Universidade Federal de Pelotas

Algoritmo e Estrutura de Dados III

Algoritmos para solução do problema do Caixeiro Viajante(TSP)

Autor:
Guilherme Hepp da Fonseca

28 de janeiro de 2025

Sumário

1 Introdução

O problema do Caixeiro Viajante (TSP - Travelling Salesman Problem) é um dos problemas mais estudados na área de Otimização Combinatória e Teoria dos Grafos, devido à sua relevância teórica e prática. Ele consiste em encontrar o menor caminho que percorre um conjunto de cidades exatamente uma vez, retornando ao ponto de origem. Embora sua formulação seja simples, o TSP é um problema NP-difícil, o que significa que não existe, até o momento, um algoritmo eficiente capaz de resolver todas as instâncias em tempo polinomial. Este desafio o torna um campo de estudo importante, com aplicações em áreas como logística, planejamento de rotas, redes de telecomunicação e biologia computacional.

2 Algoritmos Exatos

Os algoritmos exatos têm como objetivo encontrar a solução ótima para o problema do Caixeiro Viajante (TSP), garantindo que o custo total do percurso seja o menor possível. Para isso, esses algoritmos exploram exaustivamente o espaço de soluções possíveis, considerando todas as combinações de caminhos entre os vértices. No entanto, a complexidade computacional associada a esses métodos aumenta exponencialmente com o número de vértices, o que os torna inviáveis para instâncias de grande porte. A seguir, apresentaremos um exemplo de Algoritmo Exato e o resultado das execuções. Os algoritmos descritos podem ser acessados diretamente no repositório no GitHub.

2.1 Algoritmo de Força Bruta

O método de força bruta é uma abordagem direta para resolver o Problema do Caixeiro Viajante (TSP). Ele funciona enumerando todas as possíveis permutações dos vértices de um grafo, calculando o custo total de cada percurso e selecionando aquele com o menor custo como a solução ótima. Para um grafo com n vértices, o número de permutações possíveis é dado por $(n-1)!$, pois o ponto de partida é fixo e os outros vértices podem ser visitados em qualquer ordem. O algoritmo avalia cada percurso possível, calcula seu custo e armazena aquele que apresenta o menor valor. Isso garante que a solução encontrada seja exata, representando o caminho mais econômico que visita todos os vértices exatamente uma vez e retorna ao ponto inicial.

Apesar de sua simplicidade, o método de força bruta apresenta uma grave limitação: a complexidade computacional cresce de forma fatorial, $O(n!)$. Esse crescimento exponencial no número de permutações torna o algoritmo impraticável para instâncias grandes.

2.1.1 Execuções

Com a complexidade de tempo tão elevada, não conseguimos atingir a resposta final do algoritmo de Força Bruta. Mesmo a instância TSP3, TSP4 e TSP5 rodando por 12h, não conseguiu atingir a resposta final. Assim, não temos exatamente o tempo que levaria para cada uma das instâncias rodar na máquina utilizada neste trabalho. Abaixo segue a tabela comparativa com os tempos dos dois primeiros testes, que tiveram resposta exata, e dos três últimos, para os quais foi feita uma estimativa de tempo até a finalização do algoritmo:

Arquivo	**Vértices**	**Resultado Ótimo**	**Tempo**
TSP1	11	253	4.02 s
TSP2	6	1248	0.15860 ms
TSP3	15	1194	403 h
TSP4	44	7013	8.19634×10^{44} h
TSP5	29	27603	2.72621×10^{21} h

Tabela 1: Resultados Algoritmo Força Bruta com Vértices e Tempo de Execução

Para estimar o tempo de execução dos outros arquivos, usamos o tempo que o programa `tsp1_253` levou para concluir e dividimos pela quantidade de permutações que ele realizou. O tempo foi calculado da seguinte forma:

$$\frac{4,02\text{s}}{(11-1)!} = 1,11 \times 10^{-6}\,\text{s ou } 1,11\,\text{us}$$

Este é o tempo aproximado que cada permutação leva para ser gerada e testada. Com isso, conseguimos calcular quanto tempo os outros arquivos vão levar para executar, multiplicando o tempo calculado acima com a quantidade de permutações de cada matriz.

