Universidade Estadual do Oeste do Paraná - Unioeste

Campus de Cascavel
Colegiado de Ciência da Computação

2º Trabalho de Projeto e Análise de Algoritmos (PAA)

Análise Comparativa entre Floyd-Warshall e Dijkstra para o Problema de Caminhos Mínimos Entre Todos os Pares

Autores:

João Gabriel Fazio Pauli
Lucas Batista Deinzer Duarte
Fernando Seiji Onoda Inomata

Agosto de 2025

Resumo

Este relatório apresenta uma análise teórica e empírica de duas estratégias para a resolução do problema de encontrar o caminho mínimo entre todos os pares de vértices (All-Pairs Shortest Path): o algoritmo de Floyd-Warshall e a execução sucessiva do algoritmo de Dijkstra a partir de cada vértice do grafo. O objetivo é comparar as duas soluções em termos de custo assintótico, tempo de execução cronológico e correção dos resultados. As implementações foram desenvolvidas em C++ e testadas sobre um conjunto de 15 grafos, representados por matrizes de adjacência, com diferentes números de vértices.

1 Introdução

O problema de encontrar o caminho mínimo entre todos os pares de vértices (All-Pairs Shortest Path - APSP) em um grafo ponderado é um problema fundamental em ciência da computação, com aplicações em roteamento de redes, logística e bioinformática. Diversos algoritmos foram propostos para solucionar este problema, cada um com diferentes características de desempenho.

Este trabalho foca na comparação de duas abordagens clássicas: o algoritmo de Floyd-Warshall, que utiliza uma estratégia de programação dinâmica, e a aplicação repetida do algoritmo de Dijkstra, uma estratégia gulosa, partindo de cada vértice do grafo. O objetivo é analisar o trade-off entre as duas soluções, comparando a teoria de complexidade com o desempenho prático observado em testes cronometrados.

2 Metodologia

Os experimentos foram realizados em um ambiente consistente para garantir a validade da análise empírica. Os algoritmos foram implementados em C++ e executados sobre os mesmos conjuntos de dados para uma comparação justa.

- **Configuração do Ambiente:** Os testes foram executados em um computador com processador i5 12th Gen, 8 GB de memória RAM, e sistema operacional Windows11.
- **Linguagem e Ferramentas:** Os algoritmos foram implementados na linguagem C++, utilizando o compilador G++ com o padrão C++11. O tempo de execução foi cronometrado utilizando a biblioteca `<chrono>`.
- **Conjunto de Dados:** Foram utilizados os 15 arquivos de texto fornecidos, representando grafos com 'V' variando de 10 a 1.500 vértices. Cada arquivo contém o número de vértices 'V' seguido de uma matriz de adjacência 'V x V' com os pesos das arestas.
- **Procedimento Experimental:** Para cada grafo, cada um dos dois algoritmos foi executado 6 vezes. A primeira execução foi descartada (warm-up), e a média aritmética das 5 execuções subsequentes foi registrada.
- **Validação:** Ao final de cada teste, as matrizes de distância geradas pelos dois algoritmos foram comparadas para garantir que ambos produziram o mesmo resultado, validando a correção das implementações.

3 Análise Teórica de Complexidade

3.1 Algoritmo de Floyd-Warshall

O algoritmo de Floyd-Warshall é baseado em programação dinâmica e utiliza três laços de repetição aninhados, cada um iterando sobre os 'V' vértices do grafo. Isso resulta em uma complexidade de tempo consistente e independente da estrutura do grafo.

$$O(V^3)$$

3.2 Algoritmo de Dijkstra (Executado V vezes)

Para resolver o problema APSP, o algoritmo de Dijkstra é executado uma vez para cada vértice de origem. A implementação utilizada neste trabalho usa um vetor simples para a fila de prioridades, o que confere a cada execução de Dijkstra uma complexidade de $O(V^2)$. Como o processo é repetido 'V' vezes, a complexidade total é:

$$V \times O(V^2) = O(V^3)$$

Teoricamente, ambas as abordagens possuem a mesma complexidade assintótica para grafos densos, representados por matrizes de adjacência.

4 Resultados e Discussão

Os dados de tempo de execução coletados nos experimentos foram compilados na Tabela 1.

Tabela 1: Tempo médio de execução (em microssegundos) para cada algoritmo.

