Estudo do Problema do Caminho Mínimo em Grafos

Gustavo O. Guidetti, Luis F. Brasil, Eduardo H. M. Cruz

[1] Instituto Federal do Paraná (IFPR) – Campus Paranavaí
CEP 87703-536 – Paranavaí – PR – Brasil

gustavoo.guidetti@gmail.com

luis4756@live.com

eduardo.cruz@ifpr.edu.br

Abstract. *This work aims to investigate the shortest path problem, a widely recognized theme in graph theory, with several applications both in academic and practical situations in everyday life. Graph Theory is a mathematical discipline that is dedicated to the study of sets of elements known as nodes or vertices, as well as the connections between them, called edges. Each of these sets is classified as a graph. In the present study, a graph library was created with the aim of generating and manipulating graphs, including their nodes and edges. In addition, two algorithms frequently used to solve the shortest path problem were implemented: Dijkstra's algorithm and Floyd-Warshall's algorithm, which served as objects of analysis. All implementations were carried out in C#. The developed library worked as a tool to test the implemented algorithms, where manually created algorithms were also executed to verify whether each implementation presented the expected results.*

Resumo. *Este trabalho visa investigar o problema do caminho mínimo, um tema amplamente reconhecido na teoria dos grafos, com várias aplicações tanto acadêmicas quanto em situações práticas do dia a dia. A Teoria dos Grafos é uma disciplina matemática que se dedica ao estudo de conjuntos de elementos conhecidos como nós ou vértices, bem como das conexões entre eles, chamadas arestas. Cada um desses conjuntos é classificado como um grafo. No presente estudo, foi criada uma biblioteca de grafos com o intuito de gerar e manipular grafos, incluindo seus nós e arestas. Além disso, foram implementados dois algoritmos frequentemente utilizados para resolver o problema do caminho mínimo: o algoritmo de Dijkstra e o algoritmo de Floyd-Warshall, que serviram como objetos de análise. Todas as implementações foram realizadas em C#. A biblioteca desenvolvida funcionou como uma ferramenta para testar os algoritmos implementados, onde também foram executados algoritmos criados manualmente para verificar se cada implementação apresentava os resultados esperados.*

1. Introdução

O problema do caminho mínimo é um aspecto fundamental na teoria dos grafos e na ciência da computação. Trata-se de encontrar a rota mais curta ou com menor custo entre dois pontos em um grafo. Este problema é relevante em várias aplicações práticas, como na determinação da rota mais eficiente em redes de computadores, no planejamento

de rotas de transporte para minimizar o tempo ou custo de viagem, na otimização de processos logísticos, entre outros. Em redes de comunicação, por exemplo, encontrar o caminho mais curto pode reduzir o tempo de transmissão de dados e melhorar a eficiência geral.

Para resolver o problema do caminho mínimo, diferentes algoritmos podem ser empregados, dependendo das características do grafo e dos requisitos da aplicação. O algoritmo de Dijkstra é um dos algoritmos mais eficientes para solucionar problemas do caminho mínimo, porém ele se aplica somente para grafos cujas arestas tenham valores atribuídos a elas que sejam não negativos[NEGRI 2017].

Quando o objetivo é encontrar o caminho mais curto entre todos os pares de vértices, o algoritmo de Floyd-Warshall é uma opção, apesar de possuir maior complexidade. A eficiência na resolução do problema do caminho mínimo é crucial para economizar tempo e recursos, e a escolha do algoritmo adequado pode ter um impacto significativo nas soluções práticas e teóricas para esse problema.

No contexto deste atual trabalho, sendo este uma pesquisa acadêmica, concentra-se no estudo do problema do caminho mínimo e dos algoritmos utilizados para solucioná-lo. A pesquisa propõe-se a realizar um estudo detalhado desses algoritmos, examinando seus diferentes funcionamentos, e também incluirá a implementação prática destes, seguido de testes de desempenho. Partindo desse objetivo, também está inclusa a implementação de uma biblioteca de grafos. Estas implementações serão feitas utilizando a linguagem de programação C#.

