**TRABALHO DE ALGORITMOS E ESTRUTURA DE DADOS III**

**LUANA MONTILHA PINHEIRO**
**MARIA CELESTE**
**GABRIEL MENDES**

# RELATÓRIO
# PROBLEMA DO CAIXEIRO VIAJANTE (TSP)

**PELOTAS, RS**
**2025**

**INTRODUÇÃO:**

O problema em questão (**Caixeiro viajante**) é considerado um problema NP-Difícil a medida em que sua solução apresenta uma explosão combinatória em relação ao número de entradas (cidades) em que o viajante precisa percorrer, tornando assim o cenário insuficiente em termos de tempo de execução e custo computacional. Para resolver esse e tantos outros problemas dessa classe, foram desenvolvidos algoritmos aproximativos para se aproximar de respostas ótimas em tempo de execução viável e custo também.

Para esse estudo, nós utilizamos de três tipos de métodos diferentes para a obtenção de comparações e por consequência o melhor resultado. Seriam esses:

- **Força bruta**
- **Algoritmo aproximativo (Vizinho mais próximo)**
- **Algoritmo aproximativo (Algoritmo de Inserção)**

O objetivo foi analisar o desempenho e a qualidade das soluções encontradas por cada abordagem, observando o equilíbrio entre **tempo de execução** e **proximidade em relação à solução ótima**.

**ESTRUTURA E ETAPAS DE SOLUÇÃO**

1. **Leitura da Matriz de Adjacência:**
   A partir de arquivos de entrada contendo as distâncias entre as cidades, foi gerada uma matriz de adjacência representando o grafo completo.

2. **Execução dos Algoritmos:**

**Força Bruta (Exato):** testa todas as permutações possíveis das cidades, garantindo o resultado ótimo. Sua complexidade temporal é O(n!).

**Vizinho Mais Próximo (Heurístico):** inicia em uma cidade e, a cada passo, escolhe o vértice mais próximo ainda não visitado até completar o ciclo Hamiltoniano.Essa abordagem resulta em alta eficiência e complexidade $O(n^2)$

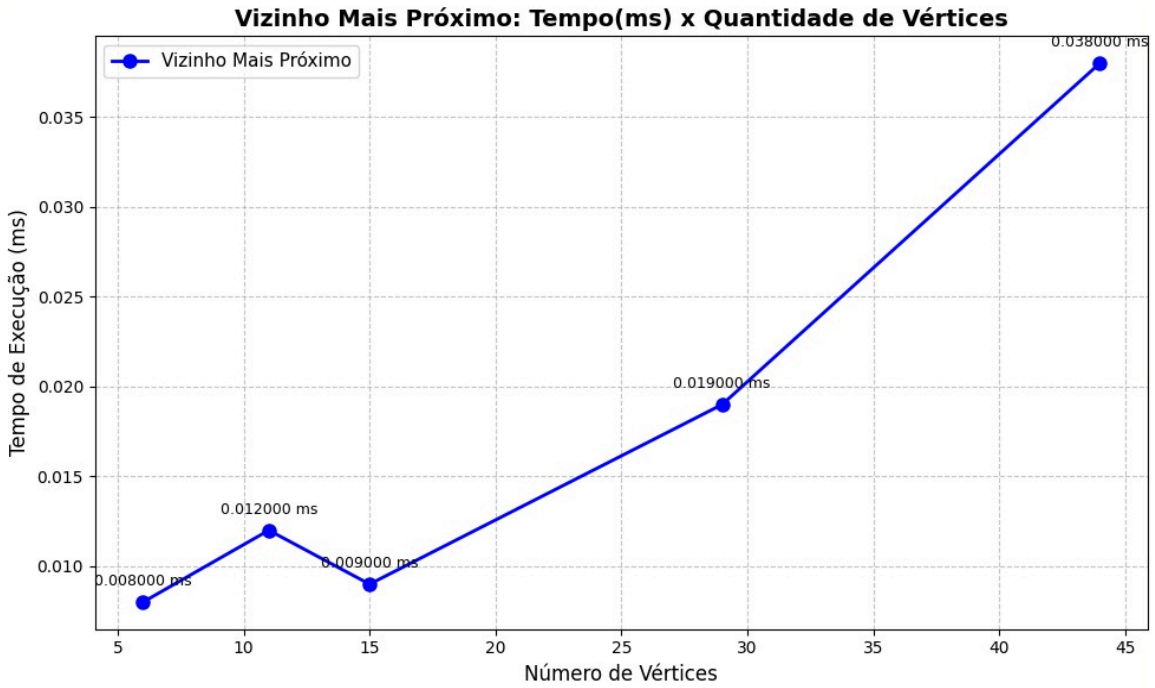
**Algoritmo de Inserção (Heurístico):** Inicialmente começa com dois vértices e vai adicionando progressivamente mais vértices que tenham um custo baixo, para atingir o menor custo. Sua complexidade também é $O(n^2)$, e seu desempenho tende a ser mais consistente e preciso do que o do Vizinho Mais Próximo.

