# Trabalho 2: Avaliação de Algoritmos para o Problema do Caixeiro Viajante

**Pedro Augusto Torres Bento - 2022104352**
**Yan Aquino Amorim - 2022043221**

[1]Universidade Federal de Minas Gerais (UFMG)

{pedro.bento,yanaquino}@dcc.ufmg.br

***Abstract.*** *Este trabalho apresenta a implementação e análise de três algoritmos para resolver o Problema do Caixeiro Viajante (TSP): branch-and-bound, twice-around-the-tree e o algoritmo de Christofides. Descrevemos a metodologia empregada, escolhas de estruturas de dados, análise de desempenho em termos de tempo e qualidade das soluções, e comparamos os resultados obtidos. Observamos que o branch-and-bound não convergiu para nenhuma instância em 30 minutos, enquanto as heurísticas aproximadas forneceram soluções em tempo razoável, com diferentes margens de erro em relação ao ótimo.*

## 1. Introdução

O Problema do Caixeiro Viajante (TSP) é um dos problemas combinatórios mais estudados em teoria da computação e otimização. Ele busca encontrar o menor percurso que visita cada cidade exatamente uma vez e retorna à cidade de origem. Este trabalho avalia o desempenho de abordagens exatas e heurísticas para resolver o TSP em instâncias geométricas.

## 2. Metodologia

Para abordar o Problema do Caixeiro Viajante (TSP), optamos por implementar três algoritmos distintos, cada um com abordagens e características próprias, visando comparar suas eficiências e qualidade das soluções obtidas.

1. **Branch-and-Bound (BB):** Trata-se de um algoritmo exato que explora sistematicamente todo o espaço de soluções possíveis, podando ramos que não podem levar a uma solução melhor do que a já encontrada. Utilizamos uma estratégia *best-first*, na qual priorizamos a expansão de subproblemas com menor custo parcial combinado a uma estimativa de custo futuro (obtida pela função limite). Embora essa abordagem garanta a optimalidade, notamos que, para as instâncias testadas, o algoritmo não conseguiu convergir para uma solução dentro do limite de 30 minutos, evidenciando as dificuldades computacionais inerentes a problemas NP-difíceis conforme o tamanho da instância cresce [Martello and Toth 1990, Lawler and Wood 1966, Pisinger 1995].
2. **Twice-Around-the-Tree (TAT):** Essa heurística constrói primeiro uma Árvore Geradora Mínima (MST) do grafo completo representando as cidades e suas distâncias. Em seguida, duplica todas as arestas da MST para obter um multigrafo Euleriano, do qual é possível extrair um circuito Euleriano que visita cada

aresta exatamente uma vez. Por fim, o circuito Euleriano é simplificado em um ciclo Hamiltoniano, removendo repetições de vértices enquanto se mantêm a ordem relativa de visita, resultando em uma solução aproximada para o TSP. Essa abordagem tem a vantagem de ser computacionalmente eficiente, embora não garanta a optimalidade [Rosenkrantz et al. 1977].

3. **Algoritmo de Christofides:** Como uma melhoria sobre o TAT, o algoritmo de Christofides adiciona um passo intermediário crucial. Após a construção da MST, identifica-se o conjunto de vértices de grau ímpar. Em seguida, computa-se um emparelhamento perfeito mínimo entre esses vértices, adicionando essas arestas à MST para formar um multigrafo Euleriano. Esse procedimento assegura que todas as vértices terão grau par, o que possibilita a construção de um circuito Euleriano. Posteriormente, como no TAT, aplica-se a técnica de atalho para obter um ciclo Hamiltoniano. O algoritmo de Christofides é notável por garantir um fator de aproximação de 1.5 da solução ótima no caso de instâncias com a propriedade da desigualdade triangular, oferecendo um compromisso entre qualidade da solução e eficiência computacional [Christofides 1976, Cornuéjols and Pulleyblank 1985].

