Implementação N. 04 – Fluxo Máximo

Aluna: Caroline Freitas Alvernaz
Disciplina: Teoria dos Grafos e Computabilidade
Professor: Zenilton Kleber Gonçalves do Patrocínio Júnior

Para a realização do trabalho, foi adaptado uma implementação baseada na técnica de Edmonds-Karp que utiliza a busca em largura para encontrar caminhos aumentantes e o processo se repete até que não haja mais caminhos disjuntos possíveis. Além disso, foi feita uma implementação de um gerador de grafos que gera grafos aleatórios direcionados e do tipo "grid" e, portanto, foram utilizados números quadrados perfeitos para a implementação. Os grafos gerados foram de tamanhos 100, 14.444, 10.000 e 45.729.

Grafos Aleatórios

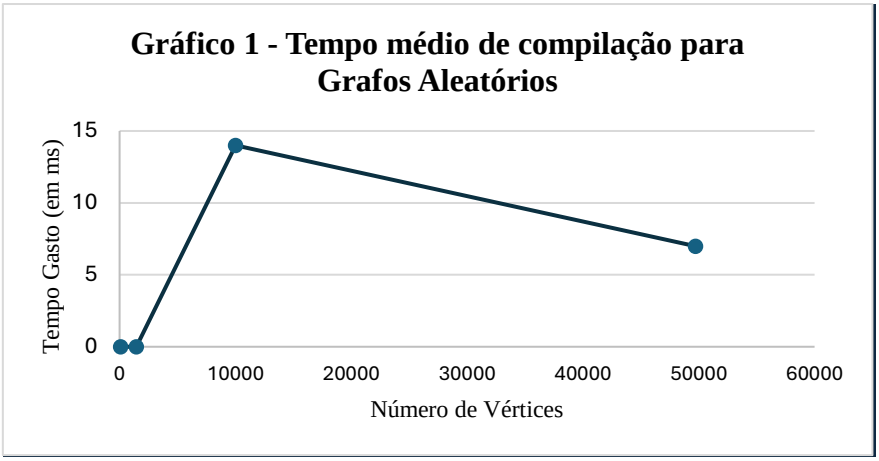
Para grafos aleatórios, foram obtidos os seguintes resultados:

Tabela 1 – resultados para Grafos Aleatórios

Número de vértices	Tempo gasto (em ms)	Número de caminhos disjuntos
100	0	1
1.444	0	2
10.000	14	3
49.729	7	2

Conforme mostra a tabela, o algoritmo apresenta grande eficiência, pois os tempos de execução são baixos, mesmo para grafos com muitos vértices. Para um grafo com 10.000 vértices, o tempo gasto foi de apenas 14 ms, e para 49.729 vértices, surpreendentemente menor de 7 ms. Esse comportamento mostra que o algoritmo continua rápido mesmo quando o tamanho do grafo aumenta bastante, graças ao uso da busca em largura (BFS) como base para o cálculo dos caminhos.

O gráfico abaixo demonstra a oscilação do tempo médio de execução apresentado:



A eficácia do algoritmo está relacionada à sua capacidade de encontrar corretamente todos os caminhos disjuntos possíveis entre dois vértices em um grafo. Nos testes realizados com grafos direcionados aleatórios, o número de caminhos encontrados variou conforme o tamanho e a estrutura de cada grafo. Em grafos menores, como o de 100 vértices, foi identificado apenas 1 caminho disjunto, o que é esperado, já que a chance de haver várias rotas independentes é menor em estruturas pequenas e menos conectadas. À medida que o número de vértices aumenta, a conectividade tende a melhorar, o que possibilitou a descoberta de mais caminhos como ocorreu com o grafo de 10.000 vértices que obteve 3 caminhos disjuntos.

