UNIVERSIDADE ESTADUAL DO PARANÁ - UNESPAR
CAMPUS DE APUCARANA

LISSA GUIRAU KAWASAKI
THEO OKAGAWA RODRIGUES

ALGORITMO GENÉTICO PARA RESOLVER O PROBLEMA DO CAIXEIRO VIAJANTE

Apucarana
Maio de 2025

LISSA GUIRAU KAWASAKI
THEO OKAGAWA RODRIGUES

ALGORITMO GENÉTICO PARA RESOLVER O PROBLEMA DO CAIXEIRO VIAJANTE

Trabalho acadêmico apresentado à disciplina Inteligência Artificial.

Orientadora: Prof.ª Lailla Milainny Siqueira Bine

Apucarana
Maio de 2025

Sumário

1 Introdução

O Problema do Caixeiro Viajante (TSP, do inglês Traveling Salesman Problem) é um dos problemas de otimização combinatória mais estudados na literatura científica. Classificado como NP-difícil, o TSP consiste em encontrar o menor caminho possível que permita a um viajante visitar todas as cidades de um conjunto exatamente uma vez, retornando à cidade de origem. Apesar de sua formulação aparentemente simples, o TSP possui uma complexidade computacional que cresce exponencialmente com o número de cidades, tornando inviável a aplicação de métodos exatos para instâncias de grande porte.

Nesse contexto, novas soluções computacionais emergem como alternativas para obtenção de soluções aproximadas de boa qualidade e em tempo computacional aceitável. Entre essas abordagens, os Algoritmos Genéticos (AGs) destacam-se por sua capacidade de explorar eficientemente o espaço de busca através de mecanismos inspirados na teoria da evolução natural.

Este trabalho tem como objetivo analisar o desempenho de um Algoritmo Genético na resolução do TSP em instâncias de diferentes dimensões. Especificamente, busca-se:

- Implementar um Algoritmo Genético com operadores adequados para o TSP;
- Avaliar a qualidade das soluções obtidas em comparação com os valores ótimos conhecidos;
- Analisar o comportamento de convergência do algoritmo em diferentes cenários;

Esse algoritmo tem como objetivo o entendimento das potencialidades e limitações dos Algoritmos Genéticos quando aplicados a problemas combinatórios de diferentes escalas.

2 Implemetação do Algoritmo

Para a implementação do Algoritmo Genético, foi necessário definir componentes especificios para o funcionamento do código.

- Representação dos cromossomos: Os cromossomos são representados como permutação das cidades. Uma lista gerada aleatoriamente representa uma ordem de visitação das cidades, ou seja, uma rota. Cada gene é o índice de uma cidade, e a sequência indica a ordem do percurso.

- Método de seleção: O método de seleção utilizado é a seleção por roleta (roulette wheel selection). Cada indivíduo tem uma probabilidade proporcional ao seu fitness. Indivíduos com melhor desempenho (rota mais curta) têm maior chance de serem selecionados como pais.

- Operadores de crossover e mutação: O operador de crossover é o crossover ordenado. Esse tipo de crossover mantém parte da sequência de um dos pais e preenche o restante com os genes do outro pai, preservando a ordem e evitando duplicações.

- Critério de parada: O critério de parada é um número fixo de gerações.

2.1 Parâmetros do Algoritmo

Os parâmetros foram gerados aleatóriamentes pelo chatbot de Inteligência Artificial.

Os seguintes parâmetros foram utilizados em todas as execuções:

- Tamanho da População: 50 indivíduos
- Número de Gerações: 100
- Taxa de Crossover: 0.8 (80%)
- Taxa de Mutação: 0.05 (5%)
- Tamanho do Elitismo: 2 indivíduos
- Semente Aleatória: 42 (para reprodutibilidade)

Tamanho da População (50): Um valor moderado que equilibra diversidade genética e custo computacional. Populações muito pequenas podem convergir prematuramente para soluções subótimas, enquanto populações muito grandes aumentam o tempo de processamento sem necessariamente melhorar os resultados.

Número de Gerações (100): Definido para permitir evolução suficiente das soluções sem tornar o tempo de execução excessivo, especialmente para as instâncias maiores como pcb442.

Taxa de Crossover (0.8 ou 80): Valores entre 0.6 e 0.9 são comuns na literatura. Uma taxa alta de crossover favorece a exploração do espaço de busca através da recombinação de soluções existentes.

Taxa de Mutação (0.05 ou 5): Valores baixos (entre 1 e 10) são típicos para evitar disrupção excessiva de boas soluções, enquanto ainda introduzem diversidade suficiente para escapar de ótimos locais.

Tamanho do Elitismo (2): Preservar os dois melhores indivíduos de cada geração garante que boas soluções não sejam perdidas, acelerando a convergência sem comprometer a diversidade.

Semente Aleatória (42): Valor arbitrário para garantir reprodutibilidade dos resultados entre execuções.

2.2 Métricas de Avaliação

Para avaliar o desempenho do algoritmo nas diferentes instâncias, foram utilizadas as seguintes métricas:

- Qualidade da Solução: Medida pela distância total do melhor circuito encontrado e pelo erro percentual em relação ao valor ótimo conhecido.
- Convergência: Análise da evolução da melhor solução ao longo das gerações, incluindo a velocidade de convergência e a capacidade de escapar de ótimos locais.
- Tempo de Execução: Medida do tempo computacional requerido para completar o número especificado de gerações, permitindo analisar a escalabilidade do algoritmo em função do tamanho da instância.

