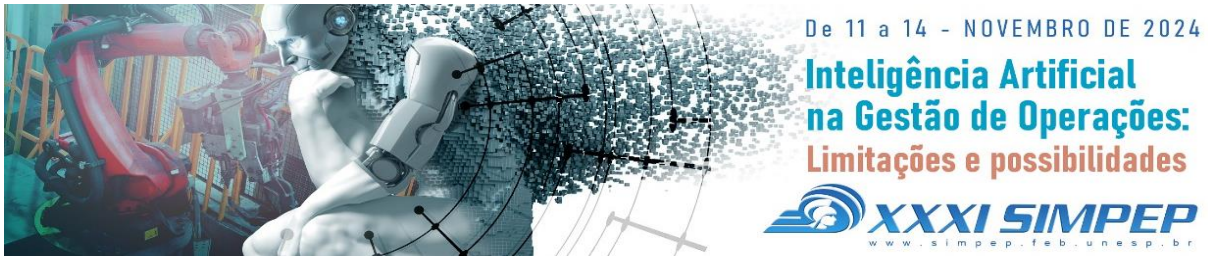


ANÁLISE COMPARATIVA DE ALGORITMOS PARA RESOLUÇÃO DO PROBLEMA DO CAIXEIRO VIAJANTE APLICADO À OTIMIZAÇÃO DE ENTREGAS

VINÍCIUS DOURADO SILVA – dourado.vinicius@uel.br
UNIVERSIDADE ESTADUAL DE LONDRINA - UEL

ERNESTO F. FERREYRA RAMIREZ - ferreyra@uel.br
UNIVERSIDADE ESTADUAL DE LONDRINA - UEL

FRANCISCO GRANZIERA JR - granziera@uel.br
UNIVERSIDADE ESTADUAL DE LONDRINA - UEL

ÁREA: *3. PESQUISA OPERACIONAL*
SUBÁREA: *3.1 - MODELAGEM, SIMULAÇÃO E OTIMIZAÇÃO*

RESUMO: *ESTE TRABALHO TEM COMO OBJETIVO APRESENTAR UMA METODOLOGIA PARA RESOLVER O PROBLEMA DO CAIXEIRO VIAJANTE, VOLTADO PARA REALIZAÇÃO DE ENTREGAS POR PEQUENOS NEGÓCIOS. PARA ISSO, FORAM COMPARADOS DOIS MÉTODOS DE INTELIGÊNCIA ARTIFICIAL (ALGORITMO GENÉTICO DE SELEÇÃO SIMPLES E SIMULATED ANNEALING) COM A BUSCA EXAUSTIVA, QUE CONSISTE EM ENCONTRAR A SOLUÇÃO ÓTIMA PARA UM PROBLEMA, CONSIDERANDO TODAS AS COMBINAÇÕES POSSÍVEIS. OS ALGORITMOS DESENVOLVIDOS FORAM IMPLEMENTADOS EM MATLAB, PARA TESTE INICIAL, E DEPOIS EM PHYTON NA FORMA DE UM APP QUE POSSA SER UTILIZADO PELOS EMPRESÁRIOS E ENTREGADORES. AS DIVERSAS SIMULAÇÕES REALIZADAS SUGEREM QUE A BUSCA EXAUSTIVA É VIÁVEL PARA ENTREGAS MENORES QUE 10 LOCALIDADES. ENTRETANTO, PARA ENTREGAS MAIS ABRANGENTES (SUPERIORES A 10 LOCALIDADES), TANTO O ALGORITMO GENÉTICO COMO O SIMULATED ANNEALING APRESENTAM RESULTADOS MAIS SATISFATÓRIOS, POIS SUGEREM TRAJETOS COM DIFERENÇAS MÁXIMAS DE 20% EM TEMPOS INFERIORES A 02 SEGUNDOS, SEM UTILIZAR MUITOS RECURSOS DE PROCESSAMENTO COMPUTACIONAL, COMO A BUSCA EXAUSTIVA.*

PALAVRAS-CHAVES: *PROGRAMAÇÃO; INTELIGÊNCIA ARTIFICIAL; ALGORITMO GENÉTICO DE SELEÇÃO SIMPLES; RECOZIMENTO SIMULADO.*

INTRODUÇÃO

Com a chegada da era pós-pandemia, a logística enfrenta uma série de desafios e oportunidades que moldam o seu futuro. A crise econômica mundial foi a principal responsável por acelerar o uso de ferramentas digitais na logística, visando o benefício e simplificação das relações de comércio, bem como diferentes estratégias de otimização, visando o aprimoramento nos diversos setores de transporte (LMXlogistica, 2022).

Nesse sentido, destaca-se o Problema do Caixeiro Viajante (PCV), que propõe um algoritmo para otimização de rotas e que abrange várias adaptações na atualidade (BARBOSA; JR.; KASHIWABARA, 2015).

