Algoritmo Genético Para Resolver O Problema do Caixeiro Viajante.

Análise de Desempenho em Diferentes Instâncias

Lissa Guirau Kawasaki e Theo Okagawa Rodrigues

Disciplina de Inteligência Artifical no Curso de Ciência da Computação
Universidade Estadual do Paraná

May 19, 2025

Objetivos do Algoritmo:

- Implementar um Algoritmo Genético com operadores que resultam em uma resolução aproximada do TSP.
- Avaliar a qualidade das soluções em comparação com valores ótimos conhecidos.
- Analisar o comportamento de convergência em diferentes cenários.
- Investigar a relação entre tamanho da instância e tempo computacional.

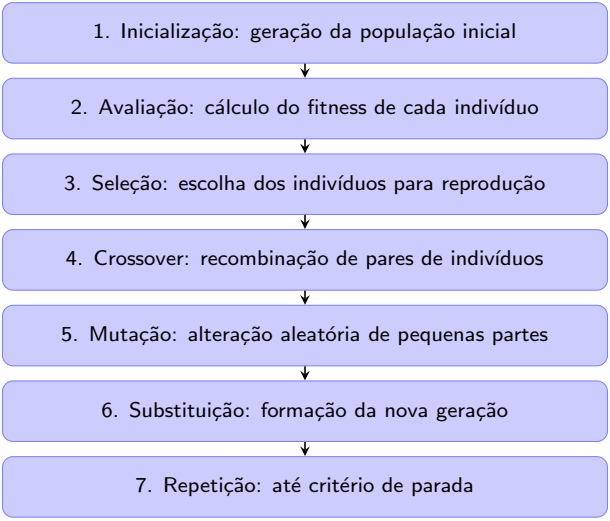
O Problema do Caixeiro Viajante

- Problema clássico de otimização combinatória
- Objetivo: encontrar o menor caminho que permita visitar todas as cidades uma vez, retornando à origem
- Aplicações: logística, planejamento de rotas, design de circuitos
- Representação: grafo completo com custos associados às arestas

Implementação

- Linguagem: Python
- Biblioteca: NumPy , Random, TSPLIB95, Time, Argparse, Sys
- Estrutura de dados: matrizes de adjacência para distâncias
- Representação do indivíduo: lista de inteiros (permutação)
- Função de fitness: inverso da distância total do circuito

Algoritmos Genéticos: Processo



Operadores Genéticos para o TSP

Representação de Cromossomos: permutação de inteiros.

Crossover Ordenado (OX1):

Pai 1: [1 2 3 **4 5 6** 7 8 9]
Pai 2: [9 8 7 6 5 4 3 2 1]
↓
Filho: [9 8 **4 5 6** 7 3 2 1]

Mutação por Troca (Swap):

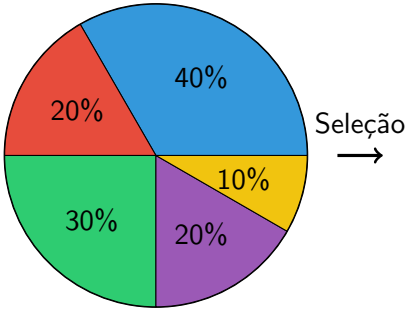
Antes: [1 2 **3** 4 5 **6** 7 8 9]
↓
Depois: [1 2 **6** 4 5 **3** 7 8 9]

Ambos os operadores mantêm a validade da solução (sem repetições de cidades).

Operadores Genéticos para o TSP

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

Representação Visual



- Indivíduo 1 (fitness = 40)
- Indivíduo 2 (fitness = 20)
- Indivíduo 3 (fitness = 30)
- Indivíduo 4 (fitness = 20)
- Indivíduo 5 (fitness = 10)

Instâncias e Parâmetros

Instância	Cidades	Característica	Valor Ótimo
burma14	14	Pequeno porte	3.323
kroA100	100	Médio porte	21.282
pcb442	442	Grande porte	50.778

- **Tamanho da população:** 50 indivíduos
- **Número de gerações:** 100
- **Crossover:** taxa = 0,8
- **Mutação:** taxa = 0,05
- **Elitismo:** 2 melhores indivíduos

Métricas de Avaliação

Qualidade da Solução:

- Distância total do melhor circuito
- Erro percentual em relação ao ótimo

Convergência:

- Evolução da melhor solução ao longo das gerações
- Melhoria percentual em diferentes fases

Tempo de Execução:

- Relação com o tamanho da instância
- Escalabilidade do algoritmo

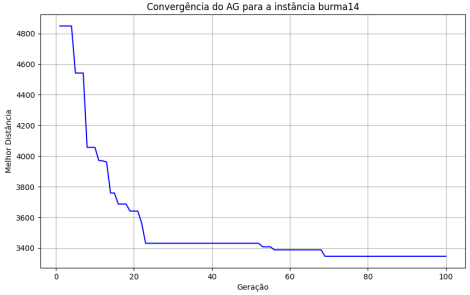
Resultados: Qualidade das Soluções

Instância	Cidades	Melhor Distância	Ótimo	Erro (%)	Tempo (s)
burma14	14	3.346,00	3.323	0,69	0,35
kroA100	100	115.120,00	21.282	440,93	2,13
pcb442	442	677.410,00	50.778	1.234,06	13,58

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.

