Relatório - Algoritmo Aproximativo para TSP
Algoritmos e Estruturas de Dados 3
Henrique Versiani e Sofia Petersen

Neste trabalho, abordamos o problema do Caixeiro Viajante (TSP), utilizando algoritmo de força bruta que dá o resultado exato e algoritmos aproximativos para encontrar soluções práticas e eficientes. Devido à alta complexidade e ao tempo de execução necessário, tornou-se inviável resolver instâncias maiores de forma exata dentro dos limites computacionais disponíveis.

Os algoritmos escolhidos para este estudo foram Vizinho Mais Próximo e Inserção, que se destacam pela simplicidade e eficiência na obtenção de soluções aproximadas. Ambos fornecem resultados satisfatórios em instâncias de diversos tamanhos, permitindo o equilíbrio entre precisão e tempo de execução.

Estrutura e Etapas da Solução

O código foi estruturado em diferentes funções. As etapas realizadas para obter as soluções foram:

1. **Carregamento da Matriz de Adjacência**
   A partir de um arquivo de entrada contendo as distâncias entre as cidades, a matriz de adjacência é carregada e armazenada como uma lista de listas. Essa matriz é a base para o processamento dos algoritmos.
2. **Implementação dos Algoritmos de Solução**
   Foram implementados três métodos principais para a resolução do TSP:
   - **Força Bruta (Exato):** Testa todas as permutações possíveis de cidades para encontrar o caminho de menor custo.
   - **Heurística do Vizinho Mais Próximo:** Escolhe iterativamente o vértice mais próximo ainda não visitado, formando um ciclo Hamiltoniano.
   - **Heurística de Inserção:** Parte de um ciclo inicial com dois vértices e insere progressivamente novos vértices em posições que minimizem o custo total.

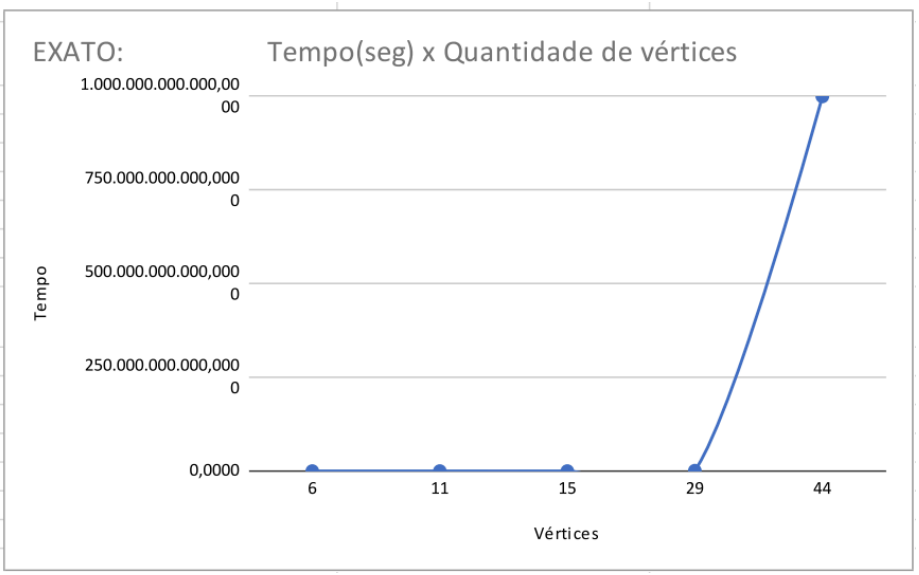
Resultados

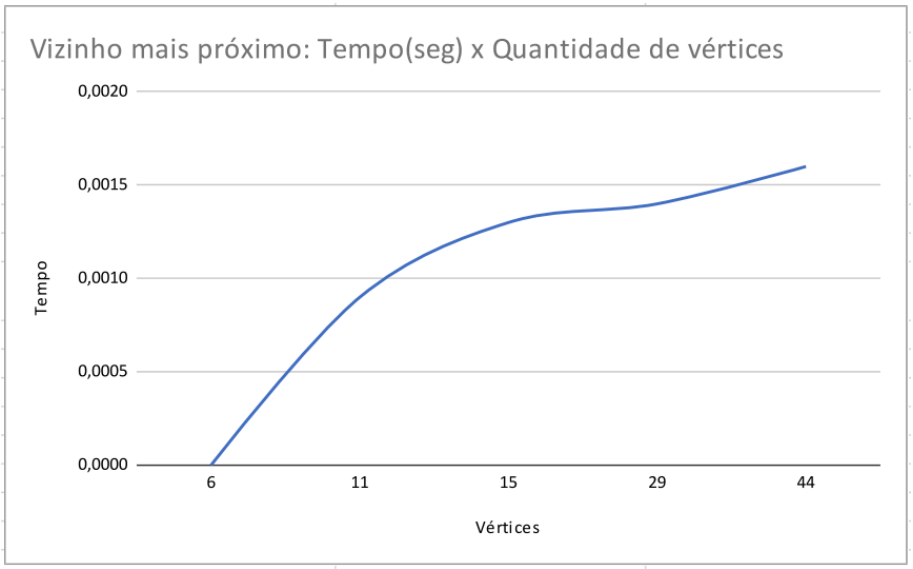
Exato		Tempo (segundos):	Resultados:
Arquivo:	Qnt de Vertices:	Exato:	Exato:
tsp2_1248	6	0,0000	1248
tsp1_253	11	5,8640	253
tsp3_1194	15	96.212,8987	1194
tsp5_27603	29	999.999.999,0000	-
tsp4_7013	44	999.999.999.999,0000	-

