Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Curso de Graduação em Ciência da Computação

Trabalho Prático 3 - Algoritmos 1

Lucas Affonso Pires

Matrícula: 2023028420

Belo Horizonte, Fevereiro de 2025

1 Introdução

Como proposta do trabalho, foi selecionado a seguinte situação: Ajudar a empresa DelivExpress a resolver o problema de definir um ponto de distribuição central e encontrar as melhores rotas indo para outras cidades e voltando para a original. O problema foi definido como o seguinte: dado uma região com cidades e estradas, sendo que uma estrada entre duas cidades é caracterizada por uma distância d, qual a menor rota que passa por todas as cidades sem repetir nenhuma e retorna a origem. A partir disso, foi necessário criar 3 possíveis soluções para o problema, uma que utiliza a força bruta como base, uma que utiliza programação dinâmica e por fim uma que utiliza solução gulosa.

2 Modelagem

Para solucionar o problema, toda solução tem como base encontrar a menor rota a partir do ponto central dado. A solução referente a força bruta simplemente tenta todas as possibilidades de caminho e retorna a menor achada. A solução referente a programação dinâmica utiliza como base o algoritmo Held-Karp. A solução gulosa tem como base a heurística do vizinho mais próximo.

A forma como cada solução foi implementada será especificada na parte de Solução.

3 Solução

Nessa seção, serão primeiramente explicados a implementação dos algoritmos de maneira breve, já que seus códigos são bem simples. Em sequência, iremos comparar os algoritmos entre si, mostrando como cada um se destaca em comparação aos outros, destacando suas vantagens e desvantagens.

- Algoritmo de Força Bruta:

  A abordagem utilizada foi a seguinte: Explora todas as permutações possíveis das cidades para encontrar a rota que minimiza o custo total. Para isso o algoritmo gera todas as sequências em que as cidades podem ser visitadas (incluindo o retorno à cidade inicial), calcula o custo de cada rota somando os pesos das arestas entre as cidades na sequência, compara os custos e seleciona a rota com o menor valor.

- Algoritmo Held-Karp:

  A abordagem utilizada foi a seguinte: Utiliza programação dinâmica com representação por máscaras de bits para reduzir a complexidade do problema, evitando o cálculo repetido de subproblemas. O funcionamento é como segue: Considera todos os subconjuntos de cidades (exceto a cidade inicial) usando uma máscara de bits. Armazena em uma tabela dp[mask][i] o custo mínimo para alcançar a cidade i tendo

visitado o conjunto de cidades representado por mask. Atualiza a tabela para cada subconjunto e, ao final, adiciona o custo para retornar à cidade inicial. Reconstrói o caminho ótimo a partir da tabela de custos e dos predecessores.

- Algoritmo Vizinho Mais Próximo:

  A abordagem foi a seguinte: Adota uma estratégia heurística que constrói a solução de forma incremental, sempre escolhendo o próximo vértice mais próximo (não visitado) a partir da posição atual. O funcionamento é como segue: Inicia na cidade de origem e, a cada passo, seleciona a cidade não visitada com a menor distância da cidade atual. Repete esse processo até que todas as cidades sejam visitadas, e por fim, retorna à cidade inicial para completar o ciclo.

