Soluções para o problema do Caixeiro Viajante: Uma Ánalise de Algoritmos Exatos e Aproximados

Lucas Vitor [*1] **, Hélio Martins** [†2]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, MG, Brasil - CEP 31270-901

Resumo. *Este trabalho aborda a implementação e análise de desempenho de três algoritmos para solução do problema do caixeiro-viajante (Traveling Salesman Problem, TSP). Os algoritmos avaliados incluem uma solução exata baseada no método de Branch-and-Bound e duas soluções aproximadas: Twice-Around-the-Tree e o algoritmo de Christofides. A implementação foi realizada utilizando a linguagem Python, e os experimentos analisaram aspectos de tempo de execução, consumo de memória e qualidade das soluções em instâncias de diferentes tamanhos extraídas da TSPLIB. Os resultados demonstram o compromisso entre precisão e eficiência computacional dos algoritmos, fornecendo insights sobre as condições em que cada abordagem é mais adequada.*

1. Introdução

O problema do caixeiro-viajante (TSP) é um dos desafios mais estudados em otimização combinatória, consistindo em encontrar o menor percurso que visite cada cidade de um conjunto exatamente uma vez e retorne à cidade de origem. Este problema tem aplicações em diversas áreas, incluindo logística, planejamento de rotas e bioinformática, além de ser fundamental na teoria da complexidade computacional, por ser um problema NP-difícil.

Para abordar o TSP, diferentes métodos foram desenvolvidos, desde algoritmos exatos, que garantem a solução ótima, até algoritmos aproximativos, que trocam precisão por eficiência. No âmbito deste trabalho, foram implementados três algoritmos para resolver o TSP euclidiano, em que as distâncias entre as cidades são calculadas geometricamente: o Branch-and-Bound, o Twice-Around-the-Tree e o algoritmo de Christofides. Enquanto o Branch-and-Bound busca a solução ótima através de um processo de exploração estruturada do espaço de soluções, os algoritmos Twice-Around-the-Tree e Christofides oferecem soluções aproximadas com garantias teóricas de desempenho em relação ao ótimo.

O objetivo deste trabalho é comparar a performance desses algoritmos em termos de tempo de execução, uso de memória e qualidade das soluções, aplicando-os a instâncias de diferentes tamanhos retiradas da biblioteca TSPLIB. A avaliação explora as limitações e vantagens de cada abordagem, buscando identificar cenários ideais para sua utilização. Além disso, a implementação desses algoritmos permite vivenciar as dificuldades práticas inerentes ao desenvolvimento de soluções eficientes para problemas NP-difíceis, como a representação de dados e escolha de estruturas adequadas.

---

[*]Graduando em Ciência da Computação e Técnico pela UFMG
[†]Graduando em Ciência da Computação pela UFMG

As seções seguintes apresentam a descrição detalhada das implementações realizadas, os experimentos conduzidos, a análise dos resultados obtidos e as conclusões derivadas do trabalho.

2. Metodologia

Nesta seção, descrevemos as implementações realizadas para os algoritmos Branch-and-Bound, Twice-Around-the-Tree e Christofides, destacando as escolhas de estruturas de dados, estimativas de custo e estratégias empregadas.

2.1. Branch-and-Bound

O Branch-and-Bound foi projetado para encontrar soluções ótimas explorando o espaço de soluções de forma sistemática. Para estimar os custos dos caminhos parciais, utilizamos a soma dos dois menores pesos das arestas incidentes a cada nó, uma estratégia que balanceia simplicidade e eficácia.

A exploração do espaço de soluções segue uma abordagem depth-first, que prioriza o uso eficiente de memória. Durante a busca, utilizamos poda para eliminar caminhos cujo custo acumulado, somado à estimativa de custo restante, exceda o melhor custo encontrado até o momento. Essa abordagem tem uma complexidade no pior caso de $O(n!)$, onde n é o número de vértices, devido à necessidade de explorar todas as permutações possíveis em casos extremos.

