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

João Gabriel Fazio Pauli[1], **Lucas Batista Deinzer Duarte**[1], **Fernando Seiji Onoda Inomata**[1]

[1] Colegiado de Ciência da Computação – Universidade Estadual do Oeste do Paraná (UNIOESTE)
Campus de Cascavel – Cascavel – PR – Brasil

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.*

Abstract. *This report presents a theoretical and empirical analysis of two strategies for solving the All-Pairs Shortest Path problem: the Floyd-Warshall algorithm and the successive execution of Dijkstra's algorithm from each vertex of the graph. The objective is to compare the two solutions in terms of asymptotic cost, chronological execution time, and correctness of the results. The implementations were developed in C++ and tested on a set of 15 graphs, represented by adjacency matrices, with varying numbers of vertices.*

1. Introdução

1.1. Contexto

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. Suas aplicações são vastas, abrangendo desde o roteamento de pacotes em redes de computadores, onde cada nó precisa conhecer a rota mais eficiente para todos os outros, até sistemas de logística para otimização de entregas, e bioinformática, no alinhamento de sequências genéticas. A eficiência na resolução deste problema é, portanto, um fator crítico para o desempenho de muitos sistemas computacionais complexos.

1.2. Problema

Formalmente, dado um grafo ponderado $G = (V, E)$, onde V é o conjunto de vértices e E o conjunto de arestas com pesos (custos) associados, o problema do APSP consiste em encontrar, para cada par de vértices $(u, v) \in V$, o caminho de menor custo de u até v. A solução é tipicamente representada por uma matriz de distâncias D, onde cada entrada $D[i][j]$ contém o custo do caminho mínimo do vértice i para o vértice j. É importante notar que alguns algoritmos, como o de Dijkstra, não operam corretamente sobre grafos com arestas de peso negativo, uma restrição que não se aplica ao Floyd-Warshall.

1.3. Proposta do Experimento

Este trabalho propõe a implementação e análise comparativa de duas abordagens clássicas para a resolução do problema APSP em grafos sem arestas de peso negativo. A primeira é o algoritmo de Floyd-Warshall, um método canônico de programação dinâmica. A segunda consiste na aplicação repetida do algoritmo de Dijkstra, uma estratégia gulosa, partindo de cada um dos 'V' vértices do grafo. A comparação será realizada sob a ótica do desempenho cronológico e da validação da corretude dos resultados. A hipótese central é que, embora ambas as abordagens possuam a mesma complexidade assintótica de $O(V^3)$ para grafos densos, a solução baseada em Dijkstra apresentará um desempenho prático superior devido a um menor número de operações totais e a um melhor aproveitamento da estrutura do grafo.

2. Metodologia

Os experimentos foram realizados em um ambiente consistente para garantir a validade da análise empírica.

- ***Configuração do Ambiente:*** *Os testes foram executados em um computador com processador i5 12th Gen, 8 GB de memória RAM, e sistema operacional Windows 11.*
- ***Linguagem e Ferramentas:*** *Os algoritmos foram implementados na linguagem C++, utilizando o compilador G++ (padrão -std=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'.*
- ***Procedimento Experimental:*** *Para cada grafo, cada algoritmo foi executado 6 vezes. A primeira execução foi descartada (warm-up), e a média aritmética das 5 execuções subsequentes foi registrada.*
- ***Validação:*** *A corretude das implementações foi validada de duas formas: através da comparação programática das matrizes de distância geradas e pela verificação da soma de todos os custos de caminhos finitos em cada matriz, que deveriam ser idênticas.*

3. Análise Teórica de Complexidade

3.1. Algoritmo de Floyd-Warshall

O algoritmo de Floyd-Warshall resolve o problema APSP de forma incremental. Ele utiliza três laços de repetição aninhados, 'para k, para i, para j', onde 'k' representa o vértice intermediário que está sendo considerado para melhorar os caminhos entre 'i' e 'j'. Essa estrutura resulta em uma complexidade de tempo consistente e independente da estrutura do grafo de $O(V^3)$ [Cormen et al. 2012].

3.2. Algoritmo de Dijkstra (Executado V vezes)

A implementação utilizada do algoritmo de Dijkstra, que encontra os caminhos mínimos a partir de uma única fonte, usa um vetor simples para a fila de prioridades. Isso confere a cada execução uma complexidade de $O(V^2)$. Como o processo é repetido para cada um dos 'V' vértices como origem, a complexidade total para resolver o problema APSP é $V \times O(V^2) = O(V^3)$. Teoricamente, ambas as abordagens possuem a mesma complexidade assintótica para grafos densos.

4. Resultados e Discussão

Os dados de tempo de execução e a validação dos custos foram coletados e compilados nas tabelas a seguir. A Tabela 1 apresenta o comparativo de desempenho cronológico.

Tabela 1. Tempo médio de execução (convertido) para cada algoritmo.

