Trabalho Prático 2
Algoritmos II

Enzo Pinheiro Pierazolli
2021031564

Universidade Federal de Minas Gerais

[1]

Abstract. *This paper addresses the practical implementation of algorithms to solve the Traveling Salesman Problem (TSP), focusing on instances with Euclidean distances. Three approaches were evaluated: Branch and Bound, Christofides, and Twice Around the Tree. The data structure employed is based on graphs, implemented using the NetworkX library. Branch and Bound utilize the Best First Search strategy to explore the solution space in pursuit of more precise results. Both the Christofides and Twice Around The Tree algorithms seek approximate solutions. Therefore, each algorithm balances precision and execution time, making them suitable for different objectives and/or sizes of TSP instances.*

Resumo. *Este trabalho aborda a implementação prática de algoritmos para resolver o Problema do Caixeiro Viajante (TSP), com foco em instâncias com distâncias euclidianas. Três abordagens foram avaliadas: Branch and Bound, Christofides e Twice Around the Tree. A estrutura de dados utilizada é baseada em grafos, implementada com a biblioteca NetworkX. O Branch and Bound emprega a estratégia Best First Search para explorar o espaço de solução em busca de resultados mais precisos. O algoritmo de Christofides e o de Twice Around The Tree realiza a busca por soluções aproximadas. Portanto, cada algoritmo equilibra precisão e tempo de execução, sendo adequado para diferentes objetivos e/ou tamanhos de instâncias do TSP.*

1. Introdução

Este trabalho aborda os aspectos práticos da implementação de algoritmos para a resolução do Problema do Caixeiro Viajante (TSP). As implementações avaliadas incluem um método exato baseado em branch-and-bound, juntamente com duas abordagens aproximadas: twice-around-the-tree e o algoritmo de Christofides, todos adaptados para o TSP euclidiano.

2. Implementações

2.1. Estrutura de dados

Os algoritmos empregam a estrutura de dados de grafos, onde os nós correspondem às cidades e as arestas denotam as distâncias euclidianas entre essas cidades. Para a eficiente manipulação desses grafos, foi utilizada a biblioteca NetworkX. Essa escolha se baseia na capacidade abrangente da NetworkX de oferecer um conjunto útil de ferramentas e algoritmos para análise e manipulação de grafos, facilitando a implementação e o desenvolvimento do algoritmo para o Problema do Caixeiro Viajante (TSP).

2.2. Branch and Bound

Este algoritmo gera uma solução aproximada para o TSP, ao explorar sistematicamente o espaço de solução, podando ramos que não levam a uma solução ótima. Trata-se de um método exato com uma complexidade assintótica de O(n!), tornando-o adequado apenas para instâncias menores, devido ao significativo aumento de permutações com o tamanho do problema.

Para a implementação, é válido destacar a escolha da **Best First Search**. A motivação dessa escolha partiu de que essa estratégia, ao focar nos caminhos mais promissores em termos de custo, proporciona eficiência na utilização de recursos, convergência rápida para soluções de alta qualidade e adaptabilidade para o problema em questão(TSP), bem como a percepção de um melhor funcionamento para as instâncias testadas(apesar de terem excedidos o tempo limite proposto) em relação à abordagem com Depth First Search.

2.3. Twice Around the Tree

Este método também gera uma solução aproximada para o TSP. O funcionamento do algoritmo se inicia com a construção de uma árvore geradora mínima (MST) do grafo, com complexidade O(F log V), seguida por uma busca em profundidade na MST, O(V + E). JAlem de custos lineares para a construção do caminho e o cálculo de seu respectivo peso. A complexidade desse algoritmo é O(E*log(V))dominada, especialmente, pela construção da MST. Apesar de obter soluções significativamente menos precisas, esse algoritmo foi capaz de fornecer soluções de forma extremamente ágil, comparada ao algoritmo de Branch and Bound para as instâncias avaliadas.