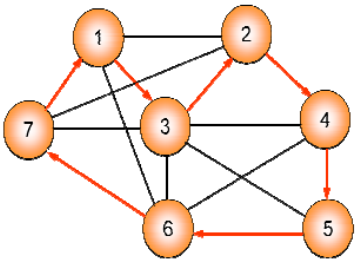
Com o intuito de obter resultados comparativos, três métodos foram utilizados: Busca Exaustiva, consistindo no teste de todas as combinações possíveis, e sempre retornando o resultado exato; e dois métodos de inteligência artificial, como os Algoritmos Genéticos, que são algoritmos comuns e geralmente aplicados em problemas complexos e de difícil modelagem e o *Simulated Annealing*, ou recozimento metalúrgico, também utilizado para problemas complexos e busca por um valor mínimo ou máximos em um sistema qualquer.

2. REFERENCIAL TEÓRICO

2.1 Problema do Caixeiro Viajante

O PCV é um desafio clássico de otimização cujo objetivo é encontrar a rota mais eficiente que um vendedor (caixeiro viajante) pode realizar para visitar uma série de cidades apenas uma vez, antes de retornar ao ponto de partida (BENEVIDES et al., 2011). Uma abordagem do PCV estabelece uma única rota que passe por cada nó de um grafo, como visto na Figura 1, apenas uma vez, retornando ao nó inicial no final do percurso. O problema tem implicações em áreas como logística, transporte e comunicação (CARVALHO, 2018).

FIGURA 1 – Exemplo do PCV.



Fonte: Benevides et al. (2011).

À medida que o número de nós (pontos) aumenta, a complexidade da resolução do PCV também aumenta. Para um problema com "n" nós e uma ligação entre cada par de nós, o número de rotas possíveis pode ser calculado pela Equação (1):

$$Q_{qtdderotas} = (n-1)!/2 \quad (1)$$

onde cada nó possui (n – 1) possibilidades de conexão, e a divisão por dois considera que cada rota tem uma rota reversa equivalente com a mesma distância. Dessa forma, um problema com 10 nós, resulta em aproximadamente 181 mil soluções, enquanto um com 20 nós pode ter até $6{,}08 \times 10^{16}$ soluções, e assim sucessivamente (LIEBERMAN, 2010).

2.2 Métodos Utilizados

Para estabelecer uma linha de otimização para o PCV, três métodos foram escolhidos, sendo eles, o método da Busca Exaustiva, Algoritmo Genético de Seleção Simples e *Simulated Annealing*.

Todos os métodos foram feitos em um ambiente de testes proporcionado pela linguagem Python, sendo está uma linguagem com alta capacidade para resolução de cálculos matemáticos, visualização de resultados, além de uma variedade de bibliotecas e documentação de fácil acesso na internet.

2.2.1 Algoritmos Genéticos

Algoritmos Genéticos (AG), são algoritmos gerais e normalmente são aplicados em problemas complexos, com grandes espaços de busca e difícil modelagem (GOUVEIA et al., 2021). Os AG são considerados como métodos de otimização e se inspira nos mecanismos de evolução de populações de seres vivos (LACERDA; CARVALHO, 1999).

Segundo Barreto (1999), os AG são métodos tradicionais de busca e otimização que se baseiam em quatro critérios, que são:

a. Trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros do problema;

b. Operam em uma população e não em pontos isolados, reduzindo o risco de busca;

c. Utilizam informações de custo ou recompensa (informações da função objetiva p*ayoff*) e não derivadas ou outro conhecimento auxiliar;

d. Procedem a busca utilizando operadores estocásticos e não determinísticos.

A busca pela melhor solução se dá a partir da população inicial, que quando combinada com os melhores desta população, gera uma nova população, substituindo a inicial. A cada mutação ou então, cruzamento, é formada uma nova população que apresenta novas e melhores soluções para o problema, assim, atingindo os melhores resultados (FILITTO, 2008).

2.2.2 *Simulated Annealing*

Outra abordagem para solução do PVC se baseia da técnica de otimização conhecida por *Simulated Annealing* (SA), ou recozimento metalúrgico, técnica que foi usada para simular em um computador o processo de *"annealing"* de cristais, processo de resfriamento gradativo de materiais a partir de uma alta temperatura levando-os a estados mínimos de energia. A ideia de aplicar este método para resolver problemas de otimização combinatória foi proposta por Kirkpatrick et al. (1983).

No recozimento, os metais são aquecidos a altas temperaturas e depois resfriados lentamente, permitindo que os átomos se reorganizem em uma estrutura mais estável. Da mesma forma, o SA começa com uma solução inicial e, em seguida, perturba essa solução de maneira controlada para explorar o espaço de busca (KIRKPATRICK et al., 1983).