Tamanho (V)	Tempo (Floyd-Warshall)	Tempo (Dijkstra V vezes)
10	10,60 µs	5,00 µs
25	100,60 µs	42,60 µs
50	753,60 µs	287,80 µs
75	2,45 ms	0,88 ms
100	5,92 ms	2,03 ms
150	19,86 ms	6,47 ms
200	46,13 ms	15,65 ms
250	90,63 ms	29,57 ms
300	156,97 ms	51,52 ms
400	377,30 ms	122,85 ms
500	0,74 s	0,24 s
650	1,77 s	0,57 s
800	3,30 s	1,07 s
1000	6,46 s	2,10 s
1500	21,76 s	7,07 s

Para evidenciar a corretude das implementações, conforme a expectativa do trabalho, a Tabela 2 apresenta a soma de todas as distâncias finitas da matriz de resultado para cada algoritmo.

Tabela 2. Tabela de Validação: Soma das distâncias finitas na matriz de resultado.

Arquivo de Entrada	Soma (Floyd-Warshall)	Soma (Dijkstra V vezes)
Entrada 10.txt	442	442
Entrada 25.txt	4486	4486
Entrada 50.txt	23282	23282
Entrada 75.txt	43908	43908
Entrada 100.txt	88362	88362
Entrada 150.txt	208108	208108
Entrada 200.txt	458462	458462
Entrada 250.txt	677962	677962
Entrada 300.txt	1062406	1062406
Entrada 400.txt	1957360	1957360
Entrada 500.txt	3069668	3069668
Entrada 650.txt	5422918	5422918
Entrada 800.txt	8105432	8105432
Entrada 1000.txt	13664302	13664302
Entrada 1500.txt	32757518	32757518

A Tabela 2 confirma que os custos totais são idênticos para todas as instâncias, validando a corretude de ambas as implementações. A análise da Tabela 1 revela que, embora ambas as estratégias possuam a mesma complexidade teórica, a abordagem de executar o Dijkstra V vezes foi consistentemente mais rápida, chegando a ser 3 vezes mais veloz para os maiores grafos. Essa diferença de performance pode ser atribuída às operações internas de cada algoritmo. O Floyd-Warshall executa um número fixo de V^3 operações, enquanto o Dijkstra, embora repetido V vezes, pode se beneficiar da estrutura do grafo (arestas com peso infinito não são exploradas), resultando em um número menor de operações efetivas.

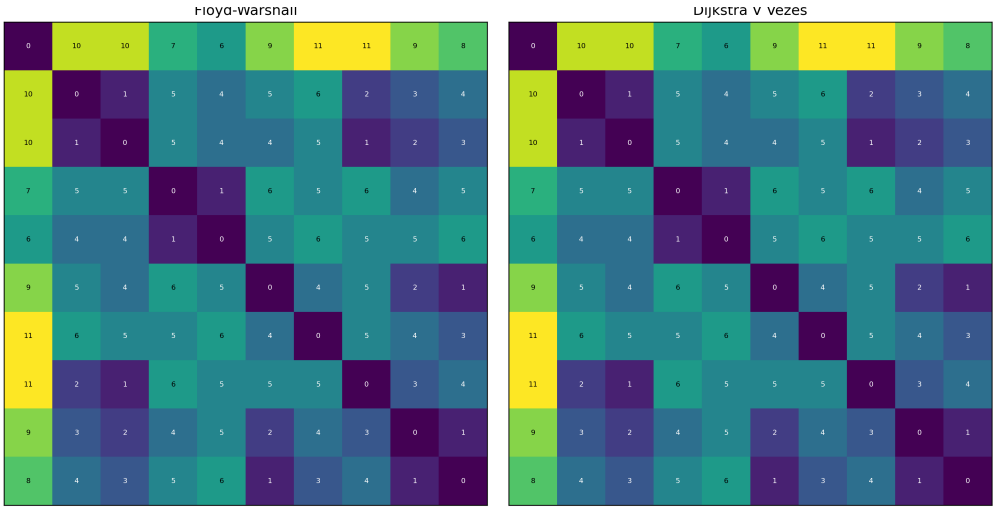

Figura 1. Visualização comparativa das matrizes de distância geradas pelos algoritmos Floyd-Warshall (esquerda) e Dijkstra V vezes (direita) para a entrada com 10 vértices.

A Figura 1 ilustra graficamente os resultados obtidos para um grafo pequeno (10 vértices). As cores representam os valores dos caminhos mínimos entre pares de vértices. É possível observar que ambas as matrizes são idênticas visualmente, o que reforça a validação da corretude entre os dois algoritmos para o problema de caminhos mínimos entre todos os pares.

5. Conclusão

Este trabalho validou empiricamente a performance de 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 para os conjuntos de dados testados. Isso evidencia que, mesmo dentro da mesma classe de complexidade, as constantes algorítmicas e a natureza das operações de cada algoritmo podem levar a diferenças de desempenho significativas no mundo real.

Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2012). Algoritmos: Teoria e Prática*. Elsevier, Rio de Janeiro, 3ª edition.*