"Planeta Água: a cultura oceânica para enfrentar as mudanças climáticas no meu território"
SECT - ICE
XII Semana de Ciência e Tecnologia SECT ICE
20 a 23 de Outubro de 2025
Realização:

Quebrando a barreira da ordenação? Desempenho de algoritmos de caminho mínimo em grafos esparsos.

Lucas Castro[1], Thailsson Clementino[1*], Rosiane de Freitas[1]

[1]Universidade Federal do Amazonas, Instituto de Computação, Av. Rodrigo Otávio Jordão Ramos, 6200, Coroado I, 69067-005, Manaus AM, Brasil.
[*] *thailsson.clementino@icomp.ufam.edu.br*

Palavras-Chave: Caminho Mínimo, algoritmos em grafos, estruturas de dados, Ordenação.

Introdução

Seja $G = (V, E)$, $|V| = n, |E| = m$, um grafo conexo com uma função de peso $w : E \to \mathbb{R}_{\geq 0}$ e um vértice fonte $s \in V$. O **Problema do Caminho Mínimo com Única Fonte** (em inglês, *Single Source Shortest Path, SSSP*) consiste em determinar, para cada vértice $v \in V$, a distancia mínima $d(v) = d(s, v)$ de um caminho em G que começa em s e termina em v.

Como um dos problemas algorítmicos mais clássicos da teoria dos grafos, o SSSP é amplamente estudado na literatura. Até o final do século passado, o algoritmo proposto por Dijkstra[3], há cerca de 70 anos, ainda era considerado o estado da arte para a resolução do SSSP. Quando combinado a uma fila de prioridade eficiente, o algoritmo de Dijkstra resolve o problema com complexidade de tempo $O((n + m) \log n)$. Em 2024, Haeupler et al.[6] demonstraram que o algoritmo de Dijkstra é universalmente ótimo para o problema natural de ordenar os vértices de acordo com suas distâncias $d(v)$ a partir da fonte s. Em grafos esparsos, onde $m = O(n)$, essa complexidade é equivalente à barreira de ordenação no modelo de computação Comparação-Adição.

Para contornar o gargalo associado à ordenação dos vértices, alguns trabalhos propuseram algoritmos com menor complexidade de tempo evitando ordenar os vértices. No modelo RAM, Thorup[7] apresentou um algoritmo com complexidade $O(m)$ para grafos não direcionados cujos pesos cabem em uma palavra, e ainda[8] estendeu a abordagem para grafos direcionados, obtendo $O(m + n \log \log n)$. Já no modelo Comparação-Adição, Duan et al.[5] propuseram um algoritmo randomizado para grafos não direcionados com complexidade $O(m\sqrt{\log n \log \log n})$, o mesmo grupo (Duan et al.[4]) desenvolveu o primeiro algoritmo determinístico cujo limite superior é $o(m + n \log n)$, com complexidade $O(m \log^{2/3} n)$.

Com o objetivo de avaliar empiricamente o desempenho do algoritmo determinístico proposto por Duan et al.[4] e compará-lo com o algoritmo clássico de Dijkstra, este trabalho realiza um estudo experimental para verificar a eficiência prática do algoritmo com o melhor limite superior assintótico disponível na literatura.

O Algoritmo de Duan et al.[4]

O algoritmo de Duan et al.[4] resolve uma extensão do SSSP denominada *Bounded Multisource Shortest Path* (BMSP), na qual se deseja computar, para cada vértice $v \in V$, a menor distância $d(v)$ a partir de algum vértice $u \in S \subseteq V$, sujeita à restrição $d(v) \leq B$.

A ideia central do algoritmo é baseada na abordagem de divisão e conquista sobre o espaço das distâncias. Em vez de processar os vértices em uma única ordem crescente de $d(v)$, como no algoritmo de Dijkstra, o método particiona o conjunto de vértices em subconjuntos delimitados por limiares de distância $B'_i \leq B$, de forma aproximadamente equilibrada. Cada subconjunto é então processado recursivamente, reduzindo o problema a instâncias menores. No caso base, um mini-Dijkstra é executado para encontrar a menor distância entre um dos pivôs e k vértices. Cada subconjunto é resolvido recursivamente, e os resultados parciais são integrados por meio de vértices *pivôs* que conectam os subproblemas.

A seleção desses pivôs é feita através de k iterações do algoritmo de Bellman–Ford[1]. O valor de k é escolhido pelos autores de modo a equilibrar o custo das execuções de Bellman–Ford com a profundidade da recursão, resultando na complexidade final $O(m \log^{2/3} n)$. O algoritmo possui três componentes principais:

(i) **Divisão e conquista** sobre o domínio das distâncias;

(ii) **Múltiplas execuções parciais de Bellman–Ford** para escolha determinística dos pivôs;

(iii) **Balanceamento analítico** entre profundidade recursiva e custo por nível, responsável pelo fator $\log^{2/3} n$.

Projeto de Experimento

Ambos os algoritmos foram implementados na linguagem `C++` (versão 20), utilizando compilação otimizada com o compilador `g++` e a flag `-O2`. Todos os experimentos foram conduzidos em um computador com `32 GB` de memória disponíveis e um processador `Intel Core i5-10400F` de `2.90 GHz`, executando o sistema operacional *Linux Mint 21.3 Cinnamon*. As implementações estão disponíveis em: `https://github.com/lcs147/bmssp`.