O fato do método *Simulated Annealing* permitir a aceitação de configurações intermediárias do problema em que cresce o valor da função objetivo que se deseja minimizar é crucial. Essa aceitação temporária de soluções "piores", significa que o método admite caminhar "morro acima", na esperança de encontrar "vales" mais profundos (BENEVIDES et al., 2011).

O SA é especialmente útil para problemas de otimização complexos, onde métodos de busca local podem ficar presos em mínimos locais. Ao permitir movimentos que aumentam a função objetivo (ou seja, pioram a solução), o SA pode escapar desses mínimos locais e explorar mais completamente o espaço de busca. A probabilidade de aceitar uma solução pior diminui gradualmente à medida que o "resfriamento" do algoritmo prossegue, de forma análoga ao processo de recozimento (REIS et al., 2023).

2.3 Implementação dos Algoritmos

Para validar os algoritmos, a matriz de distâncias e convertida em um banco de dados para possibilitar a formação de um conjunto de dados robusto e diversificado. Isso permite a

análise de uma variedade de rotas, desde as que percorrem longas distâncias entre pontos afastados até as que se concentram em áreas próximas.

Cada ponto no banco de dados é caracterizado por uma série de atributos, incluindo seu nome e a distância para todas as outras cidades, além de sua distribuição geográfica. Esses dados são fundamentais para a modelagem e implementação dos algoritmos de resolução do PCV, pois facilitam a criação de rotas realistas e pertinentes para a análise.

QUADRO 1 – Cidades do Paraná escolhidas aleatoriamente para compor o banco de dados.

Londrina	Ibiporã	Maringá	Cascavel	Paranaguá
Cruzeiro do Oeste	Santa Izabel do Oeste	Brasilândia do Sul	Sengés	Santo Antônio do Sudoeste
Curitiba	Nova Londrina	Umuarama	Arapongas	Alvorada do Sul
Roncador	Fênix	Mauá da Serra	Centenário do Sul	Uraí
Cornélio Procópio	Sapopema	General Carneiro	São José dos Pinhais	Indianópolis
Lobato	Araruma	Jesuítas	Palmas	Perola
São João	Florestópolis	Contenda	Guaratuba	Luiziana
Cafeara	Castro	Lapa	Toledo	Piraquara
Jacarezinho	Matinho	Assai	Paranacity	Primeiro de Maio
Foz do Iguaçu	Borrazópolis	Guapirama	Inajá	Nova Fátima

Fonte: De Autoria Própria.

E para agrupar todas as características e funções do modelo, foi desenvolvido um software em *Python*, dividido em duas partes, o *back-end*, que é responsável por garantir a eficiência e a modularidade do sistema. Cada algoritmo é implementado em um arquivo separado e projetado para ser facilmente integrado ao restante do código. Isso permite comparar e avaliar diferentes abordagens de forma flexível e eficaz.

Já o *front-end*, consiste em uma interface gráfica, que visa um design interativo, limpo e intuitivo, onde o usuário entra com os parâmetros fundamentais necessários: número de cidades, o arquivo .txt, que contém a matriz de distâncias. Ainda há *checkboxes* para a execução de testes onde o número de cidades ou pontos aumenta progressivamente, e *checkboxes* de quais métodos executar.

2.3.1 Busca exaustiva

A Busca Exaustiva é um método direto e sistemático de encontrar a solução ótima para um problema, considerando todas as combinações possíveis de soluções. Portanto, o código necessário deve ser o mais simples possível. A Busca Exaustiva foi escolhida devido à sua

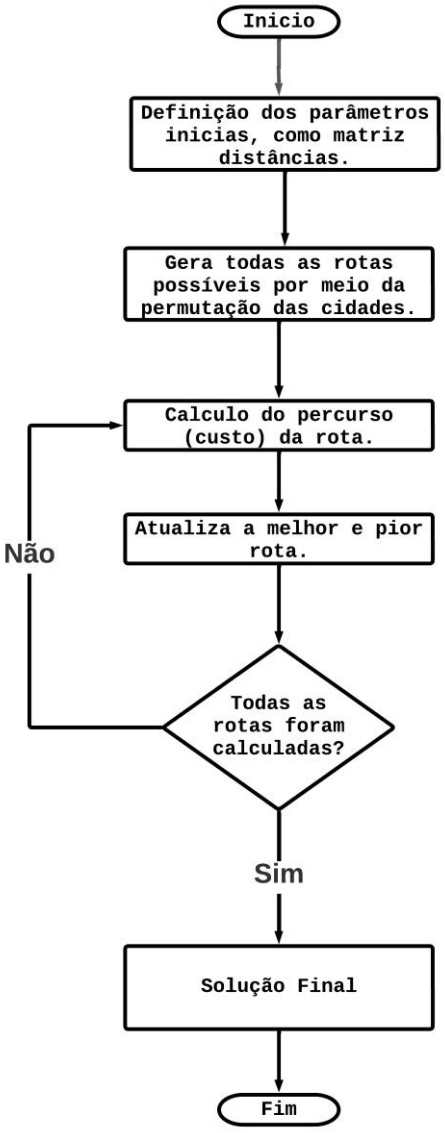
garantia de encontrar a solução ótima em problemas com espaços de busca pequenos e pela sua simplicidade de implementação (BAIDOO; OPPONG, 2016).