2.2. Twice-Around-the-Tree

O Twice-Around-the-Tree é um algoritmo aproximativo que utiliza uma árvore geradora mínima (MST) como base. Primeiramente, construímos a MST usando o algoritmo de Kruskal, com complexidade $O(m \log n)$, onde m é o número de arestas e n o número de vértices. Em seguida, realizamos um passeio Euleriano no grafo da MST, garantindo que todas as arestas sejam percorridas pelo menos uma vez. A conversão do circuito Euleriano em um caminho Hamiltoniano ocorre em $O(n)$, onde n é o número de vértices. No geral, a complexidade do algoritmo é dominada pela construção da MST, resultando em $O(m \log n)$.

Este algoritmo garante soluções com fator de aproximação de no máximo 2 em relação ao ótimo, sendo adequado para instâncias de tamanhos moderados.

2.3. Christofides

O algoritmo de Christofides fornece soluções aproximadas com fator de aproximação de 1,5. Ele inicia com a construção de uma MST, com complexidade $O(m \log n)$. Em seguida, identifica os vértices de grau ímpar e realiza um emparelhamento mínimo entre eles utilizando um algoritmo de correspondência, cuja complexidade é $O(n^3)$ para o caso geral. O circuito Euleriano é gerado em $O(n)$, e a conversão para o caminho Hamiltoniano ocorre em $O(n)$. Assim, a complexidade total do algoritmo é dominada pela etapa de emparelhamento mínimo, resultando em $O(n^3)$.

Esses algoritmos foram implementados em Python, utilizando a biblioteca NetworkX para operações em grafos. A escolha da NetworkX facilitou a manipulação de grafos e otimizou o desenvolvimento, permitindo foco na lógica dos algoritmos em vez de detalhes de baixo nível.

3. Metodologia Experimental

Os experimentos foram divididos em duas baterias de testes para avaliar os algoritmos implementados. A primeira bateria consistiu em instâncias menores do TSP, enquanto a segunda explorou instâncias maiores. As decisões foram tomadas com base na eficiência prática dos algoritmos.

Na primeira bateria, foram utilizados grafos com 5, 10, 15 e 20 vértices, permitindo a execução de todos os algoritmos: Branch-and-Bound, Twice-Around-the-Tree e Christofides. Nesses casos, os custos ótimos foram obtidos diretamente pelo Branch-and-Bound e serviram como referência para avaliar a qualidade das soluções aproximadas.

Na segunda bateria, foram avaliadas instâncias maiores, extraídas da TSPLIB, que variam desde 52 até mais de 2000 vértices. Nessa etapa, apenas os algoritmos aproximativos (Twice-Around-the-Tree e Christofides) foram executados, uma vez que o Branch-and-Bound apresentava tempos de execução inviáveis para instâncias desse porte.

As instâncias da TSPLIB já fornecem os custos ótimos ou referências de qualidade. Esses valores foram utilizados para calcular a proporção entre o custo das soluções geradas e o ótimo conhecido, possibilitando uma análise da qualidade das soluções aproximadas em diferentes escalas.

Os dados de desempenho, como tempo de execução, uso de memória e qualidade das soluções, foram armazenados em formato JSON. Essa escolha permitiu flexibilidade para análise posterior utilizando o Google Colab, onde gráficos e estatísticas foram gerados para explorar tendências e insights. Essa abordagem combinou praticidade no armazenamento e poder analítico na visualização dos resultados.