Grafos Grid

Para grafos do tipo grid, ou seja, grafos onde os vértices estão organizados em forma de malha bidimensional de tamanho 10 x 10, por exemplo, foram obtidos os seguintes resultados:

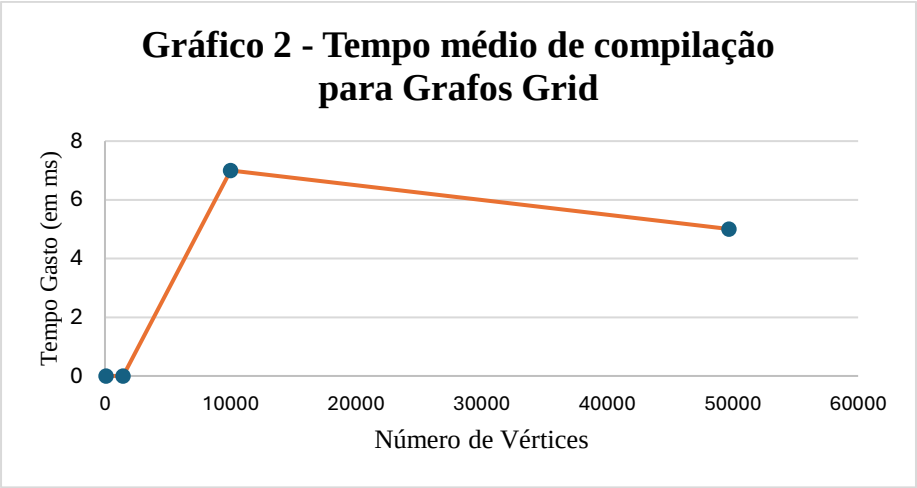
Tabela 2 – resultados para Grafos Grid

Número de vértices	Tempo gasto (em ms)	Número de caminhos disjuntos
100	0	2
1444	0	2
10000	7	2
49729	5	2

Para grafos do tipo grid, o algoritmo também demonstrou grande eficiência, com tempos de execução muito baixos mesmo para grafos maiores. Por exemplo, no grafo

com 10.000 vértices, o tempo foi de apenas 7 milissegundos, e para 49.729 vértices, o tempo caiu para 5 milissegundos, confirmando que o desempenho se mantém estável conforme o grafo cresce.

O gráfico a seguir ilustra a variação no tempo médio de execução observado.



Quanto à eficácia, o número de caminhos disjuntos encontrados foi constante em todos os casos, encontrando sempre 2 caminhos entre origem e destino. Esse resultado é esperado na estrutura de grid, já que o padrão fixo de ligações entre os vértices limita o número de rotas independentes entre vértices localizados em cantos opostos da malha.

Comparação entre os dois tipos de grafos

A tabela abaixo demonstra um comparativo entre os resultados obtidos para os dois tipos de grafos:

Tabela 3 – comparação dos desempenhos para os dois tipos de grafos

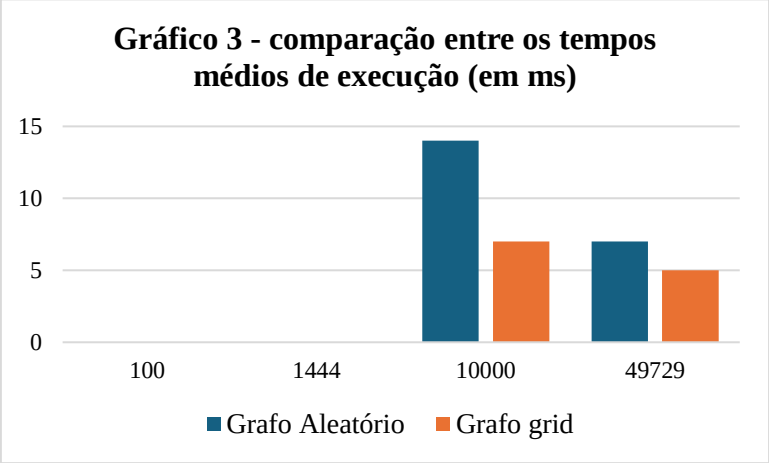
	Tempo gasto (em ms)		Número de caminhos disjuntos	
Número de vértices	Grafos Alatórios	Grafos Grid	Grafos Alatórios	Grafos Grid
100	0	0	1	2
1444	0	0	2	2
10000	14	7	3	2
49729	7	5	2	2