Cada algoritmo apresenta um equilíbrio único entre precisão da solução e tempo de execução, tornando-os mais ou menos adequados dependendo do tamanho e das características específicas da instância do problema do TSP.

2.4. Christofides

O funcionamento do algoritmo, resumidamente, consiste em encontrar um emparelhamento mínimo nos nós de grau ímpar, atualiza a MST(criada inicialmente) e calcula um caminho Euleriano, eliminando arestas repetidas. O peso do caminho é calculado, resultando em uma solução aproximada para o TSP. A complexidade assintótica é dominada pela etapa de emparelhamento mínimo, sendo $O(n^3)$, tornando-a adequada para instâncias de tamanho moderado. Possui um fator aproximativo de 3/2 em relação ao ótimo, produzindo, portanto, resultados menos precisos do que o Branch and Bound, porém encontra soluções de forma significativamente mais rápida para as instâncias calculadas.

3. Experimentos e Resultados

Os algoritmos foram avaliados em relação a solução obtida(precisão), tempo de execução e o uso da memória. Além disso, é válido destacar que a implementação do **Branch and Bound** excedeu o tempo limite de execução(30 minutos) para as instâncias avaliadas e, portanto, não foi incluso nas comparações, apesar de ter alcançado as soluções ótimas.

3.1. Análise do Tempo de Execução

Foi gerado um gráfico para melhor visualização do tempo gasto por algoritmo em função do tamanho da instância executada.

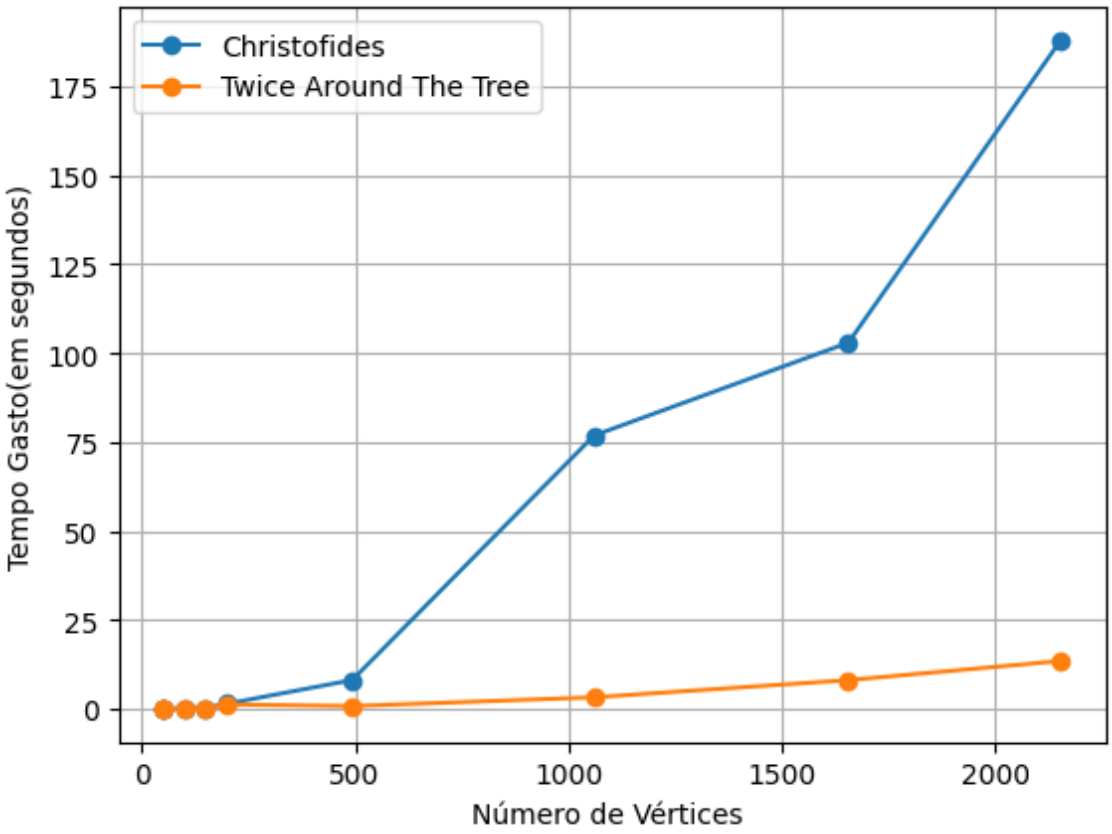

Figure 1. Tempo gasto por algoritmo em função do número de cidades(vértices)