O presente trabalho está organizado da seguinte forma. A Seção 2 apresenta os resultados de uma pesquisa feita filtrando outros trabalhos que tratam da utilização de grafos para lidar com distâncias, problema do caminho mínimo ou dos algoritmos estudados neste trabalho. Em seguida, a Seção 3 apresenta uma explicação detalhada e contextualizada do problema do caminho mínimo, enunciando os principais algoritmos utilizados na resolução do mesmo. Além disso, contém uma demonstração em pseudocódigo dos algoritmos estudados neste trabalho: Dijkstra e Floyd-Warshall. A Seção 4 apresenta os principais resultados obtidos no desenvolvimento deste trabalho. Por fim, a Seção 5 apresenta uma conclusão para o trabalho desenvolvido e as perspectivas futuras.

2. Trabalhos Relacionados

A teoria dos grafos e o problema do caminho mínimo são temas amplamente estudados na comunidade acadêmica. Como resultado, existem diversos trabalhos na área que exploram esses temas, no caso do problema do caminho mínimo, implementando diferentes algoritmos, como os algoritmos de Dijkstra e de Floyd-Warshall. Por essa razão, foi realizada uma busca por estudos que abordassem temas semelhantes ao deste trabalho.

[André Luiz Gonçalves Larrosa 2021] explica sobre o Problema do Caixeiro Viajante, um tópico diretamente relacionado ao estudo do problema do caminho mínimo, uma vez que o "caixeiro" deve encontrar a melhor rota entre os pontos. Neste, os autores citam sobre como a utilização de grafos pode ser útil para formular soluções, através dos algoritmos "Vizinho mais próximo" e "Inserção do mais distante".

O Problema do Caixeiro Viajante busca encontrar o ciclo de menor custo que visita um conjunto de cidades exatamente uma vez e retorna à cidade inicial. É um problema

NP-difícil, o que significa que encontrar uma solução exata para grandes instâncias é computacionalmente desafiador e não há algoritmos polinomiais conhecidos. Semelhante a ele, o problema de caminho mínimo procura o caminho mais curto entre dois vértices específicos em um grafo. Este problema é mais simples e pode ser resolvido de forma eficiente com algoritmos como Dijkstra ou Floyd-Warshall, que são adequados para encontrar rotas ótimas entre dois pontos ou de um ponto a todos os outros em grafos. Em resumo, o Caixeiro Viajante lida com a visita a todos os pontos e retorno ao início, enquanto o Caminho Mínimo se concentra em encontrar a rota mais curta entre dois pontos específicos.

Os algoritmos de caminho mínimo, quando citados por [Alicia Cavasso de Oliveira et al. 2020], mostram a eficiência de uma ferramenta da pesquisa operacional na otimização da rota de entregas de uma empresa do setor elétrico localizada no Município de Mauá envolvendo a aplicação e análise dos mesmos para melhorar a logística de distribuição. No caso em questão, a implementação do algoritmo de Dijkstra, revelou rotas mais eficientes, reduzindo o tempo de entrega e a distância percorrida em comparação com os métodos atualmente utilizados. A análise técnica incluiu a comparação entre as rotas otimizadas e as rotas tradicionais traçadas pelo Google Maps, avaliando métricas de desempenho como custo, tempo e distância.

Os pontos ressaltados pelos autores foram fundamentais para a escolha criteriosa dos algoritmos a serem implementados no desenvolvimento deste trabalho, que visa realizar o estudo do problema do caminho mínimo, utilizando uma biblioteca desenvolvida pelos próprios autores para comprovar a assertividade destes algoritmos em resolver o problema em questão. A análise aprofundada das técnicas discutidas pelos autores proporcionou uma base sólida para selecionar algoritmos que atendam às necessidades específicas do estudo, garantindo que a escolha seja fundamentada em critérios teóricos e experimentais. Ao longo do processo, foram considerados aspectos como complexidade computacional, capacidade de lidar com grandes volumes de dados e adequação dos algoritmos para diferentes tipos de grafos.

3. Estudo do problema do caminho mínimo em grafos

Esta seção aborda o Problema do Caminho Mínimo, um problema significativo na programação e matemática, com diversas aplicações em áreas como redes de computadores, telecomunicações e transportes. Contudo, há um número limitado de algoritmos disponíveis para sua solução.

