhor média de distâncias obtida entre as combinações da Tabela 5. Foi utilizado a melhor distância obtida com a taxa de elitismo em 0%, 20% e 50% para obtenção do gráfico na Figura 24.

Figura 24 - Melhor distância obtida (km) por número de geração para representar a influência do elitismo



Fonte: Elaborado pela autora

De acordo com a Figura 24, é nítido verificar que o elitismo garante que as melhores soluções sejam transferidas para a próxima geração. Sem ele, a influência da seleção, probabilística, e dos operadores de cruzamento e mutação sozinhos fazem com que não seja possível garantir a existência do melhor cromossomo e, portanto, podem gerar indivíduos indesejados para a solução da geração, como é o caso do intervalo de 600 a 800 gerações.

Embora a execução sem elitismo esteja obtendo uma certa convergência a partir da geração 800, a busca pela melhor solução pode se perpetuar por um número maior de gerações quando comparado às execuções com elitismo em 20% e em 50%. No entanto, entre 20% e 50% de taxa de elitismo, não há diferenças significativas.

4.6 Influência da Carga do Veículo

Além dos testes apresentados foi verificado a influência do aumento e diminuição de carga para o veículo no resultado. O aumento da capacidade de carga do veículo gerou menores valores de distâncias ao final da execução, enquanto a diminuição do limite do veículo acarretou um aumento significativo para a distância final obtida. A escolha da cidade inicial influencia na solução final, pois implica na variação da distância percorrida do veículo a ela para as descargas.

5 CONCLUSÃO

De acordo com as Tabelas 1 e 2, em que são apresentados os desempenhos de tempo de cada combinação de operadores, é possível afirmar que a utilização do operador de cruzamento CX não é aconselhável quando se refere a desempenho de tempo. No entanto, os operadores de cruzamento PMX e OX apresentaram bons resultados em suas execuções, podendo ser apresentadas como soluções principais para situações de exigência de prazos curtos para respostas, principalmente quando combinados à mutação ISM.

As Tabelas 3 e 4, em que são apresentadas as melhores soluções do algoritmo para cada combinação, garantem que esta não é uma boa forma de avaliação do algoritmo visto que é um algoritmo probabilístico e que o resultado de uma execução pode não definir o comportamento frequente dos operadores. Portanto, quase todas as combinações apresentaram bons resultados ou semelhantes após dez execuções.

As Tabelas 5 e 6 permitiram verificar as melhores combinações de operadores para o uso de AG aplicado aos problemas PCV e PRVC, pois os valores obtidos representam a média das melhores soluções encontradas após dez execuções. Assim, merecem destaque aqueles que possuíram as menores médias, pois possuem chances maiores de garantir bons resultados a cada execução.

Após essas análises pode-se dizer que a melhor combinação a ser utilizada é a seleção por torneio, cruzamento PMX e mutação DM. Além disso, o operador OX também pode ser combinado com alguns operadores como o de mutação EM com seleção por torneio e o de mutação SIM com seleção por roleta. E por fim, não é aconselhável o uso do operador de mutação SM visto que apresentou os piores resultados durante as dez execuções do algoritmo com diferentes combinações de cruzamento.

Quanto à convergência das combinações estabelecidas, deve-se priorizar a utilização da seleção por torneio com os operadores de cruzamento PMX e OX visto que obtiveram uma solução adequada logo nos primeiros 300 ciclos. Quanto maior a convergência, menor o tempo gasto na busca por soluções. Para trabalhos futuros, é válido verificar se essa constatação se mantém para grande número de localidades.

Além disso, os gráficos apontados nas Figuras 22 e 23 confirmam que não é necessária a implementação de um novo critério de parada que interrompa a execução no caso de valores estagnados por muitas gerações, pois algumas combinações convergiram nos ciclos finais do algoritmo, como é o caso da mutação ISM da Figura 23(a).

Ademais, é válido ressaltar que a utilização do elitismo também previne a execução do AG por muitos minutos para obtenção de um bom resultado. Selecionando os melhores indivíduos para serem transferidos para a próxima geração, bons resultados são obtidos de forma rápida nas primeiras gerações.

No que se refere a capacidade de carga do veículo foi confirmado que o aumento do limite do veículo gera melhores resultados pois são necessárias menos retornos do veículo à cidade de depósito para descarga e, portanto, economia na distância total de deslocamento.

Dessa forma, é possível garantir que a aplicação de AGs para problemas de roteirização e carregamento de veículos é viável visto que estes permitem diferentes modificações para adaptação do problema com soluções aceitáveis em um limite de tempo razoável.

Para futuros trabalhos destacam-se a implementação de outras heurísticas combinadas aos AGs para exploração de novos espaços de busca do algoritmo, outra abordagem com relação à restrição de carga do veículo e a utilização de mais de um veículo para a roteirização. Ainda com relação à trabalhos futuros, pode ser verificada a viabilidade de utilização de uma API de geolocalização com alta capacidade de processamento para geração das distâncias entre as localidades em tempo de execução do AG. Ademais, é válido a implementação de explicações sobre a utilidade e modo de preenchimento de cada campo nas páginas web do software para melhor navegação do usuário.