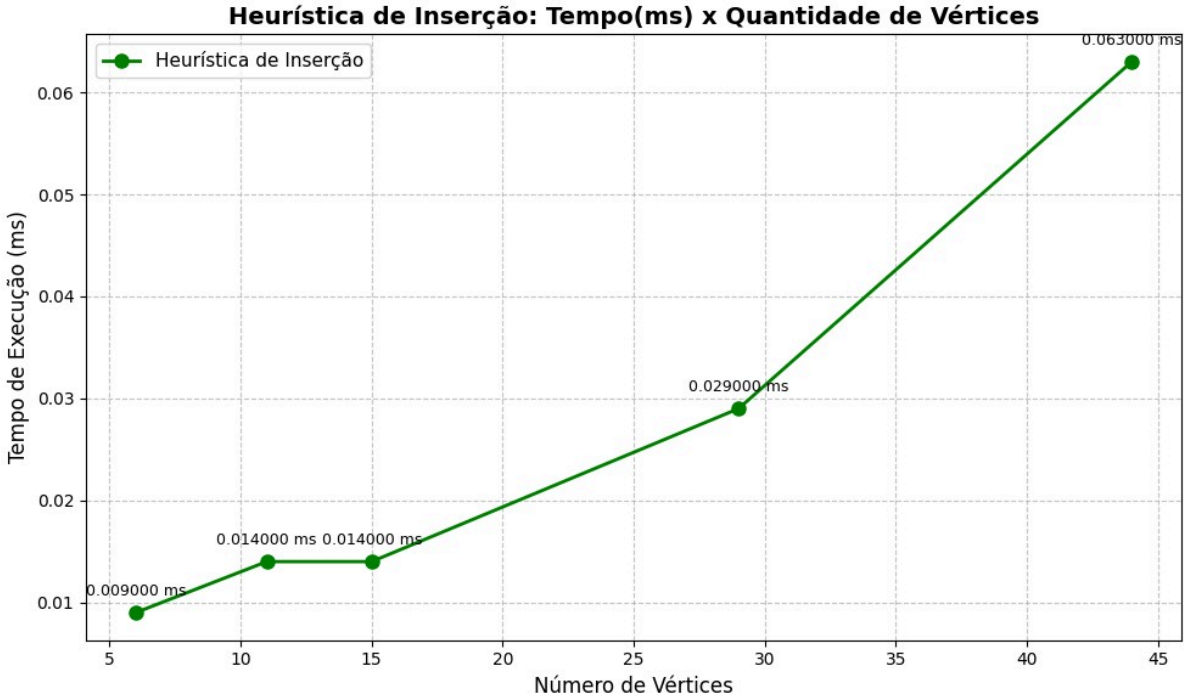
3. **Medição de Desempenho:**
   Foram medidos o **tempo de execução**, o **custo da solução obtida** e a **qualidade em relação ao ótimo**.