Para representar a distância entre um vértice e outro, é necessário atribuir peso às arestas, como visto na Figura 1. Este peso pode ser interpretado como distância, tempo ou custo e é importante para determinar qual é a rota menos custosa a ser percorrida. No caso da figura, o caminho de menor custo tomando como origem o Nó 0 e como destino o Nó 6 é dado por $0 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 6$.

Dado um grafo ponderado $G = (V, E)$ com uma função de peso $w : E \rightarrow \mathbb{R}$ (onde $\mathbb{R}$ representa os números reais), o caminho mínimo de u a v é o caminho p que minimiza a soma dos pesos das arestas ao longo do caminho. Isto é:

$$\text{minimize} \sum_{(x,y)\in p} w(x, y) \tag{1}$$

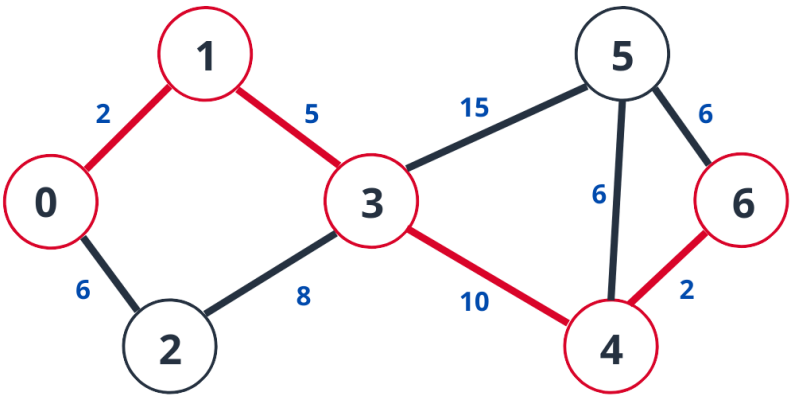

Figura 1. Exemplo de menor caminho.

Para encontrar o caminho mínimo em grafos ponderados existem alguns algoritmos que são comumente usados, descritos a seguir. Considere que G é o grafo, w é a função de peso, s é o vértice de origem, e h é a função heurística.

Algoritmo de Dijkstra Encontra o caminho mais curto de um vértice de origem para todos os outros vértices, com pesos não negativos.

$$\text{Dijkstra}(G, w, s)$$

Algoritmo de Bellman-Ford Encontra o caminho mais curto e pode lidar com pesos negativos, detectando também ciclos de peso negativo.

$$\text{Bellman-Ford}(G, w, s)$$

Algoritmo de Floyd-Warshall Calcula os caminhos mais curtos entre todos os pares de vértices.

$$\text{Floyd-Warshall}(G, w)$$

Algoritmo de A* O A* é um algoritmo de busca heurística que pode ser utilizado para encontrar o caminho mínimo entre dois pontos em um grafo, mas ao contrário de algoritmos como Dijkstra, que seguem um caminho mais determinístico para explorar os vértices, o A* combina a distância percorrida até o momento com uma função heurística que estima a distância restante até o destino [Junior 2021].

$$\text{A*}(G, w, s, h)$$

Embora o A* seja extremamente eficiente em muitos cenários, ele depende de uma função heurística que deve ser cuidadosamente definida e ajustada para o tipo específico de grafo e problema. Desta forma, este algoritmo exige um esforço adicional em desenvolver uma função heurística que seja admissível e garanta a eficiência do algoritmo.

Em vista de sua complexidade adicional que aumentaria o escopo do projeto, a implementação do A* foi descartada para manter a simplicidade e generalidade do estudo, evitando a necessidade de definir heurísticas específicas que poderiam não ser adequadas para todos os tipos de grafos. Em vista disso, o algoritmo de Dijkstra pode ser mais simples e adequado, pois não requer heurísticas e garante a solução ótima para todos os tipos de grafos com pesos não negativos.

