UNIVERSIDADE FEDERAL DE MINAS GERAIS

Instituto de Ciências Exatas

Graduação em Sistemas de Informação

Gabriel Bifano Freddi

# Implementação de soluções algorítmicas em problemas difíceis

**Belo Horizonte**

**2023**

Gabriel Bifano Freddi

# Implementação de soluções algorítmicas em problemas difíceis

Trabalho proposto como forma de experimentação prática dos conceitos de “Backtracking”, “Branch-and-Bound”, e “Algoritmos Aproximativos” aprendidos na disciplina de Algoritmos II, ministrada na Universidade Federal de Minas Gerais (UFMG), como requisito parcial para obtenção de nota.

Professor: Renato Vimieiro

Belo Horizonte

2023

# 1. Introdução ao Trabalho:

O presente trabalho tem por objetivo documentar a exploração prática realizada, baseada nos conhecimentos aprendidos na matéria de Algoritmos II, sobre soluções para problemas difíceis. Durante o documento será apresentado o Problema do Caixeiro Viajante (Travelling Salesman Problem - TSP), assim como as soluções: 2-Aproximativa Twice Around the Tree (TAT), e 1.5-Aproximativa Christofides Algorithm. O trabalho também propõe a realização de uma solução exata, utilizando a técnica de Branch-and-Bound, porém não foi possível sua implementação neste projeto.

# 2. Introdução ao Problema:

O Problema do Caixeiro Viajante (TSP) é uma das questões mais emblemáticas e desafiadoras na teoria dos grafos e otimização combinatória. Originando-se nos anos 30, o TSP estabelece uma questão fundamental: dada uma lista de cidades e as distâncias entre cada par de cidades, qual é a rota mais curta que visita cada cidade exatamente uma vez e retorna ao ponto de origem? Formalmente, seja G um grafo completo ponderado, no qual cada vértice representa uma cidade e cada aresta representa a distância entre as cidades associadas. O objetivo é encontrar um ciclo hamiltoniano mínimo, isto é, uma permutação dos vértices que minimize a soma das distâncias ao longo do ciclo.

Matematicamente, se $n$ denota o número de cidades e $c_{ij}$ representa a distância entre as cidades $i$ e $j$, o TSP pode ser formulado como um problema de minimização:

$$Minimizar \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij}.x_{ij}$$

Em que $x_{ij}$ é uma variável descrita da seguinte forma:

$$x_{ij} = 1 \mid se\ c_{ij}\ \text{é}\ levado\ em\ consideração$$

$$x_{ij} = 0 \mid se\ c_{ij}\ caso\ contrário$$

A representação é sujeita às seguintes restrições:

1. Cada cidade deve ser visitada exatamente uma vez: $\sum_{i=1}^{n} x_{ij} = 1$ para $j = 1, 2, \ldots, n$.
2. Deve haver uma cidade de origem e uma cidade de destino: $\sum_{j=1}^{n} x_{ij} = 1$ para $i = 1, 2, \ldots, n$.
3. Garantir a ausência de sub ciclos no ciclo hamiltoniano: $u + n.x_{ij} - n.x_{ji} \leq u - 1$ para $2 \leq u \leq n$ e $i, j$ pertencentes ao ciclo hamiltoniano.

O Problema do Caixeiro Viajante figura de maneira proeminente entre os Problemas NP-completos, destacando-se como um paradigma desafiador de otimização combinatória. A comprovação de sua pertinência a essa classe foi estabelecida por Richard Karp em 1972, que demonstrou a redução polinomial do Problema da Satisfatibilidade Booleana para o TSP, consolidando-o como NP-completo. A formulação do TSP como NP-completo implica que não existe um algoritmo polinomial conhecido para encontrar a solução ótima em tempo razoável para instâncias de grande porte, demandando a exploração de abordagens heurísticas e aproximativas. Esta complexidade intrínseca do TSP não apenas ressalta sua relevância na teoria da computação, mas também inspira a busca contínua por estratégias inovadoras para resolver problemas práticos e suas variantes em diversos campos, desde logística até planejamento de redes.

