Relatório — Solução do Problema do Caixeiro Viajante com Algoritmo Genético

---

1. Introdução a Algoritmos Genéticos e ao Problema do Caixeiro Viajante (TSP)

Algoritmos Genéticos (AGs) são métodos de busca e otimização inspirados nos mecanismos de evolução natural. Eles utilizam uma população de soluções candidatas, as quais evoluem ao longo de várias gerações por meio de operadores como seleção, crossover (recombinação) e mutação. Seu poder reside na capacidade de explorar eficientemente grandes espaços de busca mesmo quando não há informações derivadas (como gradientes).

O Problema do Caixeiro Viajante (TSP — *Travelling Salesman Problem*) é um problema clássico da ciência da computação e otimização combinatória. Dado um conjunto de cidades e as distâncias entre elas, o objetivo é encontrar o menor caminho possível que visite cada cidade exatamente uma vez e retorne à cidade de origem. O TSP pertence à classe de problemas NP-difíceis.

---

2. Detalhes da Implementação

a. Arquitetura Geral

O algoritmo foi implementado em Python utilizando a biblioteca `tsplib95` para carregar instâncias do TSP em formato `.tsp`. As distâncias são pré-computadas em uma matriz com base no tipo da instância (EUC_2D, ATT ou GEO). A abordagem foi projetada para ser modular e extensível, permitindo o uso de diferentes tipos de instâncias.

b. Decisões de Projeto

- A escolha pelo operador de seleção por torneio se deu pela simplicidade e robustez.
- O crossover implementado foi o de ordem (Order Crossover - OX), uma técnica adequada para problemas de permutação como o TSP.
- A mutação foi realizada por troca de posições aleatórias com uma taxa pequena (2%).
- Não foi utilizado elitismo na primeira versão, mas o código pode ser facilmente estendido para isso.

- A avaliação da solução é baseada no custo total da rota (fitness).
- O critério de parada foi definido por um número fixo de gerações (1000).

---

3. Experimentos Realizados e Análise dos Gráficos

Três instâncias do TSP foram executadas:

- `burma14.tsp` (14 cidades, tipo GEO)
- `att48.tsp` (48 cidades, tipo ATT)
- `gr202.tsp` (202 cidades, tipo EUC_2D)

Para cada instância, o algoritmo gerou:

- **Gráfico de Convergência**: mostra a evolução da melhor distância ao longo das gerações.
- **Gráfico Comparativo de Tempo**: compara o tempo de execução entre instâncias.

Esses gráficos revelaram que:

- O algoritmo convergiu rapidamente para instâncias pequenas.
- Instâncias maiores, como `gr202`, exigiram mais gerações e tempo para estabilizar a solução.
- A qualidade da solução diminui à medida que o número de cidades aumenta, o que é esperado dada a complexidade do problema.

---

4. Conclusão

O Algoritmo Genético demonstrou ser eficaz na obtenção de soluções razoáveis para o TSP em tempo aceitável, especialmente para instâncias pequenas e médias. A qualidade da solução foi próxima do ótimo conhecido em muitos casos, especialmente para `burma14`. Para

instâncias maiores, há espaço para melhorias com ajustes de parâmetros, elitismo e hibridização com heurísticas locais.

O AG mostrou-se uma ferramenta valiosa e flexível para problemas de otimização combinatória como o TSP.

---

b. Parâmetros e Estratégias Utilizadas

i. Representação dos Cromossomos:

Cada cromossomo é uma **permutação das cidades** (lista de inteiros representando a ordem de visita).

ii. Método de Seleção:

Utilizou-se **seleção por torneio** com tamanho `k = 3`, onde três indivíduos são sorteados e o de melhor aptidão é escolhido.

iii. Operadores de Crossover e Mutação:

- **Crossover:** do tipo **Order Crossover (OX)**, preservando a ordem parcial de um dos pais.
- **Mutação:** por troca aleatória de duas posições, com taxa de mutação de **2%**.

iv. Critério de Parada:

O algoritmo é executado por **1000 gerações fixas**, independentemente da convergência da solução.