### 2.1. Escolhas de Implementação

Diversas decisões de projeto foram tomadas ao longo da implementação para otimizar a performance e facilitar a manutenção do código:

- **Representação de Dados:** Os grafos foram representados como completos e ponderados, onde cada vértice corresponde a uma cidade e as arestas são ponderadas pela distância euclidiana entre pares de cidades. Essa escolha simplifica o acesso às distâncias durante a execução dos algoritmos e assegura que todas as soluções consideradas são viáveis no contexto do TSP.
- **Estruturas de Dados:** Para o algoritmo Branch-and-Bound, utilizamos matrizes de distância para acessar rapidamente os custos entre cidades e filas de prioridade para gerenciar os nós na busca *best-first*. Nas heurísticas (TAT e Christofides), a estrutura escolhida foi a lista de adjacência, facilitando a manipulação de grafos esparsos resultantes das MSTs e dos emparelhamentos. A biblioteca `NetworkX` foi empregada para operações complexas em grafos, como construção de árvores geradoras mínimas, emparelhamento perfeito e circuitos Eulerianos, beneficiando-se de implementações otimizadas dessas operações.

## 3. Experimentos e Resultados

### 3.1. Experimentos

Os experimentos foram conduzidos utilizando todas as instâncias de arquivos `.tsp` disponíveis no repositório TSPLIB. Estes arquivos contêm coordenadas das cidades, permitindo a construção completa dos grafos necessários para testar os algoritmos implementados. Cada instância foi processada para extrair as coordenadas das cidades, calcular a matriz de distâncias correspondente e, em seguida, aplicar os algoritmos Twice-Around-the-Tree (TAT) e Christofides dentro de um limite de tempo de 30 minutos. A seleção das instâncias da TSPLIB proporcionou uma ampla variedade de tamanhos e complexidades de problemas, desde pequenos conjuntos de cidades até instâncias que envolvem milhares de nós, oferecendo um panorama robusto sobre o desempenho e a escalabilidade dos algoritmos.

### 3.2. Resultados

Para quase todas as instâncias analisadas, o algoritmo de Christofides apresentou desempenho superior ao Twice-Around-the-Tree, conforme evidenciado na Figura 1. Para visualizar essas diferenças de forma mais detalhada, selecionamos cinco instâncias aleatórias do repositório TSPLIB. Os resultados obtidos com os algoritmos para essas cinco instâncias foram compilados em uma tabela comparativa (Tabela **??**), demonstrando os custos das rotas encontradas e os tempos de execução para cada caso específico do TSP. A seguir, apresentamos a tabela para referência:

**Table 1. Resultados atualizados de TAT e Christofides para as instâncias do TSP.**

| Instância | Custo TAT | Tempo TAT | Custo Chris | Tempo Chris | Custo Ótimo |
|---|---|---|---|---|---|
| pr439 | 146.236,089 | 0,22524 | 116.048,597 | 1,29763 | 109.704,369 |
| rd100 | 10.089,162 | 0,01264 | 9.068,681 | 0,09328 | 8.199,645 |
| rl5934 | 83.8575,313 | 72,35491 | 588.199,693 | 480,40944 | 584.330,629 |
| pcb442 | 71.480,903 | 0,21342 | 53.628,251 | 2,36172 | 50.903,708 |
| u2319 | 362.982,360 | 6,40118 | 273.369,244 | 318,53449 | 229.386,864 |

A análise dos resultados evidencia que o algoritmo de Christofides consistentemente supera o Twice-Around-the-Tree em termos de qualidade das soluções. Em cada instância testada, os custos das rotas geradas por Christofides ficaram mais próximos do ótimo conhecido, com diferenças de até 10% em relação ao custo ideal. Essa proximidade com a solução ótima é atribuída ao uso do emparelhamento perfeito mínimo e à garantia teórica do fator de aproximação de 1.5 proporcionado por Christofides. Embora isso requeira tempos de execução mais longos em comparação com o TAT, a qualidade das rotas é significativamente melhorada, especialmente em instâncias de maior dimensão.