O Problema do Caixeiro Viajante manifesta-se em diversas variantes, cada uma com características específicas que refletem distintas aplicações do problema. O Caixeiro Viajante Simétrico (TSP Simétrico) é a forma clássica, onde as distâncias entre as cidades são idênticas em ambas as direções, obedecendo à propriedade da simetria. Em contraste, o Caixeiro Viajante Assimétrico (ATSP) considera distâncias que podem variar dependendo da direção percorrida entre duas cidades, ampliando a representação de situações mais realistas, como rotas de sentido único ou custos de transporte assimétricos. Por sua vez, o Caixeiro Viajante Euclidiano (ETSP) limita o problema para cenários euclidianos, incorporando coordenadas espaciais para as cidades e considerando as distâncias euclidianas como medidas de proximidade. Cada variação do TSP apresenta desafios únicos e suscita a necessidade de algoritmos e estratégias específicas, influenciando as aplicações práticas dessas formulações em áreas tão diversas quanto logística, telecomunicações e roteamento de veículos. Neste projeto serão estudadas as soluções para a ETSP, ou seja, a variante do TSP aqui apresentada é simétrica e respeita a desigualdade triangular.

# 4. Implementações:

Nesta seção, serão explorados os algoritmos utilizados para a solução do TSP, assim como detalhes a nível de implementação e execução do código, a fim de expor os impedimentos, desafios, e soluções encontrados durante a arquitetura do projeto, e o seu desenvolvimento.

## 4.1. Seleção de ferramentas:

Para a execução do projeto, foi escolhida a linguagem de programação Python, em sua versão 3.10, que foi executada nas máquinas da Google através da ferramenta Google Colab por ter sido sugerida pelo professor em sala de aula, além da familiaridade do autor com a linguagem e seus recursos. Como forma de agilizar o desenvolvimento, foi permitida nas especificações do trabalho, a utilização da biblioteca NetworkX como forma de abstrair os grafos para memória do computador, e por sua API completa de métodos e operações sobre grafos. Durante a implementação dos algoritmos, foram usadas duas estruturas principais de dados: Matrizes, e Dicionários. Ambas possuem tempo de acesso pertencentes a O(1).

## 4.2. Twice Around the Tree TAT:

O Twice Around the Tree é um algoritmo aproximativo desenvolvido para o TSP. O TAT destaca-se por sua simplicidade conceitual e eficácia na obtenção de soluções aproximadas. Este algoritmo possui um fator de aproximação de 2, o que significa que a solução gerada pelo TAT é, no máximo, duas vezes mais longa do que a solução ótima. A sua complexidade computacional do TAT é notavelmente eficiente, com uma complexidade assintótica de $O(n^2 \log n)$ para o ETSP, onde 'n' é o número de nós a serem visitadas. Essa eficiência torna o TAT uma escolha atrativa para instâncias do TSP em que a obtenção de uma solução exata em tempo polinomial é impraticável.

O funcionamento do TAT é fundamentado em um raciocínio intuitivo, onde o ciclo hamiltoniano é construído a partir de um grafo gerador mínimo. Inicialmente, uma árvore geradora mínima de G é identificada usando um algoritmo como o de Kruskal. Em seguida, é realizada uma DFS (tour), em que ocorre a ordenação dos nós em um vetor H, baseado na ordem em que são visitados pela primeira vez. Finalmente, o caminho é dado pelo ciclo hamiltoniano percorrendo-se os nós de H. A simplicidade do TAT, juntamente com sua capacidade de fornecer soluções razoáveis em um tempo computacional eficiente, torna-o uma ferramenta valiosa na resolução aproximada do TSP em diversos contextos práticos.

## 4.3. Christofides Algorithm:

O Algoritmo de Christofides representa uma significativa contribuição no campo dos algoritmos aproximativos para o Problema do Caixeiro Viajante, apresentando uma solução que combina elegância algorítmica com um fator de aproximação notável. Desenvolvido por Nicos Christofides em 1976, o algoritmo é reconhecido por alcançar um fator de aproximação de 3/2, representando um marco importante em relação ao desempenho teórico dos algoritmos para o TSP.