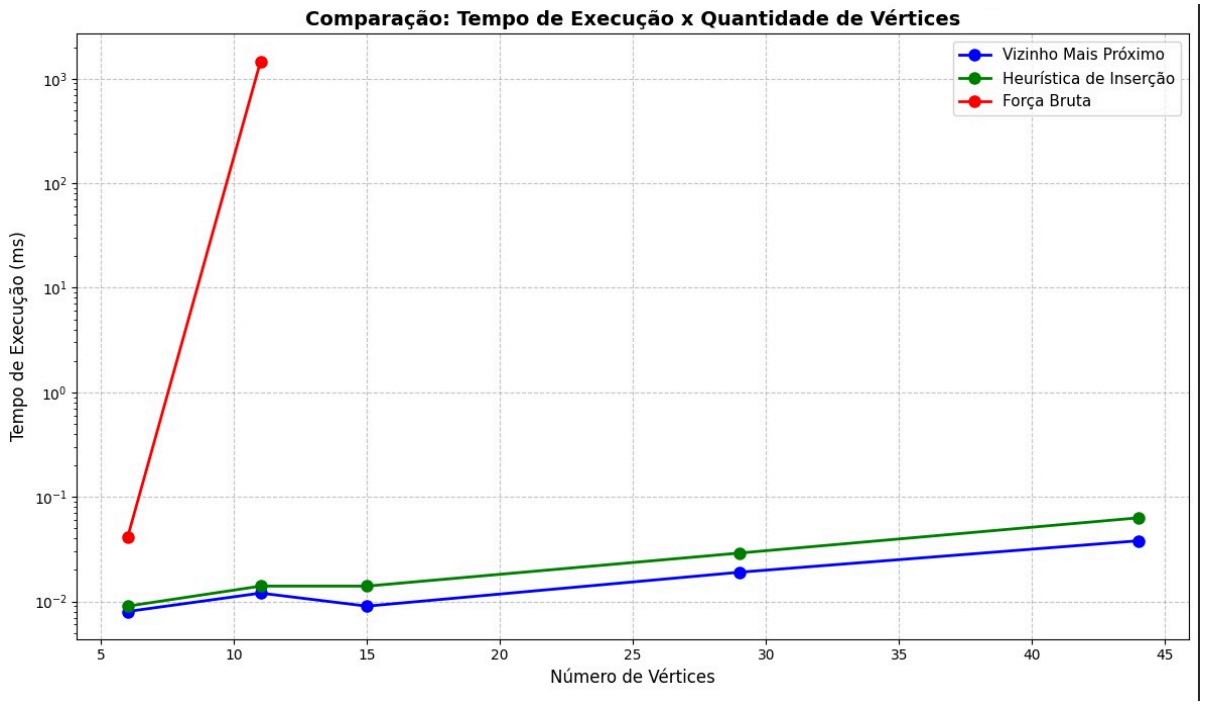
**RESULTADOS**

| Instância | Nº de Cidades | Algoritmo | Tempo de Execução (s) | Custo da Solução | Custo Ótimo | Qualidade (%) | Observação |
|---|---|---|---|---|---|---|---|
| tsp1_253.txt | 11 | Vizinho Mais Próximo | 0.000012 | 299 | 253 | 118.18 | 18% acima do ótimo |
| | | Inserção | 0.000014 | 253 | 253 | 100.00 | Igual ao ótimo |
| | | Força Bruta | 1.467202 | 253 | 253 | 100.00 | Achou o ótimo |
| | | | | | | | |
| tsp2_1248.txt | 6 | Vizinho Mais Próximo | 0.000008 | 1272 | 1248 | 101.92 | Muito próximo do ótimo |
| | | Inserção | 0.000009 | 1272 | 1248 | 101.92 | Mesmo desempenho |
| | | Força Bruta | 0.000041 | 1248 | 1248 | 100.00 | Achou o ótimo |
| | | | | | | | |
| tsp3_1194.txt | 15 | Vizinho Mais Próximo | 0.000009 | 1260 | 1194 | 105.53 | 5% acima do ótimo |
| | | Inserção | 0.000014 | 1351 | 1194 | 113.15 | 13% acima do ótimo |
| | | Força Bruta | — | — | 1194 | — | Inviável (muitas cidades) |
| | | | | | | | |