REFERÊNCIAS

BENEVIDES, P. F. **Aplicação de Heurísticas e Metaheurísticas para o Problema do Caixeiro Viajante em um Problema Real de Roteirização de Veículos**. 2011. 157 f. Dissertação (Mestrado em Ciências) - Universidade Federal do Paraná, Curitiba, 2011. Disponível em: http://paginapessoal.utfpr.edu.br/paulabenevides/publicacoes/publicacoes/PaulaBenevides_Dissertao.pdf. Acesso em: 22 dez. 2022.

CHRISTOPHER, M. **Logística e gerenciamento da cadeia de suprimentos**. Tradução Priscilla Rodrigues da Silva e Lopes. 4. ed. São Paulo: Cengage Learning, 2018. 392 p.

FRANCISCO, R. H. C.; GILBERTO, T. M. J. Pesquisa Operacional Aplicada na Área de Logística de Transporte Rodoviário em uma Transportadora do Município de Franca/SP. **Creare-Revista das Engenharias**, v. 1, n. 1, 2018. Disponível em: http://periodicos.unifacef.com.br/index.php/creare/article/view/1643/1166. Acesso em: 10 dez. 2022.

GOLDBARG, E.; GOLDBARG, M. C.; LUNA, H. P. L. **Otimização Combinatória e Meta-heurísticas - Algoritmos e Aplicações**. Rio de Janeiro: GEN LTC, 2021. E-book. 409 p.

GOLDBARG, M. C; LUNA, H. P. L. **Otimização combinatória e programação linear: modelos e algoritmos**. 2 ed. Rio de Janeiro: Elsevier Editora Ltda, 2005.

GOMES, J. A. C. et al. Aplicação de ferramenta computacional na otimização e mitigação de custos na roteirização da logística de transporte de cargas. **Brazilian Journal of Development**, v. 5, n. 7, p. 7703-7716, jul. 2019. Disponível em: https://www.brazilianjournals.com/index.php/BRJD/article/view/2120. Acesso em: 24 nov. 2022.

JUNIOR, J. D. C.; SILVA, A. A. N. Algoritmo genético aplicado ao problema de roteamento de veículos. **Revista Design e Tecnologia**, Franca, v. 2, n. 2, p. 88-109, ago./dez. 2015. Disponível em: https://publicacoes.unifran.br/revistas/designetecnologia/wp-content/uploads/2016/04/RevistaDesignTecnologia_vol_2_n2__.pdf#page=88. Acesso em: 23 dez. 2022.

LACERDA, E. G. M.; CARVALHO, A. C. P. L. F. Introdução aos algoritmos genéticos. In: GALVÃO, C. O.; VALENÇA, M. J. S. (Org.). **Sistemas inteligentes: aplicações a recursos hídricos e ciências ambientais**. Porto Alegre: Ed. Universidade da UFRGS: Associação Brasileira de Recursos Hídricos, 1999. v. 1, p 89-148.

LIMA, H. A. et al. **Algoritmos genéticos aplicados na exploração de sequências de otimização do compilador**. 2017. 52 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Federal de Uberlândia, Uberlândia, 2017. Disponível em: https://repositorio.ufu.br/handle/123456789/19623. Acesso em: 20 dez. 2022.

LINDEN, R. **Algoritmos Genéticos**. 3. ed. Rio de Janeiro: Ciência Moderna, 2012. 496 p.

MALAQUIAS, N. G. L. **Uso dos algoritmos genéticos para a otimização de rotas de distribuição**. Dissertação (mestrado) – Universidade Federal de Uberlândia, Programa de Pós-graduação em Engenharia Elétrica. 2006.

MELO, A. C. da S.; FILHO, V. J. M. F. Sistemas de roteirização e programação de veículos. **Pesquisa Operacional**, v. 21, n. 2, p. 223-232, jul. 2001. Disponível em: https://www.scielo.br/j/pope/a/TFh7mPx3Mj9RvQpgKxR6SDp/?lang=pt#. Acesso em: 23 dez. 2022.

NÉIA, S. S. et al. Roteamento de veículos utilizando otimização por colônia de formigas e algoritmo genético. In: LOPES, et al. (Eds). **Meta-heurísticas em pesquisa operacional**. cap. 14, p. 219-238, mai. 2013. Disponível em: https://www.researchgate.net/publication/300661461_Roteamento_de_Veiculos_Utilizando_Otimizacao_por_Colonia_de_Formigas_e_Algoritmo_Genetico. Acesso em: 23 dez. 2022.

PACÍFICO, D. da S.; SILVA JÚNIOR, O. S. da. Apresentação do Modelo Matemático para o Problema de Roteirização de Veículos com Janelas de Tempo. In: SIMPÓSIO DE PESQUISA OPERACIONAL E LOGÍSTICA DA MARINHA, 19. 2020, Rio de Janeiro. **Anais...** Rio de Janeiro: Centro de Análises de Sistemas Navais, 2020, p. 839-842. Disponível em: https://www.proceedings.blucher.com.br/article-details/apresentao-do-modelo-matemtico-para-o-problema-de-roteirizao-de-veculos-com-janelas-de-tempo-34475. Acesso em: 28 dez. 2022.

PRESTES, A. N. **Uma análise experimental de abordagens heurísticas aplicadas ao problema do caixeiro viajante**. 2006. 85 f. Dissertação (Mestrado em Sistemas e Computação) - Universidade Federal do Rio Grande do Norte, Natal, 2006. Disponível em: https://repositorio.ufrn.br/handle/123456789/17962. Acesso em: 22 dez. 2022.