É possível observar que, com pequenas instâncias não há uma diferença significativa entre o tempo gasto por ambos os algoritmos. Portanto, ao escalar o número de vértices(cidades) do grafo, o algortimo Twice Around the Tree apresenta um tempo de execução significativamente menor do que em relação ao Christofides. Quanto maior a instância executada, maior tende a ser a diferença entre o tempo de execução entre ambos os algoritmos.

3.2. Análise de Precisão

Foi gerado um gráfico que nos permite visualizar o quão distante a solução encontrada pelos algoritmos está da solução ótima

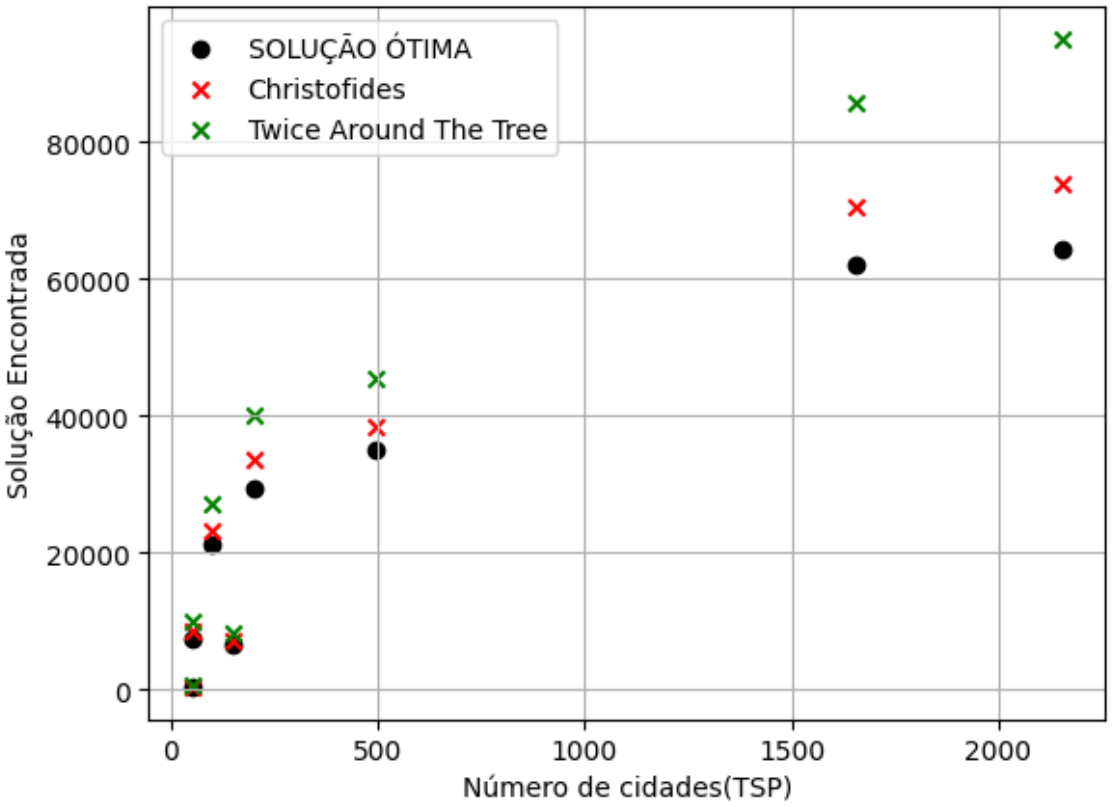

Figure 2. Solução encontrada por algoritmo em comparação a solução ótima

É possível observar que o algoritmos de Christofides tende a apresentar uma solução mais próxima da ótima do que o algoritmo Twice Around The Tree. Ao calcular o desvio médio percentual, para as intâncias executadas, em relação a solução ótima, os resultados obtidos foram de 12.27% para o Christofides e 36.36% para o Twice Around The Tree.