Por outro lado, o algoritmo Twice-Around-the-Tree demonstrou tempos de execução notavelmente mais rápidos, especialmente em casos com menos de 1000 cidades. Essa eficiência temporal torna o TAT uma escolha apropriada para aplicações em que a velocidade é crítica, ainda que isso implique em aceitar soluções que se afastem mais do ótimo.

A Figura 2 ilustra a relação entre o tempo de execução e o número de cidades para os algoritmos TAT e Christofides. Observa-se que ambos os algoritmos apresentam aumento de tempo conforme o número de cidades cresce, mas a elevação no tempo de execução é mais acentuada para Christofides. Esse comportamento se deve à maior complexidade computacional de Christofides, especialmente nas etapas de identificação de vértices ímpares, cálculo do emparelhamento perfeito e construção de circuitos Eulerianos. Em contraste, o TAT, sendo menos complexo, apresenta um crescimento mais moderado no tempo de execução, destacando a diferença nas tendências de desempenho à medida que as instâncias se tornam maiores.

Os resultados destacam um claro trade-off entre a qualidade das soluções e o tempo de execução. Em aplicações onde o tempo de resposta é fundamental, como em sistemas de tempo real, o TAT pode ser preferível devido à sua agilidade. Em contraste, para cenários em que a precisão da rota é crucial, como no planejamento logístico detalhado, o algoritmo de Christofides se mostra como a alternativa mais robusta, apesar do

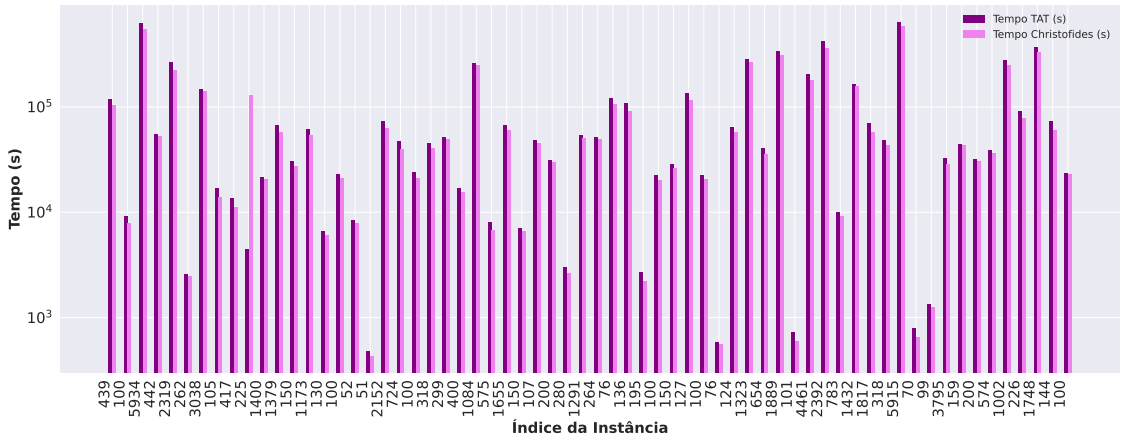


**Figure 1. Comparação dos tempos de execução entre os algoritmos TAT e Christofides para diferentes instâncias. A fim de facilitar a visualização, o eixo-y (tempo) está em escala logarítmica.**