O algoritmo de Busca Exaustiva foi implementado de acordo com os seguintes passos:

1. Geração de todas as possíveis soluções para o problema;
2. Avaliação de cada solução gerada;
3. Seleção da solução ótima com base nos critérios de avaliação definidos.

Na Figura 2, é apresentado o fluxograma do algoritmo de Busca Exaustiva, que foi implementado em *Phyton* através das bibliotecas de operações.

FIGURA 2 – Fluxograma da Busca Exaustiva.



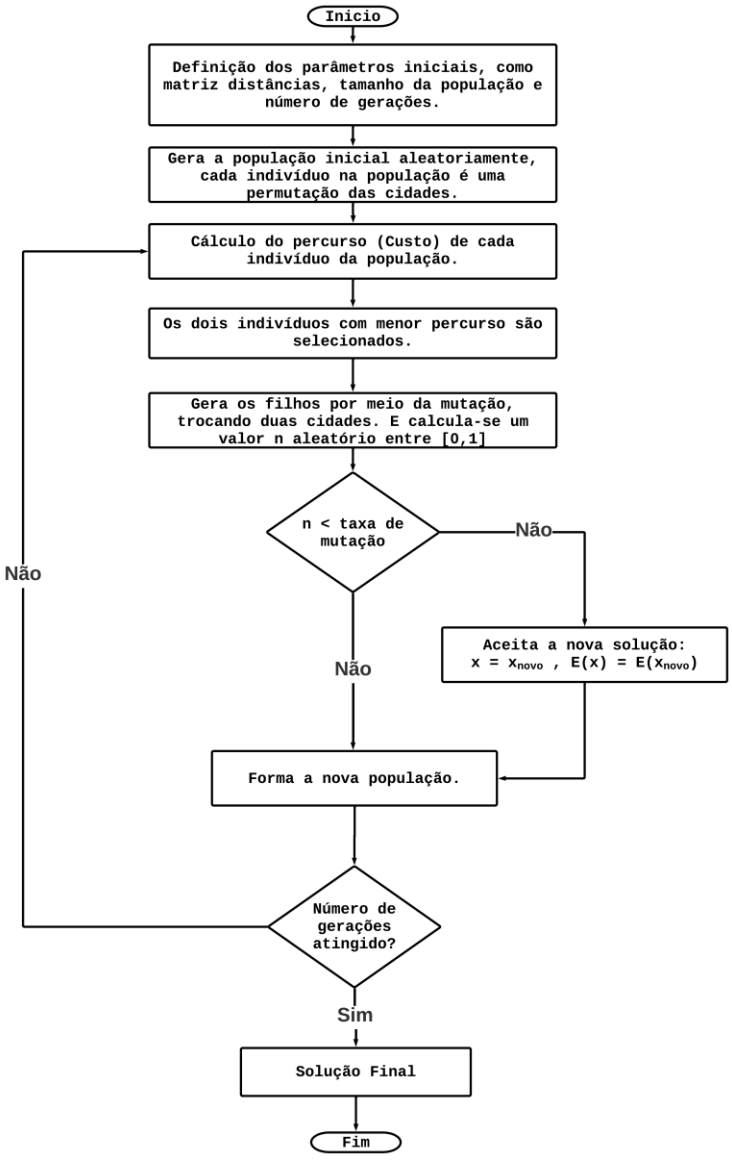
Fonte: De Autoria Própria.

2.3.2 Algoritmo Genético de Seleção Simples

O algoritmo genético proposto busca solucionar o Problema do Caixeiro Viajante aplicando a técnica de otimização baseada em processos evolutivos. Este método foi escolhido devido à sua capacidade de lidar com espaços de busca grandes e complexos, além de sua flexibilidade e adaptabilidade.

Na Figura 3, é apresentado o fluxograma do Algoritmo Genético de Seleção Simples, que foi implementado em *Phyton* através das bibliotecas de operações.

FIGURA 3 – Fluxograma do Algoritmo Genético de Seleção Simples.



Fonte: De Autoria Própria.

A implementação do algoritmo genético incluiu os seguintes passos:

1. Geração inicial de uma população de soluções;
2. Avaliação de cada indivíduo da população gerada anteriormente;
3. Seleção dos indivíduos mais aptos para reprodução;
4. Aplicação de operadores genéticos (*crossover* e mutação) para criar uma nova população;
5. Avaliação da nova população e repetição do processo até atingir um critério de parada.