As instâncias utilizadas nos experimentos provêm do 9º Desafio de Implementação do DIMACS[2]. Elas representam 12 redes rodoviárias dos Estados Unidos, correspondentes a diferentes regiões do país. Em cada instância, a função de peso atribuída às arestas indica o tempo médio necessário para percorrer a rua correspondente. Por se tratarem de malhas rodoviárias, todos os grafos são esparsos; os respectivos números de vértices e arestas podem ser consultados na Tabela 1.

Para cada algoritmo, cada instância foi executada 5 vezes de forma independente, a fim de reduzir a influência de fatores externos, como variações do sistema operacional e flutuações na medição de tempo. O tempo médio de execução, medido em milissegundos, foi adotado como métrica principal de desempenho. Os resultados obtidos são discutidos na próxima seção.

Resultados e Discussão

Após a execução dos testes, observou-se que o clássico algoritmo de Dijkstra[3] apresenta desempenho superior em todas as instâncias avaliadas. Embora, teoricamente, o algoritmo proposto por Duan et al.[4] possua um melhor limite superior assintótico, sua implementação é consideravelmente mais complexa, envolvendo diversas chamadas recursicas e nuances que introduzem uma constante elevada no tempo de execução. Na prática, essa sobrecarga faz com que o algoritmo apresente desempenho inferior ao algoritmo de Dijkstra, que é simples, direto e altamente eficiente utilizando por exemplo uma estrutura de *Heap* Binário implementado diretamente em um vetor. Os tempos de execução para todas as instâncias podem ser visualizados na Tabela 1.

Instância	n	m	Tempo (ms)	
			Dijkstra[3]	Duan et al.[4]
New York City	264.346	733.846	54	986
San Francisco Bay Area	321.270	800.172	61	1071
Colorado	435.666	1.057.066	80	1431
Florida	1.070.376	2.712.798	218	3811
Northwest USA	1.207.945	2.840.208	261	4055
Northeast USA	1.524.453	3.897.636	382	5875
California and Nevada	1.890.815	4.657.742	411	7046
Great Lakes	2.758.119	6.885.658	627	9602
Eastern USA	3.598.623	8.778.114	899	12511
Western USA	6.262.104	15.248.146	1680	21299
Central USA	14.081.816	34.292.496	5175	62174
Full USA	23.947.347	58.333.344	7731	100506

Tabela 1: Resultados experimentais para instâncias do desafio de implementação do DIMACS.

Analisando casos extremos, observa-se que, mesmo na menor instância — que representa a malha rodoviária da cidade de Nova Iorque —, o algoritmo de Dijkstra mantém desempenho superior: o novo algoritmo é aproximadamente 18 vezes mais lento. Já na maior instância, que modela toda a malha rodoviária dos Estados Unidos, o algoritmo de Dijkstra resolve o problema em cerca de 7 segundos, enquanto o novo algoritmo requer aproximadamente 100 segundos, ou seja, é cerca de **13 vezes mais lento**.

Esses resultados indicam que o ganho teórico obtido por meio da análise assintótica não se traduz necessariamente em ganho prático, devido às elevadas constantes ocultas e a complexidade estrutural do novo algoritmo. Em ambientes reais, com implementações em linguagens de programação convencionais e execução em computadores pessoais, utilizar algoritmo de Dijkstra, na prática, é substancialmente mais eficiente.

Conclusões

Este trabalho apresentou uma análise experimental comparando o algoritmo clássico de Dijkstra[3] com o algoritmo recentemente proposto por Duan et al.[4], que estabelece o melhor limite superior assintótico determinístico conhecido para o Problema do Caminho Mínimo com Única Fonte (SSSP) no modelo de computação Comparação–Adição.

Os experimentos realizados sobre instâncias reais do 9º Desafio de Implementação do DIMACS demonstraram que, na prática, o algoritmo de Dijkstra supera o novo algoritmo em todas as instâncias testadas, mesmo nas de maior porte.

Como trabalhos futuros, seria interessante ampliar a análise experimental para incluir os demais algoritmos citados na introdução, que representam os melhores limites superiores conhecidos nos modelos de computação RAM e Comparação–Adição, a fim de avaliar empiricamente o impacto prático de suas diferenças teóricas.

Agradecimentos

Este trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES-PROEX) - Código de Financiamento 001, pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), bem como parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado do Amazonas – FAPEAM – por meio do projeto POSGRAD 2024/2025.

Referências

[1] Bellman, R. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90. 1958.

[2] Demetrescu, C.; Goldberg, A.; Johnson, D. 9th dimacs implementation challenge–shortest paths. *American Mathematical Society*. 2006.

[3] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, 50:269–271. 1959.

[4] Duan, R.; Mao, J.; Mao, X.; Shu, X.; Yin, L. Breaking the sorting barrier for directed single-source shortest paths. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pages 36–44. 2025.

[5] Duan, R.; Mao, J.; Shu, X.; Yin, L. A randomized algorithm for single-source shortest path on undirected real-weighted graphs. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 484–492. IEEE. 2023.

[6] Haeupler, B.; Hladík, R.; Rozhoň, V.; Tarjan, R. E.; Tetěk, J. Universal optimality of dijkstra via beyond-worst-case heaps. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2099–2130. IEEE. 2024.

[7] Thorup, M. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM (JACM)*, 46(3):362–394. 1999.

[8] Thorup, M. Integer priority queues with decrease key in constant time and the single source shortest paths problem. *Journal of Computer and System Sciences*, 69:330–353. 2004.