3.3. Análise do uso de Memória

Por fim, foi feita a análise do uso de memória dos algoritmos e, para isso, foi utilizada a biblioteca memory_profiler, que registrou os seguintes dados para os algoritmos:

Número de vértices	Uso de memória inicial(MiB)	Incremento(MiB)	Uso de memória final(MiB)
51	89	0.2	89.2
52	89.7	0.2	89.9
100	91.8	0.7	92.5
200	100	3	103
493	144.8	1.4	146.2
2152	1075.2	42.4	1117.6

Figure 3. Uso de memória - Christofides

Número de vértices	Uso de memória inicial(MiB)	Incremento(MiB)	Uso de memória final(MiB)
51	88.8	0.1	88.9
52	89.4	0.1	89.5
100	90.9	0.5	91.4
200	99.1	1.9	101
493	144.4	1.2	145.6
2152	1067.7	25.6	1093.3

Figure 4. Uso de memória - Twice Around The Tree

Sendo o ”Incremento” a variação do uso da memória após a execução do algoritmo, no geral o algoritmo de Christofides possuiu um uso da memória ligeiramente maior e, portanto, o algoritmo Twice Around The Tree obteve uma ligeira vantagem em relação ao uso de memória. Porém, para pequenas instâncias não houve uma diferença significativa. Além disso, vale destacar que por exceder o tempo limite, os valores do uso da memória da implementação do Branch and Bound não foram registrados.

4. Conclusão

Neste trabalho, exploramos a implementação prática de algoritmos para a resolução do Problema do Caixeiro Viajante (TSP) em instâncias com distâncias euclidianas. Três abordagens foram implementadas: Branch and Bound, Christofides e Twice Around the Tree, cada uma apresentando características únicas em termos de precisão e eficiência computacional.

O Branch and Bound, apesar de sua complexidade assintótica elevada, mostrou-se capaz de encontrar soluções ótimas para as instâncias avaliadas. No entanto, a estratégia de busca Best First Search, embora promissora, enfrentou desafios em instâncias maiores, excedendo o tempo limite proposto e impossibilitando a obtenção dos valores do uso da memória.

O Twice Around the Tree e o algoritmo de Christofides, por outro lado, demonstraram eficácia na obtenção de soluções aproximadas. O primeiro, com sua abordagem de construção de uma árvore geradora mínima e busca em profundidade, revelou-se especialmente rápido, embora menos preciso. Enquanto isso, o algoritmo de Christofides, com um fator aproximativo de 3/2 em relação ao ótimo, obteve soluções mais precisas,

sacrificando um pouco o tempo de execução e apresentando um uso ligeiramente maior de memória.

Analisando o desempenho, observamos que o Twice Around The Tree apresentou tempos de execução significativamente menores em comparação ao Christofides para instâncias maiores, além de uma ligeira vantagem em relação ao uso de memória. No entanto, o Christofides destacou-se ao produzir soluções mais próximas do ótimo.

Em resumo, cada algoritmo apresenta trade-offs entre precisão, eficiência e uso de memória, sendo adequado para diferentes tamanhos e naturezas de instâncias do TSP. A escolha do algoritmo dependerá dos requisitos específicos de cada aplicação, considerando a relação entre a qualidade da solução desejada, o tempo disponível para sua obtenção e as restrições de uso de memória.

References

[1] TSPLIB - A Library of Traveling Salesman and Related Problems, howpublished = `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`, note = Accessed: December 5, 2023,