Trabalho Prático 3

Ítalo Dell'Areti

Universidade Federal de Minas Gerais (UFMG)

Belo Horizonte – MG – Brasil

1. Introdução

O problema proposto a ser resolvido consiste em desenvolver um sistema para a empresa DelivExpress determinar a localização ideal de seus pontos de distribuição e as rotas de entrega. A empresa precisa estabelecer centros de distribuição em diferentes regiões e, a partir de cada centro, realizar entregas passando por todas as cidades da região exatamente uma vez, retornando ao ponto inicial. O desafio principal está em:

1. Encontrar a rota de menor custo que visita todas as cidades
2. Lidar com o fato de que a complexidade do problema cresce fatorialmente com o número de cidades
3. Avaliar diferentes estratégias de solução conforme o tamanho da região

Ao final, devemos analisar e comparar estas três estratégias para fornecer à DelivExpress uma compreensão clara de quando utilizar cada abordagem. Portanto, com base nas características e propriedades de grafos, algoritmos de busca exaustiva, programação dinâmica e heurísticas gulosas, podemos implementar essa solução.

2. Modelagem

O problema foi modelado utilizando grafos completos, o que simplifica as implementações por garantir caminhos diretos entre todas as cidades. O grafo é estruturado de forma onde:

- Vértices (V): Representam as cidades de cada região onde cada cidade pode ser um ponto de distribuição, tendo seu custo determinado pela distância total da rota.
- Arestas (E): Representam estradas entre duas cidades, com peso que indica a distância entre elas (grafo ponderado).
- Grafo G(V,E): Representa uma região completa, onde todas as cidades estão conectadas diretamente entre si.
- Matriz de Adjacência: Estrutura escolhida para representar o grafo por ser mais eficiente para grafos completos e por facilitar o acesso às distâncias entre cidades.

Principais algoritmos utilizados:

- Força Bruta: Gera todas as permutações possíveis de cidades e encontra a rota de menor custo.
- Held-Karp (Programação Dinâmica): Resolve subproblemas de forma recursiva usando memoização para evitar recálculos.
- Vizinho Mais Próximo (Guloso): Constroi a rota incrementalmente escolhendo sempre a cidade não visitada mais próxima da atual.

3. Solução

3.1 Força Bruta

A solução por força bruta se baseia no conceito de busca exaustiva para encontrar a rota ótima. Para cada cidade inicial possível, o algoritmo:

1. Gera todas as permutações possíveis das cidades restantes
2. Calcula o custo total de cada rota
3. Mantém registro da melhor solução encontrada

O processo completo envolve:

1. Fixar a cidade inicial como ponto de distribuição
2. Para cada permutação das outras cidades:
    - Calcular custo acumulado seguindo a sequência
    - Adicionar custo de retorno à cidade inicial
    - Atualizar melhor solução se necessário

Esta abordagem garante encontrar a solução ótima por testar exaustivamente todas as possibilidades válidas.

3.2 Programação Dinâmica

A solução por programação dinâmica utiliza o algoritmo de Held-Karp, que:

1. Define estados como pares (cidade_atual, cidades_visitadas)
2. Utiliza memoização para armazenar resultados parciais
3. Constroi a solução ótima de forma bottom-up

O processo de resolução envolve:

1. Representar conjunto de cidades visitadas com máscara de bits
2. Para cada estado (cidade, máscara):
    - Calcular menor custo para visitar cidades restantes
    - Armazenar resultado para reuso
    - Reconstruir caminho ótimo ao final

Esta estratégia reduz a complexidade de tempo do problema de $O(n!)$ para $O(n^2 2^n)$ através do reuso de soluções de subproblemas, mas aumenta a complexidade de espaço em contrapartida, sendo ela ($O(n2^n)$).

