Análise de Algoritmos para o Problema do Caixeiro Viajante

Arthur Linhares Madureira[1]

[1]Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

arthurlm1234@gmail.com

Abstract. *This paper presents a detailed analysis of algorithms for solving the Traveling Salesman Problem (TSP), utilizing techniques such as Branch and Bound, Christofides, and Twice Around the Tree. The analysis focuses on the efficiency of the algorithms in terms of execution time, memory usage, and the accuracy of the solutions found.*

Resumo. *Este trabalho apresenta uma análise detalhada de algoritmos para resolver o Problema do Caixeiro Viajante (TSP), utilizando técnicas como Branch and Bound, Christofides e Twice Around the Tree. A análise foca na eficiência dos algoritmos em termos de tempo de execução, uso de memória e precisão das soluções encontradas.*

1. Introdução

O Problema do Caixeiro Viajante (TSP) é um desafio clássico em otimização combinatória, fundamental em diversas aplicações práticas, como logística, planejamento de rotas e redes de telecomunicações. Este problema consiste em encontrar a menor rota possível que passa por uma série de cidades e retorna à cidade de origem, sendo notável pela sua natureza NP-difícil. O estudo de algoritmos para resolver o TSP é crucial para desenvolver soluções eficientes para problemas do mundo real que envolvem otimização de rotas.

Este trabalho implementa e analisa diferentes algoritmos para resolver o TSP, incluindo Branch and Bound, Christofides e Twice Around the Tree. A análise concentra-se na eficiência e precisão das soluções, considerando as características específicas de cada algoritmo, como complexidade de tempo, uso de memória e a qualidade das soluções aproximadas. Com uma comparação detalhada dessas abordagens, buscamos identificar as estratégias mais adequadas para diferentes tamanhos e configurações de problemas.

2. Descrição das Implementações

2.1. Estruturas de Dados

- **Grafos**:
  - *Uso*: Representam as cidades e as conexões entre elas, com cada cidade sendo um nó e cada aresta representando uma rota.
  - *Implementação*: Utilização da biblioteca `networkx` para manipulação e análise de grafos.
  - *Motivo da Escolha*: Capacidade natural de representar as relações entre as cidades no TSP.

2.2. Algoritmos

Esta seção detalha os três algoritmos considerados para o problema do caixeiro viajante (TSP), enfatizando suas características principais e complexidades assintóticas.

- **Branch and Bound (B&B)**: Este algoritmo explora o espaço de solução de maneira sistemática, podando ramos que não vão resultar em uma solução ótima. É um método exato que tem uma complexidade assintótica de $O((n)!)$, tornando-o adequado apenas para instâncias menores devido ao aumento exponencial do número de permutações com o aumento do tamanho do problema.
- **Christofides**: Utiliza uma abordagem heurística que fornece uma solução com um custo no máximo 1,5 vezes o ótimo. A complexidade assintótica do algoritmo é dominada pela etapa de encontrar um emparelhamento mínimo, que é $O(n^3)$, tornando-o apropriado para instâncias de tamanho moderado.
- **Twice Around the Tree (TAT)**: Este método gera uma solução aproximada para o TSP. Ele começa construindo uma árvore geradora mínima (MST) do grafo, cuja complexidade é $O(E \log V)$, seguido por uma busca em profundidade (DFS) na MST, com complexidade $O(V + E)$. Finalmente, o cálculo do peso do caminho e a construção do caminho são realizados em $O(V)$. Assim, a complexidade total do TAT é dominada pela etapa de construção da MST, que é $O(E \log V)$. Este algoritmo é útil para obter uma solução inicial rápida ou para instâncias muito grandes.

Cada algoritmo apresenta um equilíbrio diferente entre a precisão da solução e o tempo de execução, o que os torna mais ou menos adequados dependendo do tamanho e das características da instância do problema do TSP em questão.

2.3. Escolha de Best-First

- *Implementação*: Adoção de uma abordagem de best-first search, priorizando nós com menor limite estimado.
- *Razão da Escolha*: Garante que o algoritmo sempre siga o caminho mais promissor em termos de custo.

3. Experimentos e Resultados

Os experimentos foram conduzidos utilizando diversos datasets do Problema do Caixeiro Viajante (TSP). Cada algoritmo foi avaliado com base em três critérios principais: tempo de execução, uso de memória e precisão da solução. As métricas específicas para cada critério foram definidas e implementadas no código do projeto.

3.1. Limitações do Branch and Bound

Foi observado que o algoritmo de Branch and Bound não foi viável nem mesmo para o menor dataset (n=50) . Esta limitação destaca a complexidade computacional do algoritmo.

3.2. Análise de Tempo de Execução e Uso de Memória

Foram gerados gráficos para ilustrar o desempenho dos algoritmos aproximativos em termos de tempo de execução e uso de memória. Estes gráficos proporcionam uma comparação visual clara entre os algoritmos ‘twiceAroundTheTreeTSP‘ e ‘christofidesTSP‘ sob diferentes condições de teste.

