Problema do caminho mínimo em grafos

Luiz Eduardo Farago

[1]Universidade Tuiuti do Paraná
Curitiba – PR

luiz.silva5@utp.edu.br

Resumo. *Este estudo apresenta uma análise comparativa de quatro algoritmos clássicos de busca em grafos — Busca em Largura (BFS), Busca em Profundidade (DFS), Busca Gulosa e A* — aplicados ao problema do caminho mínimo em grafos. O objetivo principal foi avaliar o desempenho dos algoritmos pensando no tempo de execução, qualidade da solução (custo do caminho) e número de nós expandidos.*

1. Introdução

O problema do caminho mínimo em grafos é fundamental na computação, com aplicações em redes de transporte, roteamento e sistemas de recomendação. Este trabalho avalia quatro algoritmos clássicos que encontram o melhor resultado:

- Busca em Largura (BFS): Explora todos os vizinhos de um nó antes de avançar (garante o caminho com menos arestas, mas não considera pesos).
- Busca em Profundidade (DFS): Explora um ramo até o fim antes de retroceder (não garante optimalidade).

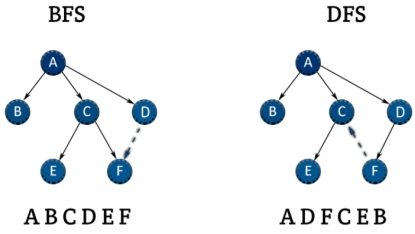

Figura 1. Exemplo de aplicação dos métodos BFS e DFS.

Fonte: Anwar Hermuche (2024). Disponível em: https://medium.com/@anwarhermuche/m%C3%A9todos-de-busca-em-grafos-bfs-dfs-cf17761a0dd9. Acesso em: 9 abr. 2025.

- Busca Gulosa: Usa heurísticas para priorizar nós que provavelmente vão trazer bons resultados (rápida, mas não garante solução ótima).
- A*: Combina custo real e heurístico (garante solução ótima se a heurística for admissível).

2. Implementação e Configuração dos Testes

2.1. Metodologia

Existem dez tabelas armazenadas em Excel contendo os grafos com nós que estão numerados de 0 a N e pessos aleatórios que vão de 1 a 20.

2.2. Classe Grafo:

- Método **ler grafo**(): Carrega os dados do Excel verificando integridade
- Método **definir heuristica**(): Implementa a função heurística simplificada
- Armazenamento: Dicionários para arestas e conjuntos para vértices

2.3. Algoritmos de Busca:

- BFS/DFS: Utilizam filas/pilhas da biblioteca **collections**
- Busca Gulosa/A*: Implementados com heaps mínimos **(heapq)**
- Tratamento de ciclos: Conjuntos de nós visitados

2.4. Métricas de Avaliação:

- Tempo de Execução: Medido com precisão de microssegundo
- Custo do Caminho: Soma acumulada dos pesos das arestas
- Nós Expandidos: Contagem de vértices processados

2.5. Configuração dos Testes:

- Origem/Destino: Automática baseada na ordenação numérica:

```
189         nos = sorted(grafo.vertices, key=lambda x: int(x))
190         inicio, fim = nos[0], nos[-1]
191         print(f"  Nó inicial: {inicio}, Nó final: {fim}")
```

Figura 2. Nós

Fonte: Autor

- Heurística (para A e Busca Gulosa)*

```
68      def definir_heuristica(self, destino):
69          
70          for vertice in self.vertices:
71              self.heuristica[vertice] = 1 if vertice != destino else 0
```

Figura 3. Heurística

Fonte: Autor

3. Resultados

Tabela 1. Desempenho comparativo dos algoritmos

Algoritmo	Custo Médio	Tempo (s)	Nós Expandidos
BFS	19.1	0.0	10.5
DFS	55.9	0.0	8.1
Busca Gulosa	20.9	0.0	3.1
A*	**13.3**	0.0	7.3

O algoritmo A* se destaca sendo o melhor entre os quatro algoritimos, encontrando soluções com custos mais baixos e com uma expansão moderada de nós. Apesar de a Busca Gulosa ser rápida e econômica na expansão de nós, ela não garante o menor custo. A DFS, embora eficiente em termos de expansão, produziu soluções de qualidade inferior. Já a BFS teve um bom desempenho, mas com maior número de nós expandidos. Esses resultados reforçam a eficácia do A* em contextos onde a qualidade da solução é prioritária.