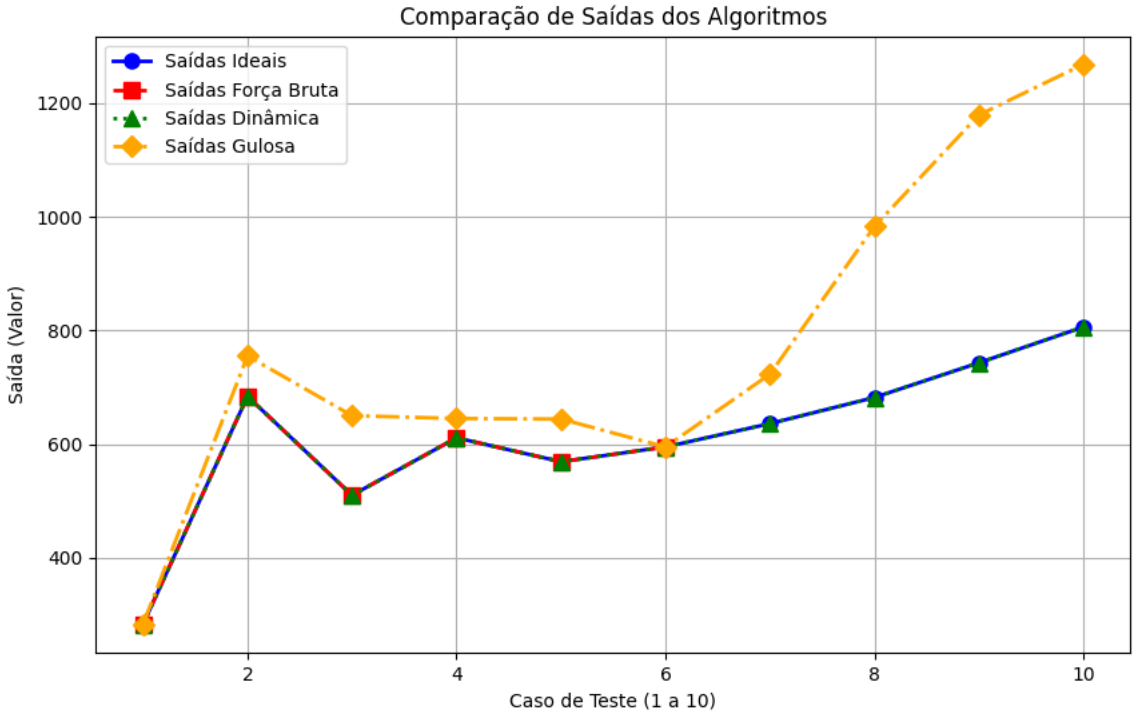

Figura 1: Comparativo da qualidade das soluções.

Percebemos, analisando o gráfico, que as soluções ideais são obtidas tanto pelo algoritmo de Força Bruta, quanto pelo algoritmo Held-Karp, apesar de que o algoritmo de Força Bruta se torna inviável a partir do caso de teste 6 por conta de sua complexidade temporal (será melhor explicado na análise de complexidade), já o algoritmo guloso em geral tem uma saída maior em cerca de 1,23 o valor do caso ideal pois tem a computação

mais rápida do problema, conforme será visto a seguir.

4 Análise de Complexidade

A análise de complexidade do programa foi relizada baseando-se no tempo e espaço utilizado pelos algoritmos. Adiante, serão analizadas as complexidades de tempo e espaço para ambos os algoritmos e em seguida serão expostos imagens comparando a eficiência de cada algoritmo nos casos de teste fornecidos.

- Algoritmo Força Bruta:

  Analisaremos primeiro a complexidade de espaço. O espaço utilizado é basicamente para armazenar a rota atual e a melhor rota encontrada, o que requer $O(n)$ espaço, além de espaço adicional para a recursão (profundidade máxima $O(n)$).

  Analisando agora a complexidade de tempo. O algoritmo gera todas as permutações das cidades. Se houver n cidades, o número de permutações é $O(n!)$.

- Algoritmo de Held-Karp:

  Analisando a complexidade de espaço, o algoritmo utiliza uma tabela dp de tamanho $(2^(n-1)) * n$ e uma tabela auxiliar parent de tamanho semelhante. Em resumo, a complexidade é $O(n * 2^n)$

  Analisando agora a complexidade de tempo, a abordagem usa uma tabela dp indexada por máscaras de bits e por cidade. O número de máscaras é $2^(n-1)$ (para as cidades exceto a inicial) e, para cada máscara, o algoritmo pode iterar sobre até n cidades, e para cada uma tenta estender para outras n cidades. Assim, a complexidade temporal é $O(n^2 * 2^n)$.

- Heurística do Vizinho Mais Próximo:

  Analisando a complexidade de espaço, o algoritmo utiliza um vetor para marcar as cidades visitadas $O(n)$ e armazena a rota $O(n)$, tendo complexidade total, também, $O(n)$.

  Analisando por fim a complexidade temporal, para cada uma das n iterações (visitando cada cidade), o algoritmo varre todas as cidades para encontrar a próxima com a menor distância, o que custa $O(n)$ em cada iteração, ou seja, a complexidade final é $O(n)$.

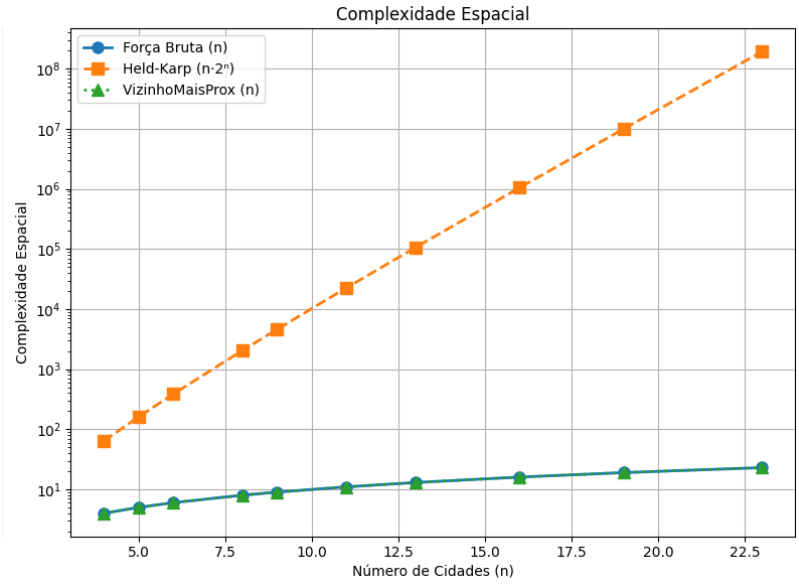

Figura 2: Comparativo Espacial teórico.