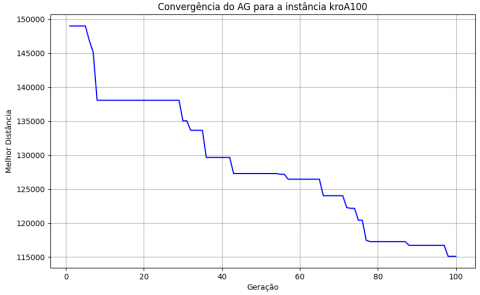
Resultados: Convergência burma14

- **Distância inicial:** 4.849,00
- **Distância final:** 3.346,00
- **Melhoria nas primeiras 10 gerações:** 792,00 (16,33%)
- **Melhoria nas gerações restantes:** 711,00 (17,53%)



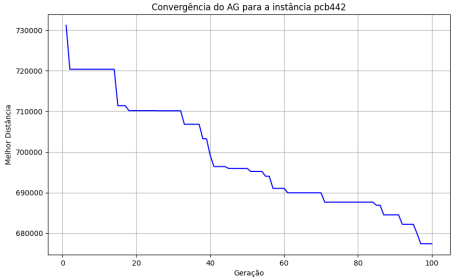
Resultados: Convergência kroA100

- **Distância inicial:** 149020.00
- **Distância final:** 115120.0
- **Melhoria nas primeiras 10 gerações:** 10930.00 (7.33%)
- **Melhoria nas gerações restantes:** 22970.00 (16.63%)



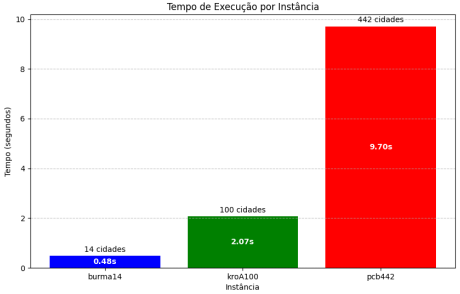
Resultados: Convergência pcb442

- **Distância inicial:** 31168.00
- **Distância final:** 77410.00
- Melhoria nas primeiras 10 gerações: 10801.00 (1.48%)
- Melhoria nas gerações restantes: 42957.00 (5.96%)



Resultados: Tempo de Execução

- Crescimento superlinear com o tamanho da instância
- Complexidade dominada pela avaliação de fitness ($O(n^2)$ por indivíduo)
- Tempo total: $O(n^2 \times pop\_size \times max\_gen)$



Discussão: Limitações Identificadas

- Deterioração significativa do desempenho em instâncias maiores
- Dificuldade em explorar eficientemente espaços de busca muito extensos
- Parâmetros fixos inadequados para instâncias de diferentes tamanhos
- Operadores genéticos básicos insuficientes para problemas complexos

Discussão: Estratégias de Melhoria

1. **Aumento dos Parâmetros:** população e gerações proporcionais ao tamanho da instância
2. **Operadores Adaptados:** crossover e mutação mais avançados para o TSP
3. **Inicialização Heurística:** uso de heurísticas construtivas em vez de permutações aleatórias
4. **Parâmetros Adaptativos:** ajuste dinâmico durante a execução

Conclusões

Para a Implementação Atual:

- AG eficaz para instâncias pequenas
- Padrões distintos de convergência entre instâncias
- Crescimento superlinear do tempo computacional
- Necessidade de estratégias avançadas para instâncias maiores

Para Implementações Futuras:

- Implementação de abordagens híbridas (AG + busca local)
- Desenvolvimento de operadores específicos para o TSP
- Estudo de mecanismos adaptativos para ajuste dinâmico de parâmetros

Agradecimento

Obrigado pela atenção!

Em seguida, vamos apresentar o código rapidamente.

Referências Principais:

TSPLIB.
Valores ótimos para instâncias simétricas do TSP.
POTVIN, J.-Y. Genetic Algorithms for the Traveling Salesman Problem. Annals of Operations Research, v. 63, n. 3, p. 337-370, 1996.