Tamanho da Entrada (V)	Tempo (Floyd-Warshall) µs	Tempo (Dijkstra) µs
10	10,60	5,00
25	100,60	42,60
50	753,60	287,80
75	2.450,20	876,20
100	5.923,40	2.025,80
150	19.863,20	6.471,00
200	46.126,80	15.650,60
250	90.627,20	29.569,20
300	156.974,80	51.521,60
400	377.300,00	122.846,00
500	740.718,20	240.407,20
650	1.769.758,20	573.197,00
800	3.298.530,20	1.071.181,80
1000	6.460.669,20	2.102.040,00
1500	21.758.981,60	7.070.735,40

A Figura 1 plota os dados da Tabela 1 para permitir uma análise visual do desempenho.

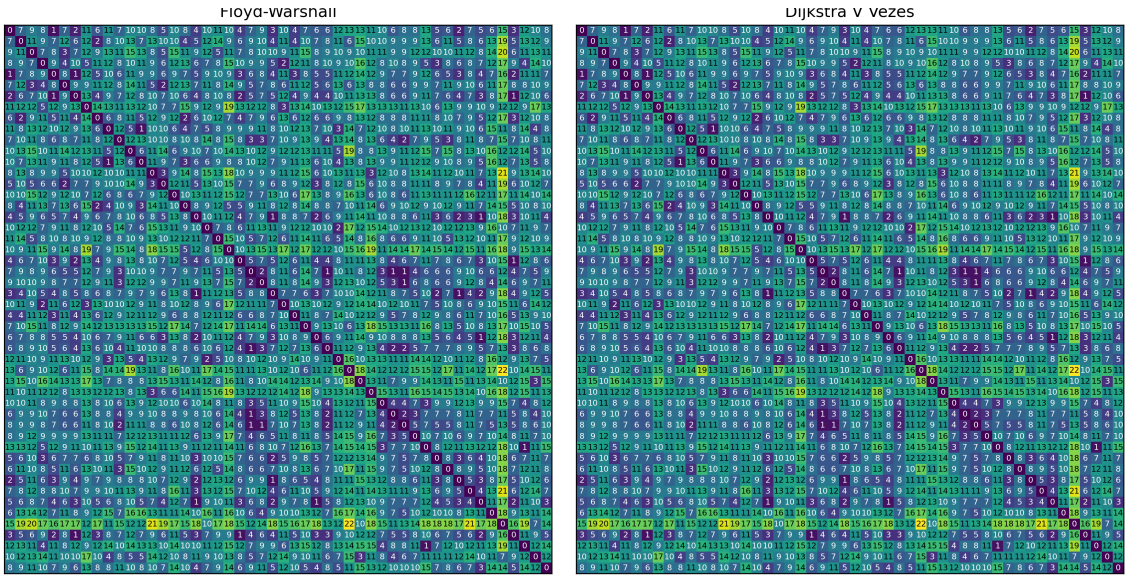

Figura 1: Gráfico comparativo do tempo de execução entre Floyd-Warshall e Dijkstra V vezes.

A análise dos resultados demonstra que, embora ambas as estratégias possuam a mesma complexidade assintótica teórica de $O(V^3)$, existe uma diferença de performance significativa na prática. A abordagem de executar o Dijkstra V vezes foi consistentemente mais rápida que o Floyd-Warshall em todas as instâncias testadas, chegando a ser aproximadamente 3 vezes mais veloz para os maiores grafos.

Essa diferença pode ser atribuída às operações realizadas dentro dos laços de cada algoritmo. O Floyd-Warshall executa um número fixo de V^3 operações de soma e comparação. O Dijkstra, por sua vez, realiza um número de operações que, embora cúbico no pior caso, pode ser menor dependendo da estrutura do grafo (número de arestas), e suas operações internas (busca por um mínimo e relaxamento) podem ser, em conjunto, computacionalmente menos custosas que os acessos contínuos à matriz do Floyd-Warshall.

Em todos os testes, foi validado que ambas as implementações produziram matrizes de distância idênticas, confirmando sua correção.

5 Conclusão

Este trabalho implementou e comparou duas soluções para o problema de caminhos mínimos entre todos os pares. Foi verificado que tanto o algoritmo de Floyd-Warshall quanto a execução repetida de Dijkstra produzem resultados corretos e idênticos.

Embora ambos apresentem uma complexidade teórica de $O(V^3)$ para grafos densos, a análise empírica revelou que a abordagem baseada em Dijkstra é consistentemente mais rápida na prática. Isso evidencia que,

mesmo dentro da mesma classe de complexidade, as constantes e a natureza das operações de cada algoritmo podem levar a diferenças de desempenho significativas no mundo real.