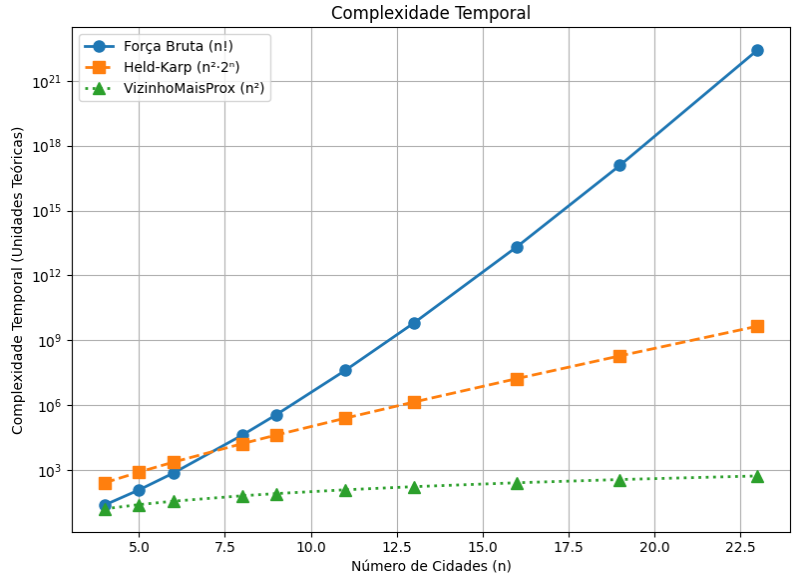

Figura 3: Comparativo Temporal teórico.

É perceptível, analisando os gráficos as vantagens e desvantagens de cada imple-

mentação. O algoritmo de Força Bruta, por ser simples e fácil de implementar compensa para grafos pequenos, tendo custo linear de espaço e sendo um dos mais rápidos, porém, conforme o aumento do grafo, é perceptível que usá-lo se torna inviável devido ao longo tempo necessário para computá-lo. A implementação dinâmica consegue produzir um resultado ótimo em todos os casos, mas ainda sim exige um tempo longo de computação para grafos muito grandes, além de utilizar uma quantidade bem maior de espaço do que os outros algoritmos. Vale a pena seu uso para encontrar soluções em grafos medianos, porém ao buscar a solução ideal em grafos muito grandes, o espaço e tempo que são necessários se tornam inviáveis (apesar de que nos casos teste ele conseguiu computar o maior grafo com uma certa demora). Por fim a implementação gulosa é a mais eficiente tanto em tempo quanto em espaço, sendo muito útil para grafos grandes, porém, vimos anteriormente que nem sempre a solução fornecida por ele é ideal. Cabe ao usuário definir se o trade-off de eficiência compensa a perda da solução ideal

5 Considerações finais

Quanto à experiência de realizar esse trabalho prático, no geral, as implementações de código para que os testes funcionassem foram muito simples. O mais trabalhoso acabou sendo tomar as decisões de qual algoritmos utilizar. O algoritmo de Força Bruta acabou sendo simples pois era apenas tentar todas as possibilidades, porém os outros dois acabaram necessitando de um pensamento mais profundo, principalmente o da solução gulosa, pois o trade-off de qualidade na solução por uma maior velocidade é uma questão bem individual, saber se a qualidade é boa o suficiente pela velocidade ganha é um ponto que não é fácil de avaliar.

Concluindo, para casos onde o grafo é simples, a força bruta é simples de se fazer e tem um resultado ideal sem muito esforço, porém se torna inviável conforme o número de cidades aumentam. Em casos onde precisamos de soluções ótimas o algoritmo Held-Karp se mostra útil, porém em casos de grafos muito grandes a qualidade de tempo perdida é bem alta. Por fim, para a solução gulosa, a qualidade de tempo e espaço é excelente porém a perda de qualidade da solução é o problema principal, definir se vale a pena essa perda ou não foi uma das principais dúvidas do trabalho.

6 Referências

`https://drive.google.com/file/d/1XKw1F9Tp-2p4K8jGOs7JGOQ155FWag9v/view?usp=sharing` (Livro do Cormen)

`https://drive.google.com/file/d/1voIWA4wS5uVhqavKBbau-71C0cVtuomi/view?usp=sharing` (Livro do Kleinberg)

Slides da matéria de Algoritmos I