| tsp4_7013.txt | 44 | Vizinho Mais Próximo | 0.000038 | 10587 | 7013 | 150.96 | 51% acima do ótimo |
| | | Inserção | 0.000063 | 7527 | 7013 | 107.33 | 7% acima do ótimo |
| | | Força Bruta | — | — | 7013 | — | Inviável (muitas cidades) |
| | | | | | | | |
| tsp5_27603.txt | 29 | Vizinho Mais Próximo | 0.000019 | 36399 | 27603 | 131.87 | 31% acima do ótimo |
| | | Inserção | 0.000029 | 30094 | 27603 | 109.02 | 9% acima do ótimo |

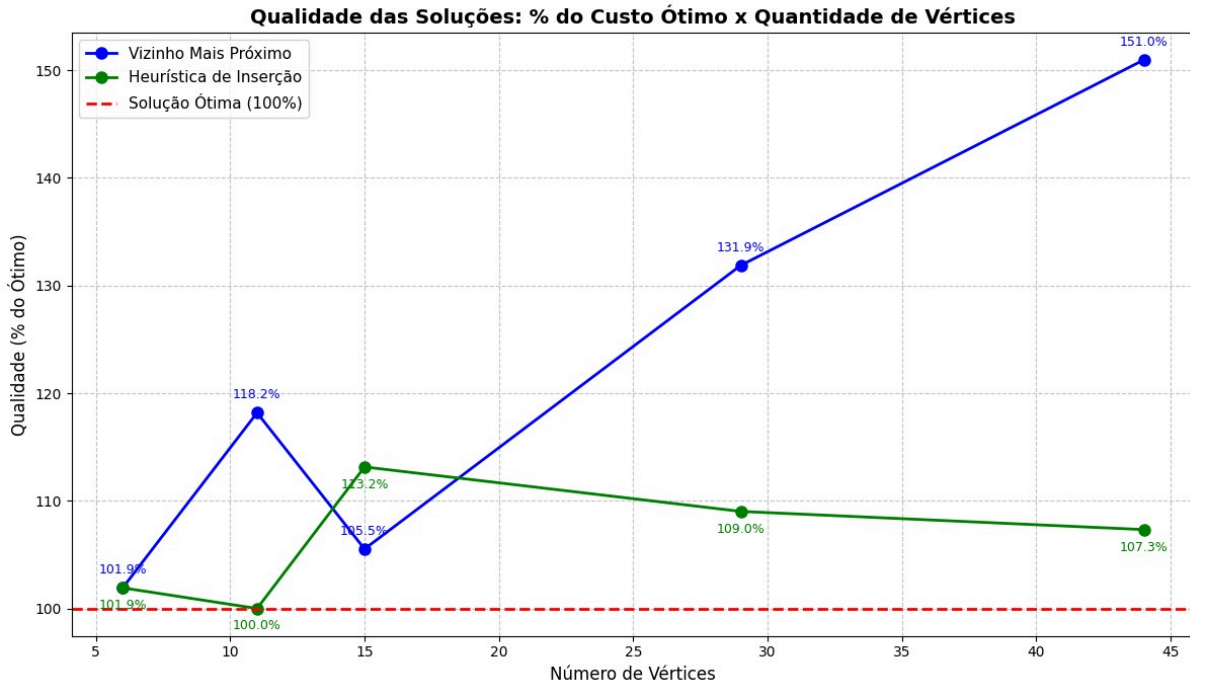
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | Força Bruta | — | — | 27603 | — | Inviável (muitas cidades) |






Os gráficos anteriores evidenciam a diferença de desempenho em tempo de execução entre os algoritmos. A Força Bruta apresenta crescimento exponencial, tornando-se inviável a partir de poucas cidades. As heurísticas, em contrapartida, exibem crescimento quase linear e tempos extremamente baixos, na ordem de microssegundos ou milissegundos, o que demonstra sua alta eficiência e escalabilidade.