3.1. Insights e Análise dos Resultados

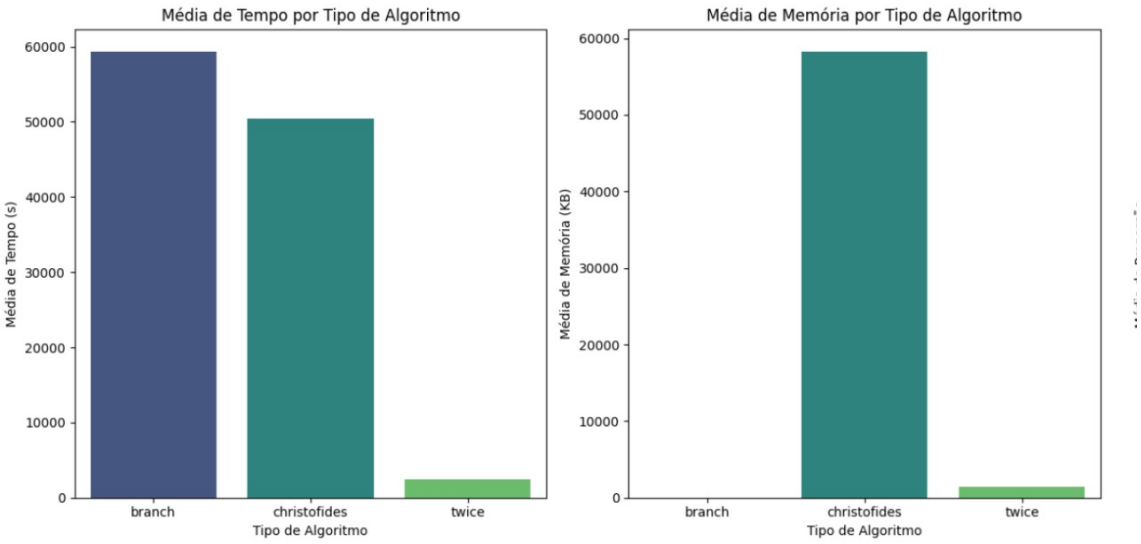

Figura 1. Tempo e memória

Os resultados obtidos nos experimentos destacaram diferenças claras no desempenho dos algoritmos analisados. Em termos de tempo de execução, o Branch-and-Bound

apresentou tempos significativamente maiores em comparação com os algoritmos aproximativos, mesmo para instâncias de pequeno porte (Figura 1). Isso reflete a alta complexidade computacional dos métodos exatos, especialmente em problemas NP-difíceis como o TSP. Por outro lado, os algoritmos aproximativos demonstraram maior eficiência, com o Twice-Around-the-Tree sendo o mais rápido em diversos casos.

No consumo de memória, o Branch-and-Bound não apresentou um uso expressivo, devido ao fato de os testes realizados serem pequenos em comparação com as instâncias maiores testadas nos algoritmos aproximativos. Em contrapartida, o Twice-Around-the-Tree continua a demonstrar um uso de memória consideravelmente mais baixo do que o Christofides (Figura 1), sendo adequado para aplicações com recursos limitados.

A qualidade das soluções aproximadas foi consistente com as garantias teóricas dos algoritmos. O Twice-Around-the-Tree gerou soluções com custo até 2 vezes o ótimo, enquanto o Christofides apresentou soluções ainda melhores, próximas de 1,5 vezes o custo ótimo. Esses resultados demonstram o equilíbrio oferecido por algoritmos aproximativos entre precisão e eficiência.