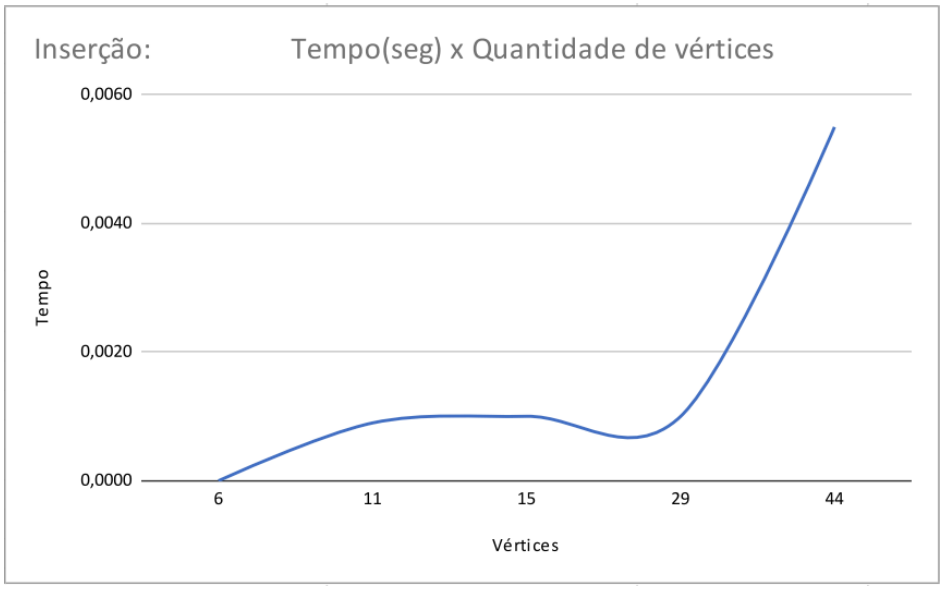
Vizinho Mais Próximo		Tempo (segundos):	Resultados:
Arquivo:	Qnt de Vertices:	Aproximativo:	Aproximativo:
tsp2_1248	6	0,0000	1272
tsp1_253	11	0,0009	299
tsp3_1194	15	0,0013	1260
tsp5_27603	29	0,0014	36399
tsp4_7013	44	0,0016	10587

Inserção		Tempo (segundos):	Resultados:
Arquivo:	Qnt de Vertices:	Aproximativo:	Aproximativo:
tsp2_1248	6	0,0000	1248
tsp1_253	11	0,0009	266
tsp3_1194	15	0,0010	1319
tsp5_27603	29	0,0010	30366
tsp4_7013	44	0,0055	7710

Gráficos







Ambos os métodos aproximativos apresentam tempos de execução extremamente baixos (praticamente zero segundos em todas as instâncias). Isso é esperado, já que são heurísticas que evitam a explosão combinatória do método exato.

A heurística do Vizinho Mais Próximo se destaca pela simplicidade e, em alguns casos (como no tsp3_1194), pode gerar soluções mais precisas do que a heurística Inserção. No entanto, tem maior chance de produzir caminhos subótimos, especialmente em instâncias maiores, como tsp4_7013 e tsp5_27603. Nestes testes, o método da Inserção teve resultados melhores em 80% das matrizes.

A heurística da Inserção, ainda que tenha resultados mais precisos na maioria dos testes, é mais lenta quando comparada com a heurística do Vizinho Mais Próximo. Isso pode ser notado nos gráficos através dos números obtidos durante a execução dos dois métodos.

Como Rodar o Código

Para executar o projeto do TSP disponível no GitHub, siga os passos abaixo:

1. Clone o repositório do projeto com o comando abaixo:

   ```
   git clone https://github.com/sofiapetersen/TSP
   ```

2. Depois de clonar, entre no diretório do projeto:

   ```
   cd TSP
   ```

3. Executar o código:

   ```
   python main.py
   ```

4. Não esqueça de alterar qual o arquivo deseja testar na chamada da função:

   ```
   files = ["tsp1_253.txt", "tsp2_1248.txt", "tsp3_1194.txt",
   "tsp4_7013.txt", "tsp5_27603.txt"]

   solve_tsp(files[0]) # 0 a 4 seguindo o vetor acima
   ```