Para cada instância, foram registrados os valores da melhor solução a cada geração, possibilitando a construção de gráficos de convergência. Além disso, foram calculadas métricas

específicas como a melhoria percentual nas primeiras 10 gerações e nas gerações restantes, fornecendo insights sobre o comportamento do algoritmo em diferentes fases do processo evolutivo.

3 Resultados Obtidos

3.1 Qualidade das Soluções

Os resultados obtidos pelo Algoritmo Genético implementado para as três instâncias do TSP são apresentados na Tabela. Para cada instância, são mostrados o número de cidades, a melhor distância encontrada pelo algoritmo, o valor ótimo conhecido, o erro percentual em relação ao ótimo e o tempo de execução.

Instância	**Cidades**	**Melhor Distância**	**Ótimo Conhecido**	**Erro (%)**	**Tempo (s)**
burma14	14	3346.00	3323	0.69	0.35
kroA100	100	115120.00	21282	440.93	2.13
pcb442	442	677410.00	50778	1234.06	13.58

Observa-se uma clara relação entre o tamanho da instância e a qualidade da solução obtida. Para a instância menor (burma14), o algoritmo conseguiu encontrar uma solução muito próxima do ótimo, com erro de apenas 0,69. No entanto, para as instâncias maiores, o desempenho deteriorou-se significativamente, com erros de 440,93 para kroA100 e 1.234,06 para pcb442. Estes resultados evidenciam uma limitação conhecida dos Algoritmos Genéticos quando aplicados a problemas combinatórios de grande porte: a dificuldade em explorar eficientemente espaços de busca muito extensos com os parâmetros e operadores tradicionais. À medida que o número de cidades aumenta, o espaço de soluções cresce fatorialmente, tornando cada vez mais improvável que o algoritmo encontre soluções próximas do ótimo global sem mecanismos adicionais de intensificação da busca. É importante ressaltar que os parâmetros utilizados (tamanho da população, número de gerações, etc.) foram mantidos constantes para todas as instâncias, visando uma comparação justa. No entanto, para instâncias maiores, seria recomendável aumentar esses valores, especialmente o tamanho da população e o número de gerações, para permitir uma exploração mais abrangente do espaço de busca.

3.2 Análise dos Gráficos de Convergência

burma14 (14 cidades):

- Distância inicial: 4849.00

- Distância final: 3346.00
- Melhoria nas primeiras 10 gerações: 792.00 (16.33%)
- Melhoria nas gerações restantes: 711.00 (17.53%)
- A convergência foi gradual, com melhorias significativas mesmo em gerações avançadas.

kroA100 (100 cidades):

- Distância inicial: 149020.00
- Distância final: 115120.00
- Melhoria nas primeiras 10 gerações: 10930.00 (7.33%)
- Melhoria nas gerações restantes: 22970.00 (16.63%)
- A convergência foi gradual, com melhorias significativas mesmo em gerações avançadas.

pcb442 (442 cidades):

- Distância inicial: 731168.00
- Distância final: 677410.00
- Melhoria nas primeiras 10 gerações: 10801.00 (1.48%)
- Melhoria nas gerações restantes: 42957.00 (5.96%)
- A convergência foi gradual, com melhorias significativas mesmo em gerações avançadas.

3.3 Análise do Tempo de Execução

O gráfico de tempo de execução mostra como o tempo computacional aumenta com o número de cidades:

- O tempo de execução aumentou 38.42x ao passar de 14 para 442 cidades.
- O número de cidades aumentou 31.57x.
- O crescimento do tempo é superlinear em relação ao tamanho da instância, o que é esperado para problemas NP-difíceis como o TSP.

4 Conclusão

Para instâncias de pequeno porte, como burma14 (14 cidades), o algoritmo demonstrou capacidade de encontrar soluções muito próximas do ótimo global, com erro percentual inferior a 1. Este resultado é particularmente notável considerando os parâmetros relativamente modestos utilizados (população de 50 indivíduos e 100 gerações). No entanto, para instâncias de médio e grande porte (kroA100 e pcb442), o desempenho deteriorou-se significativamente, com erros percentuais elevados em relação aos valores ótimos conhecidos.

Esta degradação evidencia a dificuldade dos Algoritmos Genéticos tradicionais em explorar eficientemente espaços de busca muito extensos,característicos de problemas combinatórios de grande dimensão.

A análise de convergência revelou padrões distintos entre as instâncias, com as maiores apresentando convergência mais lenta nas gerações iniciais, mas mantendo melhorias significativas mesmo em gerações avançadas. Este comportamento sugere que, para instâncias maiores, seria benéfico aumentar o número de gerações para permitir uma exploração mais completa do espaço de busca.

Quanto ao tempo de execução, observou-se um crescimento superlinear em relação ao tamanho da instância, como esperado para problemas NP-difíceis. No entanto, mesmo para a maior instância (442 cidades), o tempo computacional permaneceu relativamente baixo, demonstrando a eficiência da implementação.

As limitações identificadas neste estudo apontam para futuras melhorias, como a união do Algoritmo Genético com métodos de busca local, a utilização de operadores genéticos mais evoluídos e a implementação de mecanismos adaptativos para ajuste dinâmico dos parâmetros.

Estas estratégias poderiam melhorar significativamente o desempenho do algoritmo em instâncias de maior porte, aproximando-o de métodos especializados para o TSP.

5 Referências

- TSPLIB. Disponível em: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/. Acesso em: 19 mai. 2025.
- Valores ótimos para instâncias simétricas do TSP. Disponível em: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html. Acesso em: 19 mai. 2025.
- POTVIN, J.-Y. Genetic Algorithms for the Traveling Salesman Problem. Annals of Operations Research, v. 63, n. 3, p. 337-370, 1996.