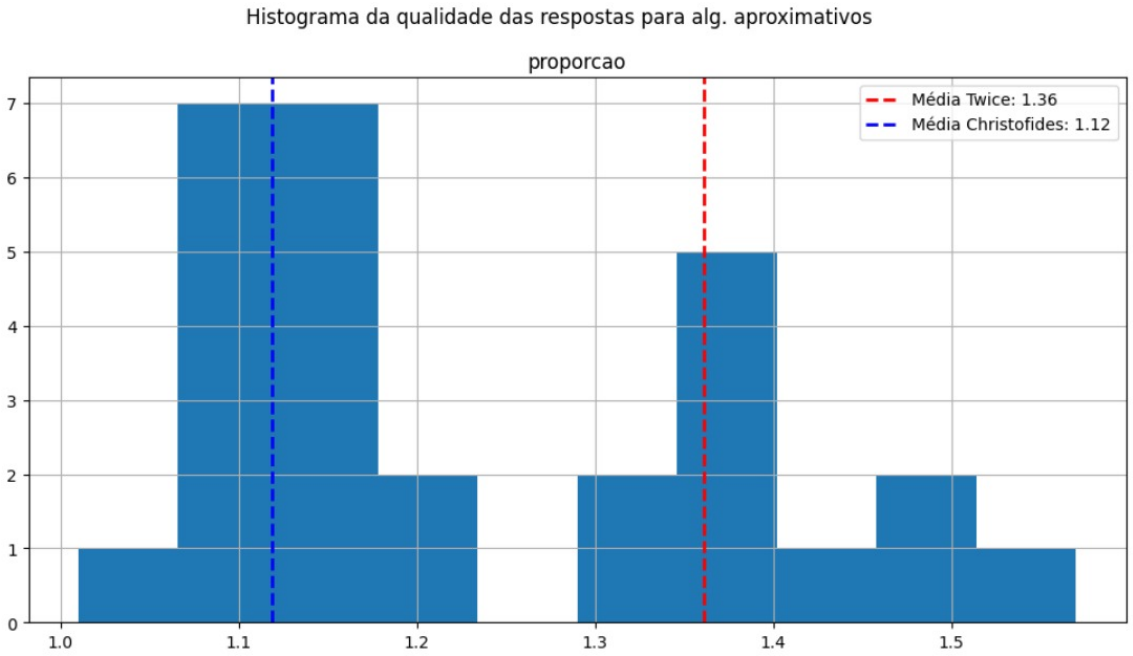

Figura 2. Qualidade das respostas aproximadas

O histograma apresentado na Figura 2 fornece uma análise visual da qualidade das soluções obtidas pelos algoritmos aproximativos Twice-Around-the-Tree e Christofides. A métrica utilizada é a proporção entre o custo da solução aproximada e o custo ótimo, sendo que valores mais próximos de 1 indicam maior qualidade.

O algoritmo Twice-Around-the-Tree apresenta uma média de 1,36, o que está alinhado com sua garantia teórica de aproximação (no máximo 2 vezes o custo ótimo). A distribuição de suas soluções é mais ampla, com proporções que variam significativamente, indicando uma maior dispersão nos resultados e soluções que, em alguns casos, podem ser consideravelmente superiores ao custo ótimo.

Por outro lado, o algoritmo de Christofides exibe uma média de 1,12, demonstrando resultados consistentemente melhores do que o Twice-Around-the-Tree. Sua distribuição é mais concentrada em torno da média, o que reflete sua maior estabilidade e a garantia de aproximação de no máximo 1,5 vezes o custo ótimo. Isso o torna uma escolha mais confiável quando se busca qualidade superior em cenários práticos.