Já o algoritmo de Bellman-Ford é menos eficiente em termos de tempo, sendo mais adequado para grafos com poucos vértices [Escola ALBK 2023]. Em muitos cenários onde o grafo não possui pesos negativos, sua implementação pode ser considerada desnecessariamente custosa em termos de desempenho. Por esse motivo, optou-se por não implementá-lo neste estudo. Devido a isso optou-se então pela utilização do algoritmo de Dijkstra e Floyd Warshall como objetos de estudo.

3.1. Algoritmo de Dijkstra

A essência do algoritmo, que pode ser consultado no Algoritmo 1, é baseada em explorar as distâncias mínimas dos vértices partindo de um ponto inicial. A ideia é ir acumulando as distâncias mínimas conforme os caminhos são percorridos. Em termos mais simples, o Dijkstra começa no ponto inicial e, a cada passo, expande o caminho para os nós vizinhos, escolhendo sempre o nó com a menor distância acumulada até aquele momento. O processo continua até que todos os vértices tenham sido visitados ou até encontrar o destino desejado [Júnior 2024].

Algoritmo 1: Algoritmo de Dijkstra

Entrada: Grafo $G = (V, E)$ com pesos não-negativos nas arestas, nó de origem s
Saída: Distância mínima de s para todos os nós de G

```
1  início
2      for cada nó v em V do
3          dist[v] ← ∞          ▷ Inicializa a distância de todos os nós como infinita
4      end
5      dist[s] ← 0              ▷ Distância da origem para ela mesma é 0
6      while V não está vazio do
7          u ← nó em V com a menor distância em dist[];
8          Remover u de V;
9          for cada vizinho v de u do
10             if dist[u] + peso(u, v) < dist[v] then
11                 dist[v] ← dist[u] + peso(u, v);
12             end
13         end
14     end
15     return dist[]            ▷ Retorna o vetor de distâncias mínimas
16 fim
```

Explicação Passo a Passo

1. Inicialização das Distâncias (referente às linhas 2-4 do algoritmo):

- Linha 3: O algoritmo inicializa a distância de todos os nós v como infinito (∞). Isso é feito porque, inicialmente, assume-se que todos os nós estão a uma distância muito grande da origem.
- Linha 5: A distância do nó de origem s para ele mesmo é definida como 0, pois não há custo para permanecer no mesmo nó.

Estado inicial das distâncias (suponha que s seja o nó de origem):

2. Processamento dos Nós (referente às linhas 6-7 do algoritmo):

- Linha 6: O algoritmo executa um loop enquanto houver nós no conjunto V. Em cada iteração, o nó u com a menor distância atual é selecionado. Este é o nó que será processado a seguir.
- Linha 7: O nó u é removido do conjunto V, indicando que foi processado e não precisa mais ser revisitado.

3. Atualização das Distâncias dos Vizinhos (referente às linhas 9-12 do algoritmo):

- Linha 9: Para cada vizinho v do nó u, o algoritmo verifica se a distância de u para v através de u é menor do que a distância atual conhecida para v.
- Linha 11: Se a nova distância é menor, a distância para o nó v é atualizada para refletir o caminho mais curto encontrado até agora.

Exemplo 1

Considere o grafo da Figura 2:

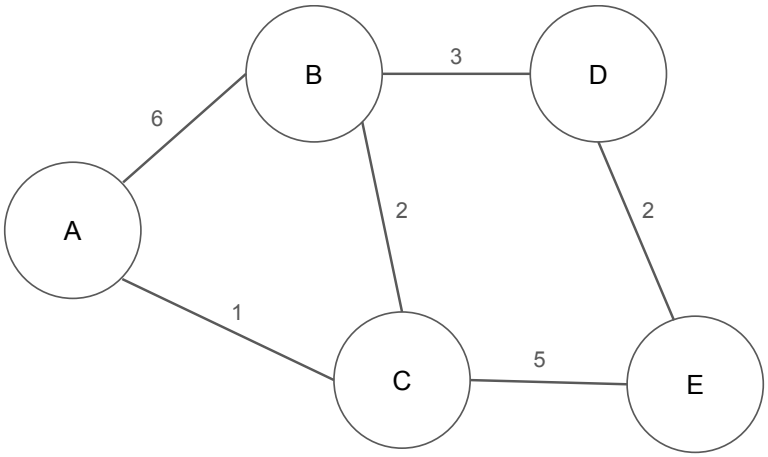

Figura 2. Grafo do Exemplo 1.