A complexidade computacional do Algoritmo de Christofides é polinomial, com uma complexidade assintótica de $O(n^3)$, onde 'n' denota o número de nós a serem visitados. Esta eficiência relativa, em conjunto com a garantia de um fator de aproximação sublinear, posiciona o algoritmo como uma escolha atrativa para instâncias do TSP em que a obtenção de uma solução exata é computacionalmente proibitiva.

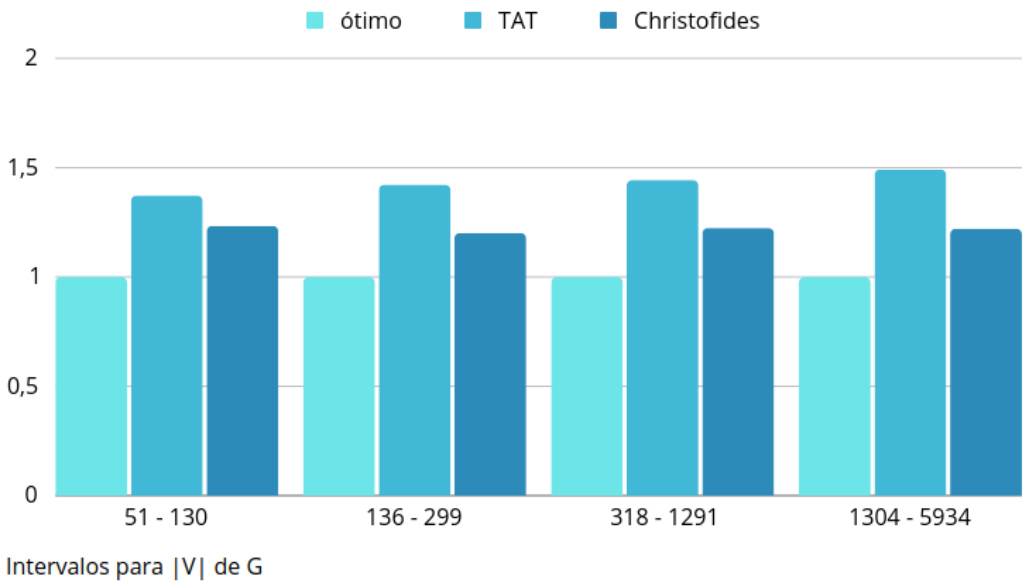
O funcionamento do Algoritmo de Christofides é notável por sua abordagem multifacetada. Inicialmente, o algoritmo constrói uma árvore geradora mínima (MST), utilizando o algoritmo de Kruskal ou uma variação adaptada. Posteriormente, é extraído um subgrafo I com todos os vértices de grau ímpar da MST, representando potenciais extremidades do ciclo hamiltoniano. Esses vértices são então pareados (através de um match perfeito mínimo) e as novas arestas formadas pelo pareamento são então adicionadas à MST para formar um grafo euleriano. E por fim é feito o tour pelo ciclo euleriano, gravando-se em um vetor H a ordem da primeira visitação dos nós, formando, assim como no TAT, o ciclo hamiltoniano que representa a solução do algoritmo. O Algoritmo de Christofides destaca-se como uma solução abrangente e eficiente para o TSP.

## 5. Exposição dos resultados:

Nesta seção será avaliada a dimensão prática da disciplina e dos resultados obtidos através dos algoritmos supracitados. Como banco para as instâncias de TSP, foi utilizada a TSPlib disponibilizada pela instituição de ensino superior Universidade de Heidelberg. Dentre as instâncias disponíveis nesta biblioteca, foram destacadas pelo professor da disciplina aquelas de caráter simétrico, e distribuição euclidiana (portanto, instâncias da ETSP). A análise proposta abaixo será feita com respeito a qualidade das respostas, e do tempo que cada um dos algoritmos levou para encerrar.