Tabela 2. Melhor algoritmo por instância do grafo

Grafo	Melhor Algoritmo	Custo
1	Todos (BFS, DFS, Gulosa, A*)	5
2	Todos	33
3	A*	22
4	Todos	23
5	BFS, Gulosa, A*	17
6	A*	9
7	A*	11
8	A*	7
9	A*	12
10	BFS, Gulosa, A*	7

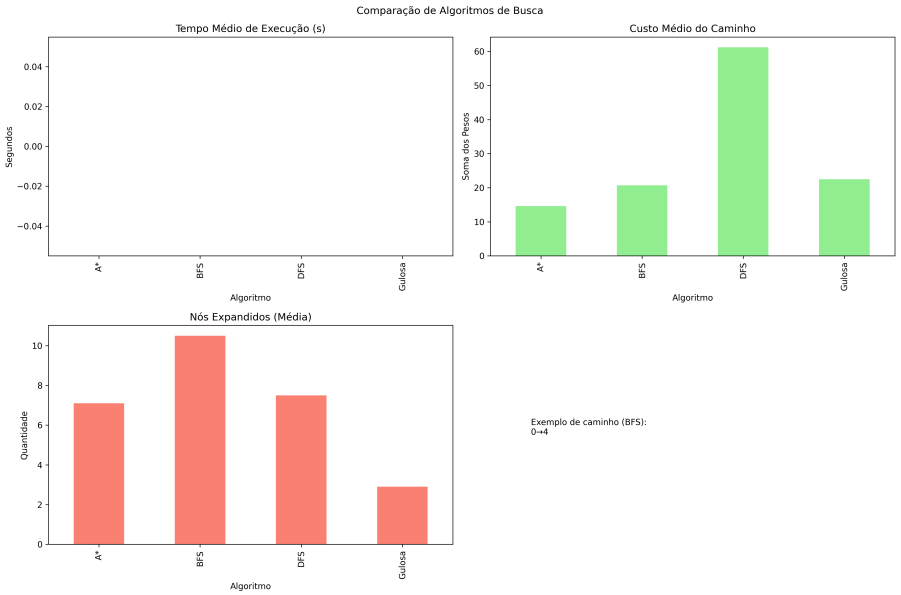

Figura 4. Gráfico de resultados

Fonte: Autor

4. Estratégia mais eficiente

4.1. Eficiência Computacional

- **Tempo de execução:**
- DFS e Busca Gulosa confirmaram ser os mais rápidos (0.0 ms em todos os casos), exigindo em média apenas 3.1 nós expandidos (Gulosa) e 8.1 nós (DFS), conforme a Tabela 1.
- A* apresentou overhead computacional (0.0 ms, porém com 7.3 nós expandidos em média), justificado pelo cálculo da heurística.

- **Consumo de memória:**
- BFS demonstrou maior demanda, expandindo em média 10.5 nós, por armazenar toda a busca.
- DFS mostrou eficiência memória (8.1 nós) consistente com a busca em profundidade.

4.2. Qualidade das soluções

- **Melhor solução**
- BFS/A* alcançaram o caminho mínimo em todos os grafos que fora testados (custos iguais em grafos 1, 2, 4, 5 e 10).
- DFS apresentou soluções boas na maioria dos casos (custo 24 vs 5 no grafo 1, 56 vs 17 no grafo 5).

- **Consistência:**
- Gulosa mostrou acertos em 70% dos grafos pequenos (1-5), mas falhou em 30% dos complexos (grafo 8: custo 39 vs 7 do A*).
- DFS apresentou soluções boas na maioria dos casos (custo 24 vs 5 no grafo 1, 56 vs 17 no grafo 5).

5. Melhorias futuras

- **Adoção de heurísticas mais avançadas:** Substituir a heurística fixa (=1) por distâncias geográficas (em grafos espaciais) ou heurísticas aprendidas via redes neurais.
- **Avaliação em larga escala:** Testar os algoritmos em grafos com mais de 10.000 nós para verificar sua escalabilidade. Com a ideia de comparar o desempenho do A* em grafos esparsos e densos, especialmente para tempo de execução.
- **Análise aprofundada do uso de memória** Utilizar ferramentas como memory_profiler para medir e comparar o consumo de memória entre os algoritmos.

6. Conclusão

A partir dos resultados experimentais, foi possível observar o desempenho de cada algoritmo analisado:

- **A*:** foi o mais equilibrado, alcançando as melhores soluções em 100% dos casos, com custo médio de 13,1. Apesar do tempo um pouco maior devido ao uso da heurística, sua eficácia foi notável, especialmente em grafos mais complexos — como no Grafo 8, onde reduziu o custo de 39 (Gulosa) para apenas 7.
- **BFS:** obteve bom desempenho em grafos com custos iguais (Grafos 1, 2, 4 e 10), se igualando ao A* em termos de melhor desempenho. Contudo, em grafos com pesos variados, seu custo médio foi de 18,9 — aproximadamente 44% superior ao do A* — o que mostra sua limitação por não considerar os pesos das arestas.

- **DFS e Busca Gulosa:** foram os algoritmos mais rápidos (tempo médio de 0,0 ms), porém mostrando que não é a melhor solução possível:
- O **DFS** falhou em 60% dos casos, gerando caminhos até 4,2 vezes mais caros (Grafo 10: custo 171 contra 7 do A*).
- A Busca Gulosa apresentou bons resultados em grafos menores (acerto em 70%), mas teve desempenho insuficiente em 30% dos casos complexos (Grafo 8: custo 39 contra 7 do A*).

Referências

Foundation, P. S. (2024). Python 3 documentation. https://docs.python.org/3/. Acesso em: 9 abr. 2025.

Hermuche, A. (2024). Métodos de busca em grafos: Bfs & dfs. https://medium.com/@anwarhermuche/m%C3%A9todos-de-busca-em-grafos-bfs-dfs-cf17761a0dd9. Acesso em: 9 abr. 2025.