Tabela 1. Distâncias mínimas a partir do vértice A.

Vértice	Distância (de A)
A	0
B	6
C	1
D	8
E	6

Neste exemplo, $dist[i][j]$ fornece a distância mínima do nó A para os outros nós, assim como demonstra a Tabela 1 percorrendo o caminho completo. Ou seja, se o desejo for encontrar a distancia de A para E, o caminho escolhido sera 1+5 uma vez que:

- $A \rightarrow C = 1$
- $C \rightarrow E = 5$

3.1.1. Complexidade

O desempenho do algoritmo de Dijkstra depende do número de vértices $|V|$ e arestas $|E|$ do grafo. Desta forma sua complexidade se dá pela Equação 2 [Penna 2021].

$$O\left((|V| + |E|) \cdot \log |V|\right) \tag{2}$$

3.2. Algoritmo de Floyd-Warshall

O Floyd-Warshall, apresentando no Algoritmo 2, é um algoritmo de programação dinâmica que resolve o problema de encontrar o caminho mais curto entre todos os pares de vértices em um grafo. Isso significa que, dado um grafo com um conjunto de vértices e arestas com pesos, o algoritmo calcula a menor distância entre cada par de vértices e a respectiva rota [Carvalho 2024].

Algoritmo 2: Algoritmo de Floyd-Warshall

Entrada: Matriz de pesos W de um grafo com n vértices
Saída: Matriz de distâncias mais curtas D

```
1  início
2      for i ← 1 to n do
3          for j ← 1 to n do
4              if i = j then
5                  D[i][j] ← 0;
6              end
7              else
8                  D[i][j] ← W[i][j];
9              end
10         end
11     end
12     for k ← 1 to n do
13         for i ← 1 to n do
14             for j ← 1 to n do
15                 D[i][j] ← min(D[i][j], D[i][k] + D[k][j]);
16             end
17         end
18     end
19     return D;
20 fim
```

Exemplo 2

Considere o grafo da Figura 3:

Explicação Passo a Passo

1. Inicialização da Matriz de Distâncias D (referente às linhas 2-9 do algoritmo):

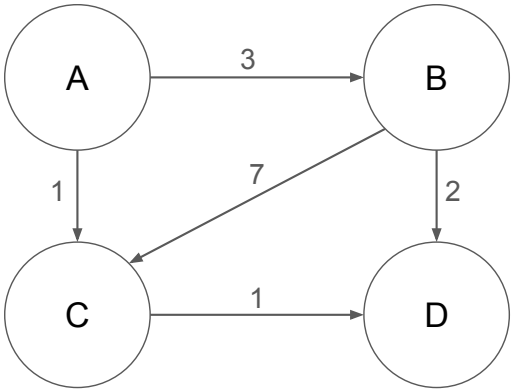

Figura 3. Grafo do Exemplo 2.

- Linha 2-3: Para cada vértice i: O primeiro loop percorre todos os vértices i no grafo. Para cada vértice j: O segundo loop percorre todos os vértices j para cada vértice i, criando uma matriz n×n.
- Linha 4-9: Para cada par de vértices (i, j), se $i = j$, então $D[i][j]$ é definido como 0 (linha 5). Caso contrário, $D[i][j]$ é definido como $W[i][j]$ (linha 8).

  Matriz inicial D:

$$D = \begin{bmatrix} 0 & 3 & 1 & \infty \\ \infty & 0 & 7 & 2 \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{bmatrix}$$

  Aqui, ∞ indica que não há uma aresta direta entre os vértices.

  2. Atualização da Matriz de Distâncias (referente às linhas 12-18 do algoritmo):

- Linha 12: O algoritmo considera cada vértice k como intermediário e atualiza a matriz de distâncias D.
- Linha 13-14: Para cada par de vértices (i, j), a distância $D[i][j]$ é atualizada para o menor valor entre a distância direta atual $D[i][j]$ e a soma das distâncias passando pelo vértice intermediário k (ou seja, $D[i][k] + D[k][j]$) (linha 15).