3.3 Algoritmo Guloso

A solução gulosa implementa o algoritmo do vizinho mais próximo, que constroi a rota incrementalmente:

1. Começa da cidade inicial definida
2. Em cada passo:
    - Escolhe cidade não visitada mais próxima
    - Adiciona ao caminho atual
    - Marca como visitada
3. Ao final, retorna à cidade inicial

O processo é repetido começando de cada cidade possível, mantendo o melhor resultado encontrado. Esta abordagem garante:

- Complexidade polinomial $O(n^2)$
- Solução aproximada em tempo viável
- Compromisso entre qualidade e eficiência

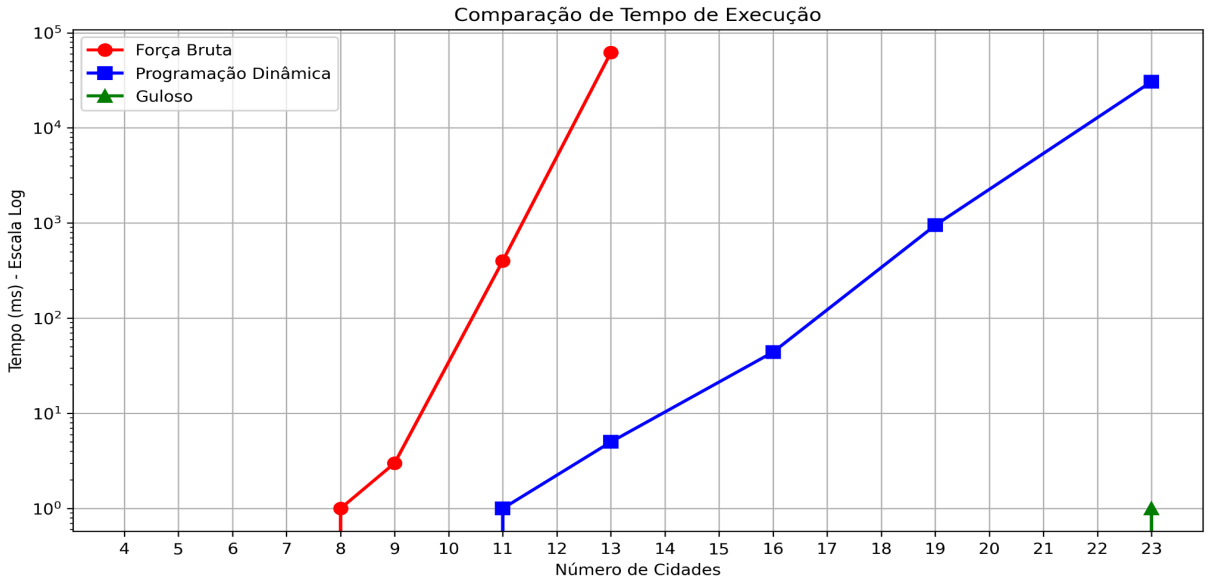
A principal diferença entre as estratégias está no trade-off entre:

- Força Bruta: Garantia de otimalidade × Custo computacional proibitivo
- Programação Dinâmica: Otimalidade × Uso expressivo de memória
- Guloso: Eficiência computacional × Possível subotimalidade

Esta combinação de abordagens permite escolher a estratégia mais adequada conforme o tamanho da região e os requisitos de qualidade da solução.