Ou seja, se um arquivo possui N permutações, o tempo estimado para sua execução será:

$$\text{Tempo estimado} = \left(1,11 \times 10^{-6}\,\text{s}\right) \times N$$

Essa estimativa permite prever o tempo de execução dos arquivos restantes com base no tempo por permutação calculado. Utilizamos a transformação de segundos para horas nos três últimos testes para estimar com maior precisão o tempo até a finalização do algoritmo.

Como não conseguimos esperar até o final da execução, deixamos cada uma das três últimas instâncias rodando por cerca de uma hora e anotamos o caminho que elas encontraram e o melhor custo após o tempo decorrido.

Arquivo	**Resultado Ótimo**	**Resultado Aproximativo**	**Tempo**
TSP3	1194	1287	1h exato
TSP4	7013	22304	1h exato
TSP5	27603	44685	1h exato

Tabela 2: Resultados Algoritmo Força Bruta

Pode-se observar que, para o TSP 3, o caminho encontrado foi satisfatório, até mesmo superando minhas expectativas. No entanto, não podemos fazer a mesma afirmação para as outras duas instâncias. É provável que, à medida que o programa continuasse a rodar, os resultados melhorassem, possivelmente se aproximando de uma solução subótima dentro de um tempo aceitável.

3 Algoritmos Aproximativos

Como foi discutido anteriormente, o Problema do Caixeiro Viajante (TSP) apresenta uma complexidade que torna inviável resolver o problema de forma exata em um tempo aceitável, especialmente para instâncias maiores. Resolver o TSP de forma exata frequentemente requer tempos de execução que ultrapassam a escala prática, podendo levar anos ou até séculos para serem concluídos, impossibilitando obter a resposta enquanto ainda estamos vivos.

Diante disso, é necessário adotar uma abordagem diferente, onde uma solução que não seja a ótima pode ser considerada aceitável, desde que seja possível prever o quão distante do ótimo ela está. Para esse fim, utilizamos algoritmos aproximativos, que oferecem uma solução próxima do ótimo em tempo polinomial. Esses algoritmos, embora não garantam a precisão absoluta, são projetados para fornecer respostas dentro de um limite aceitável, definido pelo fator de aproximação, equilibrando qualidade e eficiência computacional.

3.1 Algoritmo Christofides

O Algoritmo de Christofides foi criado pelo matemático e cientista da computação Nicos Christofides em 1976. Christofides propôs esse algoritmo como uma solução aproximativa para o Problema do Caixeiro Viajante (TSP). Ele desenvolveu a heurística para fornecer uma solução eficiente e com uma garantia de proximidade ao ótimo, especialmente em grafos completos e ponderados.

O algoritmo de Christofides é amplamente conhecido por ser uma das melhores heurísticas aproximativas para o TSP, pois ele garante uma solução que é no máximo 1,5 vezes o custo da solução ótima. Essa característica é o grande diferencial do algoritmo, já que ele não só resolve o problema de forma eficiente, mas também assegura que a solução estará dentro de uma margem aceitável do ótimo. O algoritmo de Christofides segue um passo-a-passo composto por seis etapas principais:

1. Cria uma árvore geradora mínima T de G.
2. Cria uma lista O de vértices com grau ímpar em T.
3. Acha a mínima combinação perfeita M no grafo dado pelos vértices de O.
4. Combina as arestas de M e T para formar um multigrafo H.
5. Forma um circuito Euleriano em H.
6. Remove os vértices repetidos no circuito Euleriano, formando um circuito Hamiltoniano.

3.1.1 Execuções

Por estarmos utilizando de um algoritmo aproximativo, já esperávamos que suas respostas fossem ser diferentes da resposta ótima.

Arquivo	**Resultado Ótimo**	**Resultado Aproximado**	**Erro**	**Tempo**
TSP1	253	267	1,055	0,105 ms
TSP2	1248	1208	0,967	0,0655 ms
TSP3	1194	1254	1,05	0,109 ms
TSP4	7013	13073	1,86	0,699 ms
TSP5	27603	38814	1,40	0,268 ms

Tabela 3: Resultados Algoritmo Aproximativo

No caso do TSP4, o erro entre o resultado ótimo e o resultado aproximado foi superior a 1,5, alcançando um valor de 1,86. Isso indica que, para essa instância, o algoritmo aproximativo gerou uma solução que está significativamente distante da solução ótima, apresentando um erro considerável. Esse comportamento pode ser atribuído à complexidade do problema, já que, para instâncias maiores, como o TSP4, o algoritmo aproximativo tende a fornecer resultados menos precisos, embora ainda dentro de um intervalo tolerável para problemas de otimização em grande escala.