Na etapa de inicialização, é necessário definir os pontos a serem testados, bem como os parâmetros necessários. Isso inclui determinar o tamanho da população inicial e o número máximo de gerações. Para cada cromossomo na população inicial, um vetor de "n" posições, ou seja, "n" pontos, é gerado, com os índices dos pontos distribuídos aleatoriamente.

Os parâmetros do algoritmo genético foram ajustados empiricamente para obter os melhores resultados, levando em consideração a taxa de crossover, a taxa de mutação e o tamanho da população.

Em seguida, para o processo de evolução, em cada população, o cálculo do percurso total é realizado. Os dois melhores indivíduos são selecionados como pais da próxima população. Cada pai gera filhos através da permutação de duas posições.

2.3.3 Simulated Annealing

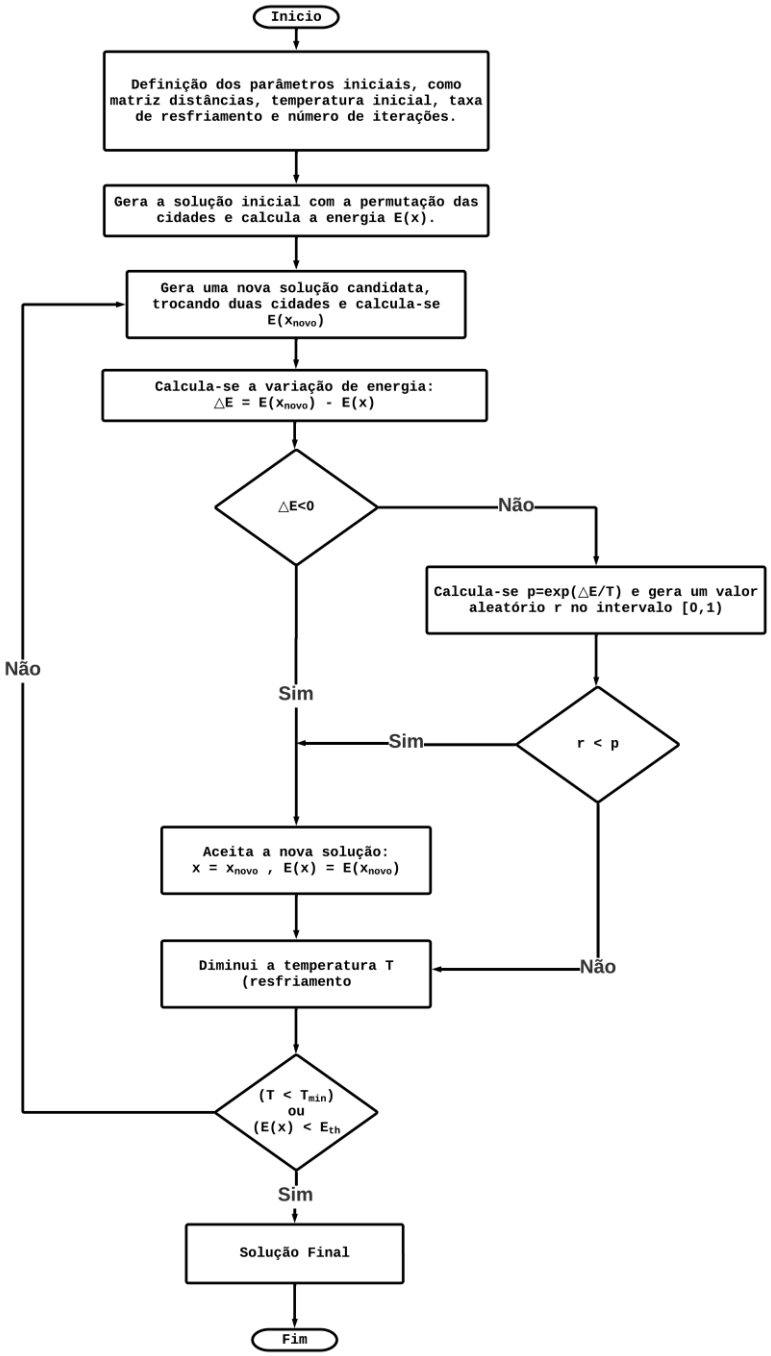
O *Simulated Annealing* é uma técnica de otimização probabilística baseada no processo de recozimento de metais. Ele foi escolhido por sua capacidade de escapar de mínimos locais e explorar mais completamente o espaço de busca. Além disso, o SA é especialmente útil para problemas com espaços de busca complexos e malcomportados.

A implementação do SA envolveu os seguintes passos:

1. Definição aleatória de uma solução inicial;
2. Geração de uma solução vizinha através de uma perturbação controlada;
3. Avaliação da solução vizinha e decisão de aceitar ou rejeitar com base em uma função de probabilidade;
4. Repetição do processo até atingir um critério de parada.

Na Figura 4, é apresentado o fluxograma do *Simulated Annealing* que foi implementado em *Phyton* através das bibliotecas de operação.