Os testes foram feitos usando as máquinas padrão do Google Colab, tendo 10.7 GB de RAM disponíveis para o programa. Com essa quantidade de recursos, vale ressaltar que o programa estourou a memória disponível no sistema para grafos com 11 mil nós ou mais. Portanto foram avaliadas 72 instâncias.

**Desempenho médio de cada algoritmo entre os intervalos de |V| de G**




Acima é possível aferir um fenômeno interessante. Apesar de o TAT e o Christofides serem algoritmos 2, e 3/2 aproximativos, respectivamente, na média eles encontram caminhos bem menores do que o seu limiar de garantia. Dependendo de por qual nó se começa a percorrer a Árvore Geradora Mínima, nos dois algoritmos, esse caminho pode ser melhor ou pior. Sendo assim, é possível conceber uma abordagem em que inúmeras inicializações aleatórias são feitas para escolher o nó raíz, e o resultado final é comparado entre elas. Tal modificação poderia inclusive ser facilmente implementada em paralelo, uma vez que as operações efetuadas nesta etapa do algoritmo são triviais e ligeiras. Tais resultados podem ser utilizados como limite superior da heurística de abordagens Branch-and-Bound, com soluções exatas para  o TSP.

Percebe-se também uma retilinearidade no crescimento da resposta média apresentada pelo TAT em relação a resposta ótima. Tal fato pode ser indício de que conforme o número de nós em G cresce, o TAT tende a retornar respostas cada vez mais próximas do

seu limite superior. Observação semelhante não pode ser feita do algoritmo de Christofides, que possui uma variação quase inexistente na qualidade de suas respostas em relação ao número de nós. De fato, nas maiores amostras, o christofides apresentou suas melhores respostas, demonstrando assim, na prática, uma robustez maior ao tamanho da entrada, em comparação à sua contraparte mais simplória.

Agora para a análise do tempo de execução de ambos os algoritmos, os dados foram colocados em uma tabela. Abaixo segue o tempo máximo em segundos necessário para completude do cálculo para cada algoritmo, em relação ao número 'n' de nós da instância:

| **n** | **< 400** | **< 575** | **< 1000** | **< 1800** | **< 3000** | **< 5000** | **Máx** |
|---|---|---|---|---|---|---|---|
| **Christofides** | 1 | 3 | 8 | 47 | 211 | 753 | 902 |
| **TAT** | 1 | 2 | 6 | 35 | 161 | 688 | 814 |

Na análise temporal feita na tabela acima é possível perceber o rápido crescimento de ambos os algoritmos, que apesar de possuírem complexidade polinomial, também crescem de forma quadrática (TAT) e cúbica (Christofides). Para instâncias maiores, a implementação provavelmente seria melhor aplicada em linguagens mais rápidas e '*memory friendly*' como C, e C + +.

## 6. Conclusão:

Uma vez encerrada a análise dos resultados, e todas as comparações tendo sido feitas, é possível perceber a discrepância no tempo e na qualidade das respostas de forma bem evidente. Como esperado, o algoritmo de Christofides apresentou respostas médias superiores ao TAT, e também um gasto maior de tempo para encerrar suas operações. No entanto é válido ressaltar que a diferença no tempo de execução foi menor do que o esperado, tendo a última instância (com 5934 nós), uma diferença de ~11% do tempo de execução do TAT em relação ao Christofides.

O trabalho demonstrou com sucesso a capacidade de obter respostas válidas dentro do limiar de garantia de ambos os algoritmos aproximativos, observando empiricamente o resultado esperado a partir da teoria. Como dito anteriormente no trabalho, a aplicação de um algoritmo de Backtracking com heurísticas (Branch-and-Bound) infelizmente não pode ser implementada a tempo, tendo os desafios com a biblioteca NetworkX na implementação do algoritmo de Christofides, tomado muito tempo da produção. Sendo assim, a conclusão aqui esboçada se apresenta incompleta, uma vez que não há a possibilidade de comparar o desempenho das soluções apresentadas, com as abordagens exatas, que podem receber muitos hiperparâmetros em sua composição (na definição dos limiares), que enriqueceriam várias das análises aqui presentes.