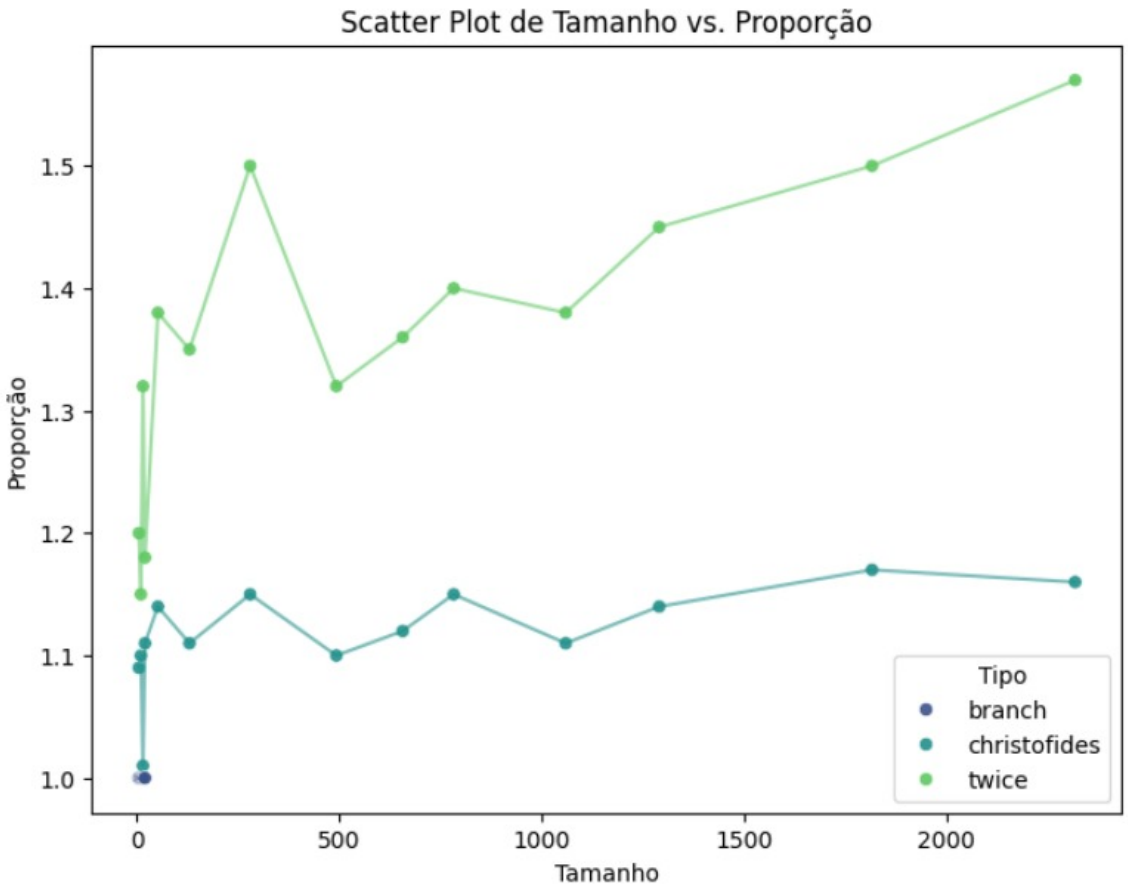

Figura 3. Proporção X Tamanho

O gráfico da Figura 3 apresentado ilustra a relação entre o tamanho das instâncias do TSP (em número de vértices) e a proporção entre o custo da solução aproximada e o custo ótimo conhecido. As curvas correspondem aos três algoritmos analisados: Branch-and-Bound, Christofides e Twice-Around-the-Tree. A análise do gráfico evidencia que, à medida que o tamanho das instâncias aumenta, a diferença na qualidade das soluções entre os algoritmos aproximativos torna-se mais evidente. O Christofides mantém sua eficiência e precisão relativa, apresentando soluções consistentemente melhores, enquanto o Twice-Around-the-Tree exibe maior variabilidade e dispersão nas proporções, alcançando valores próximos de 1,5 para as maiores instâncias. Adicionalmente, nota-se que, à medida que o grafo aumenta de tamanho, a resposta do Twice-Around-the-Tree se torna mais imprecisa, com maior desvio em relação ao custo ótimo. Isso reforça as características práticas de cada algoritmo: enquanto o Christofides é mais robusto em termos de qualidade, o Twice-Around-the-Tree se destaca pela simplicidade e execução rápida, sendo útil em cenários onde a eficiência é priorizada em detrimento da precisão.

4. Conclusão

Os experimentos realizados neste trabalho permitiram avaliar as principais características dos algoritmos Branch-and-Bound, Twice-Around-the-Tree e Christofides na resolução do problema do caixeiro-viajante. Observou-se que o Branch-and-Bound é capaz de encontrar soluções ótimas, mas sua aplicabilidade é limitada a instâncias de pequeno porte devido ao alto custo computacional. Em contrapartida, os algoritmos aproximativos demonstraram ser alternativas viáveis para instâncias maiores, com desempenhos distintos.

O Twice-Around-the-Tree destacou-se por sua simplicidade e eficiência em termos de tempo de execução, mas apresentou maior variabilidade e perda de precisão, especialmente à medida que o tamanho das instâncias aumentou. Por outro lado, o algoritmo de Christofides comprovou sua robustez, produzindo soluções de qualidade superior de forma consistente, ainda que com um custo computacional ligeiramente maior em comparação ao Twice-Around-the-Tree.

Os resultados obtidos reforçam a importância de selecionar o algoritmo adequado com base nos requisitos específicos de cada problema. Enquanto o Branch-and-Bound é ideal para cenários onde a solução ótima é indispensável, o Christofides se mostra mais apropriado para aplicações que demandam alta qualidade com eficiência razoável, e o Twice-Around-the-Tree é uma escolha prática para casos onde simplicidade e rapidez são priorizadas.

Dessa forma, este trabalho contribui para a compreensão das vantagens e limitações de diferentes abordagens na resolução do TSP, oferecendo subsídios para a escolha informada de algoritmos em contextos práticos.