FIGURA 4 – Fluxograma do *Simulated Annealing*.



Fonte: De Autoria Própria.

Os parâmetros do *Simulated Annealing*, como a temperatura inicial e a taxa de resfriamento, foram ajustados para equilibrar a exploração do espaço de busca com a intensificação da busca pela solução ótima.

2.4 Avaliação Computacional Pós Ajustes

Para o método de Busca Exaustiva, por ser um método inviável para um grande número de

cidades, o custo para implementar melhorias como paralelização das soluções ou métodos heurísticos, se torna complexo e pouco lucrativo. Portanto há melhorias como, inicializar as variáveis antes, permutar apenas os números que não são fixos, ou seja, os pontos a serem visitados e não os pontos de partida e chegada, já causam um ganho no tempo de execução.

No AG, foram ajustados os parâmetros para encontrar a melhor sintonia, as variáveis são criadas no começo do algoritmo, economizando memória e otimizando o tempo, e assim como na Busca Exaustiva, permutar apenas as cidades não fixas também pode gerar uma otimização no tempo de execução.

Já para o SA, melhorar a eficácia se tornou difícil, tendo em vista que diferentes funções de temperatura não mostravam um grande lucro pelo custo de implementação, portanto foi optado apenas por encontrar uma melhor sintonia nos parâmetros.
Para cada método, foi avaliado seu tempo de execução, suas necessidades computacionais, bem como as melhores distâncias encontradas. Sendo estes critérios os principais responsáveis por orientar a evolução dos algoritmos, apontando onde estão os possíveis problemas, e quais são as áreas de melhoria.

Analisando os Algoritmos implementados quanto aos resultados do tempo de execução, onde estes foram obtidos realizando a média entre 10 simulações, conforme mostrado na Tabela 1.

TABELA 1 – Comparação do tempo de execução (em segundos) obtidas dos algoritmos em diferentes cenários em *Python*.

	BUSCA EXAUSTIVA	**ALGORITMO GENÉTICO**	**SIMULATED ANNEALING**
5 CIDADES	0,00039 s	0,00122 s	0,00489 s
10 CIDADES	5,90100 s	0,00309 s	0,00724 s
11 CIDADES	14,923800 s	0,00819 s	0,00175 s
12 CIDADES	187,2500000 s	0,02914 s	0,00334 s
13 CIDADES	*	0,00670 s	0,00169 s
14 CIDADES	*	0,00755 s	0,00161 s
15 CIDADES	*	0,01178 s	0,00439 s

Fonte: De Autoria Própria.

Legenda:

- *: Não foi possível executar devido a grandes requisitos de memória RAM.
- **: Resultados podem variar com o estado da máquina a ser executado.

Ao analisar os tempos de execução dos algoritmos, observa-se que o *Simulated Annealing* é o método mais rápido, seguido de perto pelo Algoritmo Genético de Seleção Simples.

Para analisar as distâncias, ou seja, para o objetivo final dos algoritmos, foi executado um teste retornando os valores de 10 simulações, sendo cada uma delas com 10 cidades escolhidas aleatoriamente para cada simulação, como apresentado na Tabela 2.

TABELA 2 – Comparação das distâncias (em Km) obtidas dos algoritmos em 10 cenários no *Python*.

Simulações	Busca Exaustiva	Algoritmo Genético de Seleção Simples	Simulated Annealing
1°	1.272,00	1.449,33	1.272,00
2°	2.068,35	2.068,35	2.068,35
3°	1.761,05	1.761,05	1.807,05
4°	1.570,56	1.570,56	1.570,56
5°	1.443,09	1.443,09	1.443,09
6°	1.091,69	1.323,08	1.290,55
7°	1.858,82	2.156,66	1.906,27
8°	1.272,79	1.272,79	1.272,79
9°	1.414,59	1.414,59	1.414,59
10°	2.011,27	2.011,27	2.011,27

Fonte: De Autoria Própria.

Pode-se verificar que com os ajustes dos parâmetros, o AG atinge o resultado ótimo em sete casos, já o SA atinge os mesmos sete acertos, porém em simulações diferentes. Portanto é sim possível obter resultados proveitosos em um tempo reduzido comparado a Busca Exaustiva apenas ajustando os parâmetros.

2.5 Validação em Situação Real

Foi realizado um teste prático para avaliar o desempenho dos algoritmos em uma situação real. O teste determinou a rota mais eficiente entre uma sequência de endereços na região de Londrina-PR.

A rota foi estabelecida manualmente e levaria, segundo o *Google Maps*, aproximadamente 2 horas e 40 minutos para ser concluída, percorrendo aproximadamente 78 km. Foram comparadas as distâncias retornadas pelos algoritmos em relação a essa rota definida

previamente, também quanto tempo aproximadamente levaria a rota sugerida pelos algoritmos.