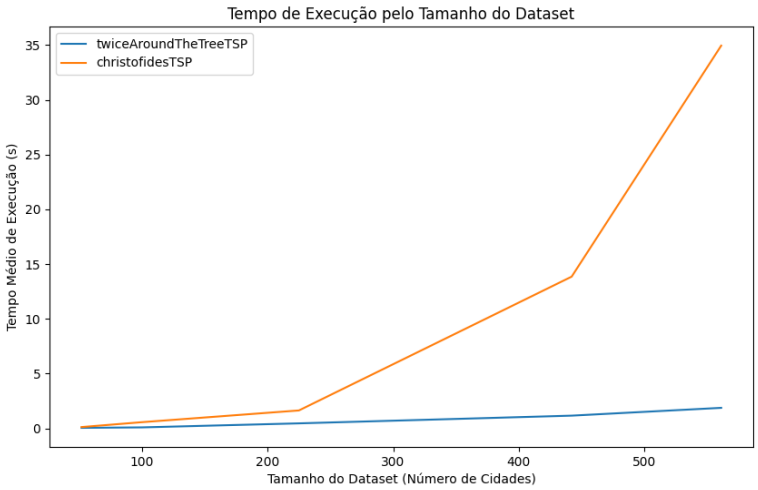

Figure 1. Gráfico do Tempo de Execução dos Algoritmos

No que diz respeito ao uso de memória, ambos os algoritmos mostraram um aumento quase linear com o crescimento do tamanho do dataset. Os dois apresentam uso de memória quase idênticos.

Em relação ao tempo de execução, observa-se que o ‘twiceAroundTheTreeTSP‘ mantém um tempo de execução estável e baixo, enquanto o ‘christofidesTSP‘ exibe um aumento expressivo no tempo necessário para completar o cálculo com o aumento do número de cidades. Isso indica que, embora ambos os algoritmos possam ser viáveis para datasets menores, o ‘twiceAroundTheTreeTSP‘ é mais eficiente para datasets maiores.

3.3. Desvio Médio dos Algoritmos Aproximativos

Os algoritmos aproximativos Christofides e Twice Around the Tree foram avaliados quanto ao seu desvio médio em relação à solução ótima. Este desvio oferece uma medida da precisão de cada algoritmo.

- **Twice Around the Tree**: Desvio médio de 34.70%.
- **Christofides**: Desvio médio de 11.76%.

4. Conclusão

A análise detalhada dos algoritmos Branch and Bound, Christofides e Twice Around the Tree para resolver o Problema do Caixeiro Viajante revelou insights significativos sobre a aplicabilidade e eficiência de cada método em diferentes contextos. O algoritmo Branch and Bound, apesar de ser um método exato, demonstrou limitações significativas em termos de viabilidade prática devido à sua alta complexidade computacional. Esta observação reforça a necessidade de abordagens heurísticas ou aproximativas para o TSP em cenários do mundo real, onde o tempo de execução e a eficiência de memória são cruciais.

Os algoritmos Christofides e Twice Around the Tree, ambos métodos aproximativos, mostraram-se mais adequados para instâncias maiores do TSP. O algoritmo Christofides destacou-se pela sua precisão, com um desvio médio significativamente

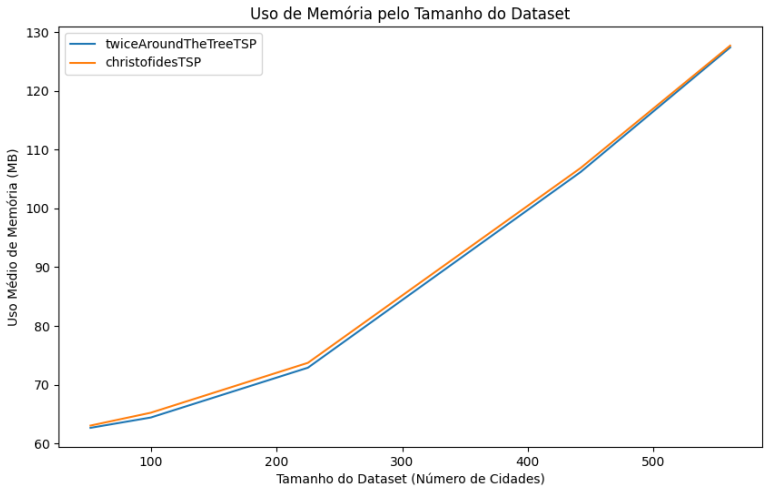

Figure 2. Gráfico do Uso de Memória dos Algoritmos

menor em relação à solução ótima, tornando-o uma escolha preferível quando a precisão é mais crítica do que o tempo de execução. Por outro lado, o Twice Around the Tree provou ser extremamente eficiente em termos de tempo de execução, embora com um desvio maior da solução ótima. Esta característica o torna uma opção viável para instâncias muito grandes do TSP, onde uma solução rápida é mais valorizada do que a precisão absoluta.

Os resultados também enfatizam a importância de escolher o algoritmo apropriado com base no tamanho e nas características específicas da instância do TSP. Enquanto o Branch and Bound pode ser adequado para instâncias muito pequenas, os algoritmos aproximativos são claramente mais viáveis para a maioria das aplicações práticas. Além disso, o uso de estruturas de dados eficientes, como grafos e técnicas de busca best-first, demonstrou ser crucial para o desempenho dos algoritmos.

Em conclusão, este estudo oferece uma visão valiosa sobre a seleção de algoritmos para o Problema do Caixeiro Viajante, destacando a necessidade de um equilíbrio entre precisão, tempo de execução e uso de memória.

References

[1] Wikipedia. *Travelling salesman problem — Wikipedia, The Free Encyclopedia*. [Online; accessed 4-December-2023]. `https://en.wikipedia.org/wiki/Travelling_salesman_problem`