4. Análise de Complexidade:

4.1 Tempo de Execução

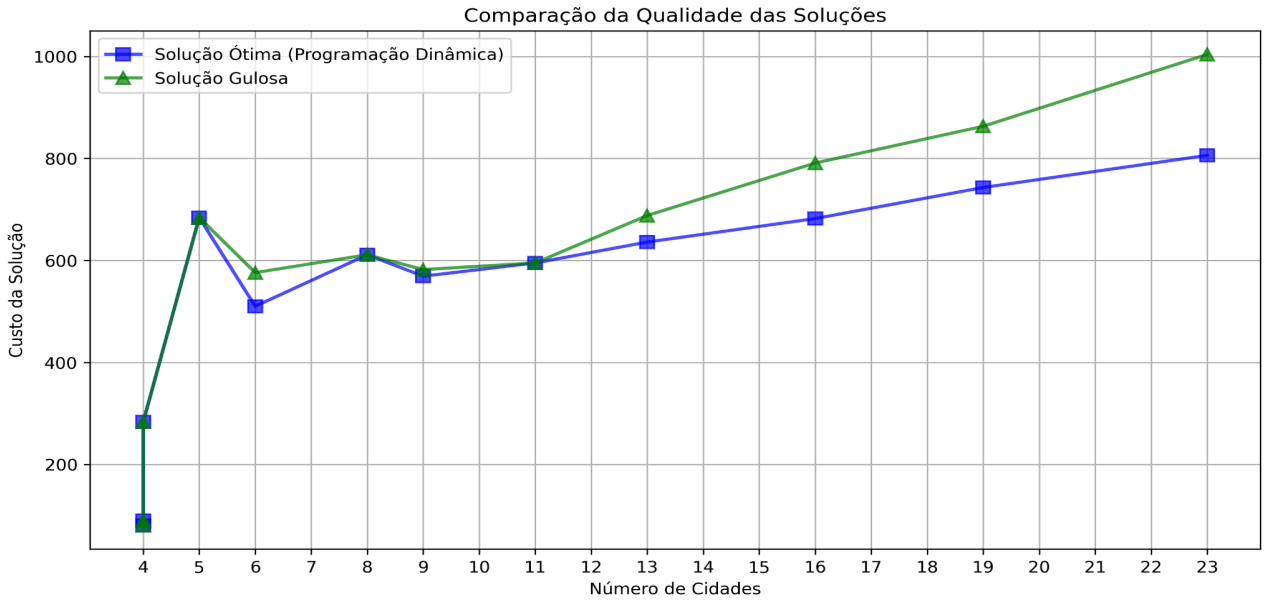


1. Força Bruta (O(n!)):
    - Viável até 13 cidades (62 segundos)
    - Inviável para 16 cidades (estimado em 58 horas)
    - Crescimento claramente fatorial
2. Programação Dinâmica ($O(n^2 2^n)$):
    - Processou todos os casos
    - 23 cidades: 30 segundos
    - Crescimento exponencial mas muito mais controlado que força bruta
3. Guloso ($O(n^2)$):
    - Tempo praticamente constante (0-1ms)
    - Escalável para qualquer número de cidades
    - Desempenho previsível

4.2 Uso de Memória

1. Força Bruta:
    - Memória constante O(n)
    - Apenas armazena permutação atual
    - Muito eficiente em memória
2. Programação Dinâmica:
    - Crescimento exponencial $O(n2^n)$
    - 16 cidades: 852 KB
    - 19 cidades: 37 MB
    - 23 cidades: 783 MB
3. Guloso:
    - Memória constante O(n)
    - Apenas armazena estado atual
    - Extremamente eficiente em memória

4.3 Qualidade das Soluções



1. Soluções Ótimas (Força Bruta e PD):
    - Garantia matemática de otimalidade
    - Mesmo custo para ambos os algoritmos
    - Benchmark para avaliar o guloso
2. Soluções Gulosas:
    - Média de desvio: 13.37%
    - Maior desvio: 24.6% (23 cidades)
    - Soluções ótimas em 6 casos

4.4 Trade-offs

A análise experimental permite estabelecer critérios claros de escolha:

1. Para regiões pequenas (≤ 13 cidades):
    - Força Bruta é viável
    - Garante solução ótima
    - Baixo uso de memória
2. Para regiões médias (14-19 cidades):
    - Programação Dinâmica é ideal
    - Garante solução ótima
    - Uso moderado de memória
3. Para regiões grandes (≥ 20 cidades):
    - Algoritmo Guloso é a única opção prática
    - Soluções cerca de 13% acima do ótimo
    - Extremamente rápido e eficiente

5. Considerações Finais

O desenvolvimento deste trabalho proporcionou insights valiosos sobre o trade-off entre qualidade de solução e eficiência computacional. A parte mais desafiadora foi determinar com precisão os limites de cada estratégia e documentar de forma clara os resultados experimentais.

Os dados coletados mostram claramente que não existe uma solução única ideal para todos os casos, mas sim um espectro de soluções que devem ser escolhidas conforme o tamanho da região e os requisitos específicos de qualidade e tempo de resposta.

Além disso, compilei o programa em vídeo para demonstrar seu funcionamento.

Link:
https://drive.google.com/file/d/1mNFCoupKQRuWETCoDUduWK0qVSvr4EoF/view?usp=sharing

6. Referências

Geeks For Geeks: The Traveling Salesman Problem.

CORMEN, T. H. et al. Introduction to Algorithms. 3rd ed. MIT Press, 2009.

KLEINBERG, J.; TARDOS, E. Algorithm Design. Pearson, 2005.

APPLEGATE, D. L. et al. The Traveling Salesman Problem: A Computational Study. Princeton University Press, 2006.