Esta comparação permitiu avaliar a eficiência dos algoritmos em relação a situação prática do mundo real. A ordem das cidades pré-estabelecida é apresentada no Quadro 2.

Quadro 2 – Rota dos endereços para o teste prático.

Ordem	**Endereços**
Ponto de Saída	Av. do Sol, 330
1°	Rua Saturno, 710
2°	Rua Pres. Washington Luís, 82
3°	Rua Samuel Moura, 400.
4°	Rua Uberlândia, 320
5°	Alexander Graham Bell, 433.
6°	Rua Fábio Paludetto, 100
7°	Rua Nilo Betoni, 226
8°	Rua Serra de Roraima, 415
9°	Rua Serra de Roraima, 325
10°	Rua Dalto, 818, Cambé
11°	Rua Mario José Romagnoli, 351
12°	Rua João Miguel Caram, 1075
13°	Rua João Stringheta, 55
14°	Rua Nilo Gonzales Vicente, 326
15°	Rua José Gabriel de Oliveira, 915
16°	Rua José Roque Salton, 250
17°	Rua João Huss, 1001
18°	Rua Eurico Hummig, 577
Ponto de Chegada	Av. do Sol, 330

Fonte: De Autoria Própria.

Executando os algoritmos com os parâmetros definidos no AG, a população inicial de 200 soluções, o número de gerações é de 1.000 (iterações), e a taxa de mutação é 0,22. Para o SA, o número de iterações é 85.000, a temperatura inicial é 1.350 °C e a taxa de resfriamento é 50 °C, a rota encontrada pode ser vista no Quadro 3.

Quadro 3 – Rota dos endereços para o teste prático.

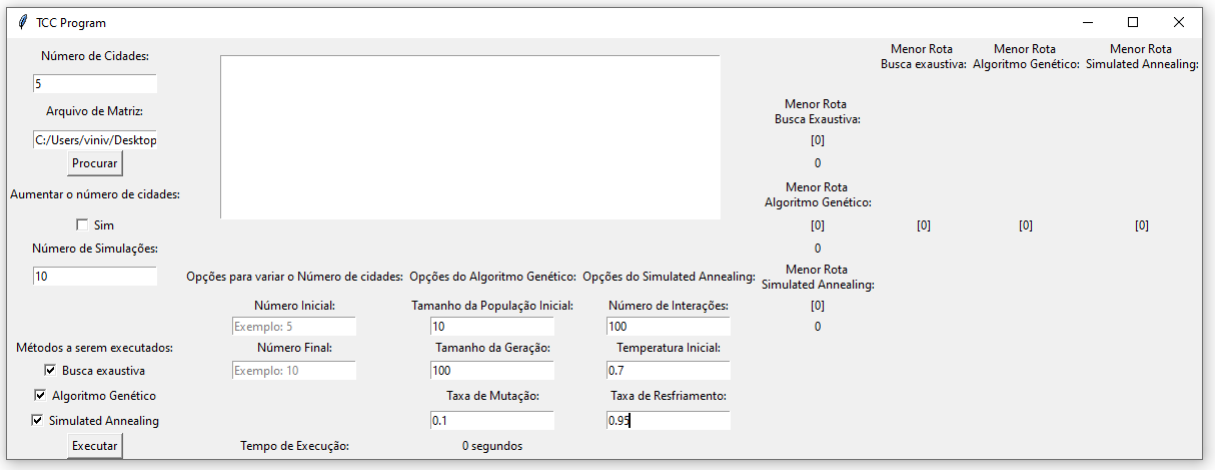
Ordem	Endereços
Ponto de Saída	Av. do Sol, 330
1°	Rua Pres. Washington Luís, 82
2°	Rua Samuel Moura, 400
3°	Rua Uberlândia, 320
4°	Rua Mario José Romagnoli, 351
5°	Rua João Miguel Caram, 1075
6°	Rua João Stringheta, 55
7°	Rua Nilo Gonzales Vicente, 326
8°	Rua José Gabriel de Oliveira, 915
9°	Rua José Roque Salton, 250
10°	Rua Eurico Hummig, 577
11°	Rua João Huss, 1001
12°	Rua Fábio Paludetto, 100
13°	Alexander Graham Bell, 433
14°	Rua Nilo Betoni, 226
15°	Rua Dalto, 818, Cambé
16°	Rua Serra de Roraima, 415
17°	Rua Serra de Roraima, 325
18°	Rua Saturno, 710
Ponto de Chegada	Av. do Sol, 330

Fonte: De Autoria Própria.

A rota sugerida pelo AG é percorrida em 2 horas e 25 minutos, ou seja, 10% mais rápida, e sendo calculada em poucos segundos.

Deste modo, com todos os algoritmos implementados de maneira satisfatória e integrada, a interface, ou programa, foi finalizada. Na Figura 5 é apresentada a interface do programa final para testes, que está disponível no *GitHub*, através do link https://github.com/ViniiciusDS/Software-para-Solu-o-do-PCV.git.