Os resultados demonstram que o algoritmo apresenta ótima eficiência, com tempos de execução muito baixos mesmo em grafos com grande número de vértices. Em

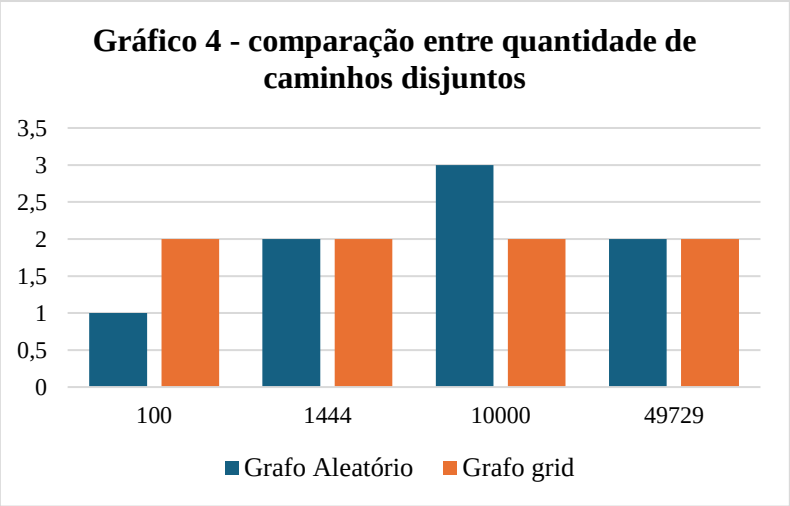
grafos aleatórios com 10.000 vértices, o tempo foi de 14 milissegundos, e em grafos ainda maiores, como o de 49.729 vértices, o tempo foi de apenas 7 ms. Nos grafos grid, os tempos foram ainda menores, variando entre 0 e 7 ms. Isso mostra que o algoritmo mantém um desempenho rápido e estável mesmo quando aumenta o número de vértices e a complexidade estrutural dos grafos, independentemente de seus tipos.

Em relação à eficácia, o número de caminhos disjuntos encontrados nos grafos aleatórios varia conforme a conectividade entre os vértices. Já para grafos grid, o número de caminhos disjuntos permaneceu constante em todos os tamanhos testados (2 caminhos).

A seguir, os gráficos fornecem uma comparação visual entre os desempenhos do algoritmo ao processar grafos aleatórios e grafos grid, em diferentes tamanhos:



O Gráfico 3 demonstra como o tempo médio de execução varia conforme o tipo e o tamanho do grafo. Observa-se que, mesmo em grafos com um número elevado de vértices, o tempo de execução permanece baixo, o que evidencia a eficiência do algoritmo. Além disso, grafos do tipo grid tendem a ter um tempo ligeiramente menor, possivelmente devido à sua estrutura mais regular e previsível.



Já o Gráfico 4 permite avaliar a eficácia do algoritmo com base na quantidade de caminhos disjuntos encontrados. Percebe-se que os grafos aleatórios apresentam uma variação maior na quantidade de caminhos, enquanto os grafos grid mantêm uma quantidade constante (2 caminhos), o que é esperado devido à sua topologia restrita.

Os resultados demonstraram que o algoritmo é eficiente, apresentando tempos de execução baixos mesmo em grafos com grande quantidade de vértices, e eficaz, ao identificar corretamente os caminhos disjuntos entre dois vértices. A comparação entre grafos aleatórios e grafos do tipo grid evidenciou que o desempenho e a quantidade de caminhos encontrados variam de acordo com a forma como os grafos são construídos, reforçando a confiabilidade do algoritmo nos diferentes cenários testados.