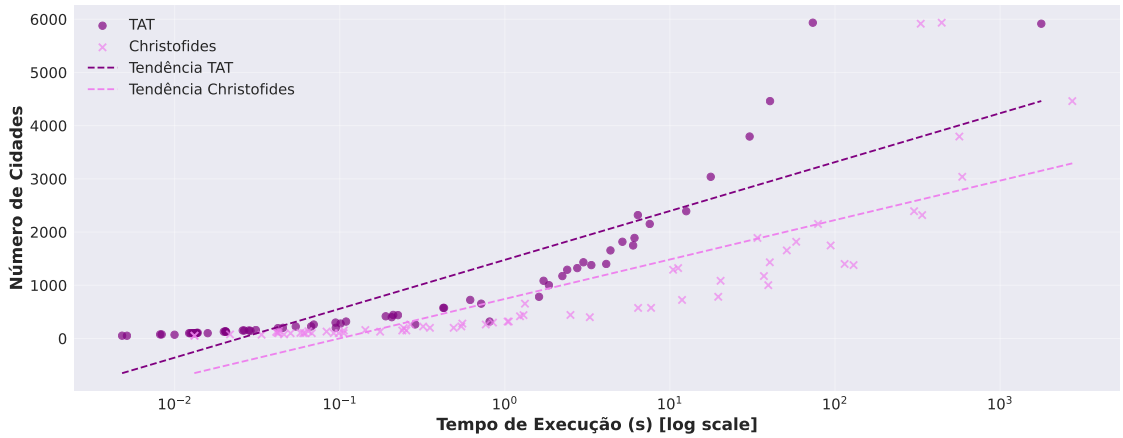


**Figure 2. Relação entre tempo de execução (escala logarítmica) e número de cidades para os algoritmos TAT e Christofides, destacando suas tendências de desempenho.**

maior tempo computacional envolvido.

Além disso, observou-se que o desempenho de ambos os algoritmos é mais previsível em instâncias menores. À medida que o número de cidades aumenta, particularmente em instâncias com mais de 2000 cidades, a diferença relativa entre os erros dos algoritmos também aumenta. Isso sugere que o fator de aproximação teórico do algoritmo de Christofides se materializa de forma mais consistente em problemas maiores, oferecendo soluções de qualidade superior de maneira confiável, mesmo diante do desafio crescente da complexidade computacional.

## 4. Conclusões

Para concluir, observa-se que, embora o algoritmo de Christofides acarrete uma ligeira deterioração no tempo de execução comparado ao Twice-Around-the-Tree, ele se revela como uma heurística mais robusta, capaz de fornecer soluções de qualidade significativamente superior. Essa constatação confirma o fator aproximativo previsto teoricamente,

demonstrando que a heurística de Christofides é mais poderosa ao oferecer rotas que se aproximam mais do ótimo conhecido.

Adicionalmente, os experimentos realizados para obtenção de soluções exatas por meio do branch-and-bound evidenciaram que o tempo requerido cresce exponencialmente, atingindo ordens de magnitude muito superiores às observadas nas heurísticas. Dessa forma, torna-se essencial ponderar a disponibilidade de tempo e recursos computacionais em relação à qualidade das soluções obtidas ao escolher entre esses métodos, haja vista a relação inversamente proporcional entre esses fatores.

Em cenários onde soluções rápidas são imprescindíveis, como em aplicativos de navegação GPS, a preferência recai sobre heurísticas como TAT ou Christofides, pois estas oferecem um bom compromisso entre eficiência e precisão em tempo reduzido. Por outro lado, situações que demandam a solução ótima exata requerem não apenas a disposição de recursos computacionais mais abundantes, mas também a disponibilidade de tempo suficiente para a realização de cálculos extensivos.

## References


Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem. *Technical report, GSIA, Carnegie Mellon University*.

Cornuéjols, G. and Pulleyblank, W. R. (1985). *Combinatorial optimization: Packing and covering*. Springer.

Lawler, E. L. and Wood, D. E. (1966). Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719.

Martello, S. and Toth, P. (1990). *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc.

Pisinger, D. (1995). An exact algorithm for large multiple knapsack problems. *European Journal of Operational Research*, 83(3):394–410.

Rosenkrantz, D. J., Stearns, R. E., and Lewis, P. M. (1977). An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581.