Figura 5 – Imagem de uma entrada padrão feita pelo usuário.



Fonte: De Autoria Própria.

3. CONSIDERAÇÕES FINAIS

Com base nos resultados obtidos neste trabalho, pode-se concluir que os algoritmos propostos de otimização de rotas para solução do problema do caixeiro viajante, por métodos de inteligência artificial como Algoritmo Genético de Seleção Simples e *Simulate Annealing*, apresentaram resultados satisfatórios, ambos com uma taxa de 80% de acerto com sete pontos e 70% com 10 pontos, entretanto no AG, o ajuste dos parâmetros é mais simples e intuitiva.

Além disso, o desenvolvimento de uma interface gráfica do usuário através da biblioteca Tkinter, facilita a análise dos resultados e ajustes dos parâmetros de simulação, pois seu layout é inspirado em linhas e colunas de uma matriz, além de possuir com um campo para entrada de dados, aceitando arquivos no formato txt. Os campos de entrada dos parâmetros servem para que o usuário defina o quão custoso vai ser o resultado obtido, ou o quão rápido ele quer a resposta em detrimento da exatidão da distância.

A seguir são listadas algumas sugestões de temas e melhorias para trabalhos futuros:

- Implementação de uma maneira automatizada da obtenção da matriz distâncias, utilizando a própria API do Google Maps, que possui ampla documentação;
- Otimização e agilização de processos dos algoritmos;
- Busca por uma melhor função temperatura do método *Simulated Annealing*;
- Comparação entre diferentes tipos de algoritmos genéticos;
- Desenvolvimento de algoritmos para ajustes automáticos dos parâmetros;
- Desenvolvimento de algoritmos para combinação entre tempo de viagem e distância;
- Elaboração de uma interface gráfica com mais funções e mais atrativa visualmente;
- Criação de um aplicativo *mobile* otimizado e de fácil visualização para o motoboy.

REFERÊNCIAS

BARBOSA, D.; JR., C.; KASHIWABARA, A. **Aplicação da otimização por colônia de formigas ao problema de múltiplos caixeiros viajantes no atendimento de ordens de serviço nas empresas de distribuição de energia elétrica.** In: Anais do XI Simpósio Brasileiro de Sistemas de Informação. Porto Alegre, RS, Brasil: SBC, 2015. p. 23–30. ISSN 0000-0000.

BAIDOO, E.; OPPONG, S. O. **Solving the tsp using traditional computing approach**. International Journal of Computer Applications, Foundation of Computer Science, v. 152, n. 8, p. 13–19, 2016.

BARRETO, J. M. **Inteligência artificial no limiar do século xxi**. Florianópolis: PPP edições, v. 97, 1999.

BENEVIDES, P. F. et al. **Aplicação de algoritmos genéticos e simulated annealing para o problema do caixeiro viajante em uma situação real de distribuição de produtos**. Brasil, 2011.

CARVALHO, A. V. d. **O problema do caixeiro viajante com múltiplos passageiros e quota.** Dissertação (Mestrado em Sistemas e Computação) – Universidade Federal do Rio Grande do Norte – UFRN, Natal, 2018.

FILITTO, D. **Algoritmos genéticos: uma visão explanatória**. Revista Multidisciplinar da, 2008

GOUVEIA, L. B. et al. **Algoritmos genéticos: aplicando a teoria a um estudo de caso**. Brazilian Journal of Development, v. 7, n. 3, p. 21053–21077, 2021.

KIRKPATRICK, S. et al. **Optimization by simulated annealing**. Science, v. 220, n. 4598, p. 671–680, 1983. Disponível em: https://www.science.org/doi/abs/10.1126/science.220.4598.671.

LACERDA, E. G. de; CARVALHO, A. D. **Introdução aos algoritmos genéticos**. Sistemas inteligentes: aplicaçoes a recursos hıdricos e ciências ambientais, v. 1, p. 99–148, 1999.

LMXlogistica. **MERCADO LOGÍSTICO PÓS-PANDEMIA: O QUE ESPERAR?** 2022. Acesso em: 11 de julho de 2024. Disponível em: https://www.lmxlogistica.com.br/mercado-logistico-pos-pandemia-o-que-esperar/#:~:text=Principais%20mudanÃğas%20na%20logÃŋstica%20pÃşs-pandemia&text=A%20crise%20tem%20contribuÃŋdo%20para,transporte%20ferroviÃąrio%2C%20rodoviÃąrio%20e%20aÃľreo.

LIEBERMAN, F. **Introdução à Pesquisa Operacional**. McGraw Hill Brasil, 2010. ISBN 9788563308283. Disponível em: https://books.google.com.br/books?id=XM_4wAEACAAJ.

REIS, A. C. F. et al. **A TERMODINÂMICA COMO ALGORITMO DE OTIMIZAÇÃO**. n. 2015, 2023.