O gráfico mostra o quanto as soluções heurísticas se afastam do custo ótimo, expresso em percentual. O custo ótimo é representado por uma linha vermelha fixa em 100%. O Vizinho Mais Próximo inicia próximo do ótimo para poucas cidades, mas chega a superar 150% do custo ótimo em instâncias maiores. A Heurística de Inserção, por sua vez, mantém estabilidade entre 100% e 110%, evidenciando um desempenho mais consistente e robusto. Esse comportamento ocorre porque o algoritmo Vizinho Mais Próximo toma decisões locais, escolhendo sempre o caminho mais curto imediato, o que pode gerar rotas ruins quando o número de cidades aumenta. Já a Heurística de Inserção considera melhor o impacto global das inserções, resultando em soluções mais estáveis e próximas do ótimo.

## ANÁLISE E DISCUSSÃO

Observa-se que o método de **Força Bruta** foi capaz de encontrar a solução ótima apenas em instâncias pequenas (como tsp2_1248), mas se tornou inviável à medida que o número de cidades aumentou. Isso confirma a explosão combinatória característica do TSP.

Já o algoritmo do **Vizinho Mais Próximo** apresentou tempos de execução praticamente nulos em todas as instâncias, uma clara vantagem em eficiência. No entanto, a qualidade da solução tende a decair conforme o número de cidades aumenta, chegando a mais de 150% do ótimo em instâncias grandes.

Finalizando, o **Algoritmo de Inserção** foi o que apresentou melhor desempenho, se mantendo em um bom tempo de execução para um resultado satisfatório em todos os casos. Em comparação ao vizinho mais próximo em que sua eficiência decaiu conforme o número de nós foi aumentando, este teve maior equilíbrio, conseguindo chegar em soluções boas o bastante independente do número de entradas.

Em síntese, os resultados reforçam o troca entre desempenho e qualidade típico dos problemas **NP-Difíceis**: enquanto métodos exatos garantem precisão ao custo de tempo exponencial, **as heurísticas oferecem soluções suficientemente boas com eficiência e escalabilidade, sendo mais adequadas para aplicações práticas do TSP.**

## CONCLUSÃO

Os resultados obtidos confirmam a relação inversa entre tempo de execução e qualidade da solução nos métodos para o TSP. Enquanto o algoritmo exato é inviável para grandes instâncias, o Vizinho Mais Próximo e a Inserção oferecem soluções rápidas e suficientemente boas para aplicações práticas, especialmente quando o tempo de resposta é mais importante que a precisão absoluta.

Assim, a escolha do algoritmo depende diretamente do contexto de uso:

**Exato:** útil para instâncias pequenas ou quando a precisão é crítica.

**Aproximativo:** indicado para grandes volumes de dados ou aplicações em tempo real.

## EXECUÇÃO DO CÓDIGO

Clone o repositório ou abra o código-fonte no ambiente de desenvolvimento.

https://github.com/Celes16/caixeiro_viajante-.git

Adicione os arquivos de instância (tsp1_253.txt, tsp2_1248.txt, etc.) na mesma pasta do programa.

No código principal, especifique qual arquivo deseja testar:

```
files = ["tsp1_253.txt", "tsp2_1248.txt", "tsp3_1194.txt", "tsp4_7013.txt", "tsp5_27603.txt"]
solve_tsp(files[0)
```

Execute o programa com:

```
python main.py
```