No entanto, os resultados para as outras instâncias foram bastante satisfatórios. Para o TSP1, TSP2 e TSP3, os erros foram menores que 1,1, o que demonstra que o algoritmo aproximativo conseguiu encontrar soluções bastante próximas da ótima, com um desempenho eficiente. Esses resultados evidenciam que, apesar das limitações do algoritmo em instâncias grandes, ele ainda é capaz de fornecer respostas adequadas e dentro de um tempo de execução razoável para problemas de menor escala.

4 Conclusão

Com base nas análises e experimentos realizados ao longo deste trabalho sobre o Problema do Caixeiro Viajante (TSP), utilizando tanto algoritmos exatos quanto aproximativos, conseguimos observar várias conclusões importantes:

Primeiramente, o Algoritmo de Força Bruta, embora garanta uma solução ótima, apresenta uma complexidade exponencial devido ao grande número de permutações que precisa avaliar. Isso torna sua aplicação inviável em instâncias maiores, como evidenciado pelos tempos de execução extremamente elevados, que em alguns casos ultrapassaram várias horas de execução, sem garantir uma resposta completa. A estimativa do tempo de execução para as instâncias maiores mostrou que, com a implementação na máquina utilizada, o tempo se tornaria inaceitável após uma hora de execução.

Por outro lado, os Algoritmos Aproximativos, como o utilizado neste trabalho, mostraram-se bastante eficazes ao gerar soluções próximas ao ótimo em um tempo significativamente menor. Embora os resultados aproximados mostrem erros em relação à solução ótima, esses erros geralmente estão dentro de um intervalo aceitável, conforme esperado para algoritmos aproximativos. De fato, os testes realizados revelaram que, mesmo com a abordagem aproximada, a diferença entre os resultados ótimos e aproximados foi relativamente pequena, especialmente para as instâncias de menor tamanho.

Ao analisarmos os tempos de execução e os erros de aproximação, podemos concluir que o algoritmo aproximativo conseguiu fornecer soluções satisfatórias em termos de tempo, ao mesmo tempo em que respeitava o limite de erro estabelecido. No caso de instâncias maiores, onde o algoritmo exato não seria viável devido ao alto tempo de execução, os algoritmos aproximativos se mostraram uma escolha prática e eficiente.

Portanto, podemos concluir que, enquanto o algoritmo de força bruta oferece a garantia de uma solução ótima, ele é impraticável para grandes instâncias devido à sua complexidade. Já os algoritmos aproximativos, apesar de não garantirem a solução exata, oferecem uma excelente alternativa para instâncias maiores, equilibrando tempo de execução e precisão. A escolha do algoritmo a ser utilizado depende diretamente do tamanho da instância e dos requisitos de precisão do problema a ser resolvido, tornando os algoritmos aproximativos uma solução prática e eficiente para o TSP em situações reais.

Este trabalho, portanto, enfatiza a importância de compreender as limitações dos métodos exatos e a utilidade dos métodos aproximativos para lidar com problemas de otimização combinatória de grande escala, como o TSP, em cenários práticos e com restrições de tempo.

5 Referências

- Soto, Andrea Rosario Pari. *Algoritmos Aproximativos para o Problema do Caixeiro Viajante*. Tese de Mestrado. UFRJ, 1999. Disponível em: https://www.cos.ufrj.br/upload
- Wikipedia. *Caminho Hamiltoniano*. Disponível em: https://pt.wikipedia.org/wiki/Caminho$_h$a Acesso em: 27 de janeiro de 2025.
- Google Developers. Christofides Algorithm Reference. Acessado em: 27 de janeiro de 2025.
- Napoleon. O que é Brute Force Search?. Acessado em: 27 de janeiro de 2025.