Iterações

- Para $k = 1$ (considerando A como intermediário):

$$D = \begin{bmatrix} 0 & 3 & 1 & \infty \\ \infty & 0 & 7 & 2 \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{bmatrix}$$

- Para $k = 2$ (considerando B como intermediário):

$$D = \begin{bmatrix} 0 & 3 & 1 & 5 \\ \infty & 0 & 7 & 2 \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{bmatrix}$$

- Para $k = 3$ (considerando C como intermediário):

$$D = \begin{bmatrix} 0 & 3 & 1 & 2 \\ \infty & 0 & 7 & 2 \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{bmatrix}$$

- Para $k = 4$ (considerando D como intermediário):

$$D = \begin{bmatrix} 0 & 3 & 1 & 2 \\ \infty & 0 & 7 & 2 \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{bmatrix}$$

Assim, após a execução do algoritmo, a matriz D, apresentada na Tabela 2, passa a representar as menores distâncias entre todos os pares de vértices.

Tabela 2. Matriz de distâncias entre os vértices.

D =

Vértice	A	B	C	D
A	0	3	1	2
B	∞	0	7	2
C	∞	∞	0	1
D	∞	∞	∞	0

3.2.1. Complexidade

Segundo [Cardoso 2021], a complexidade do algoritmo de Floyd-Warshall baseia-se no fato de que para cada vértice intermediário k de 1 a N, atualiza-se a distância de cada par de vértices (u, v). Logo, a complexidade do algoritmo se da pela Equação 3.

$$O(N^3) \tag{3}$$

4. Avaliação Experimental

Nesta seção, serão apresentados primeiramente a metodologia utilizada para a coleta dos resultados, seguido dos resultados em si do desenvolvimento do trabalho.

4.1. Metodologia

Para que fosse possível a execução dos algoritmos de Floyd Warshall e Dijkstra, seria necessário utilizar uma biblioteca de grafos já existente, ou a implementação de uma nova. Por motivos didáticos e de simplificação, foi definido então a implementação de uma nova biblioteca de grafos, que se encontra disponível no repositório do Github[1].

[1]https://github.com/luisbrasil/TCC-Caminho-Minimo-2024

4.1.1. Biblioteca de Grafos

A biblioteca de grafos, desenvolvida em C#, conta com 3 principais classes: **Grafo**, **No** (Nó) e **Aresta**. Estas classes estão estruturadas de maneira que a classe **Grafo** contém como propriedade uma lista de objetos da classe **No**, esta que, tem como propridades **Id**, **Valor** (que pode ser referido como peso) e uma lista de objetos da classe **Aresta**. Por fim a classe **Aresta** tem em sua declaração um atributo **Peso**, **NoInicial** (que guarda a referência na memoria do seu nó de origem) e **NoFinal** (que guarda a referência na memoria do seu nó destino).

No intuito de complementar a utilização da biblioteca, na classe Grafo, existem algumas funções. A função **GerarGrafoAleatorioInteiros** recebe como parâmetro um inteiro referente à quantidade de nós, e retorna um objeto da classe Grafo, com uma lista de nós de acordo com a quantidade especificada e valores aleatórios para cada um destes, além disso também monta aleatoriamente as arestas do grafo e atribuí na lista de arestas de cada nó. O método **ImprimirGrafo** recebe como parâmetro um objeto da classe **Grafo** e imprime no console cada nó e suas respectivas arestas.

A biblioteca é capaz de suportar qualquer tipo de grafo, sejam eles conexos ou desconexos:

Grafo Conexo É aquele onde existe um ou mais caminhos que nos levam a qualquer vértice do grafo, independente de qual seja o ponto de partida.

Grafo Desconexo É aquele onde existe pelo menos um vértice que não pode ser conectado a outro por um caminho.

Considerando o escopo do estudo deste trabalho, tal atribuição se torna mais do que o ideal, já que no problema do caminho mínimo considera-se somente grafos conexos, uma vez que os grafos desconexos podem resultar em caminhos impossíveis.

4.1.2. Algoritmos

Foram implementado os algoritmos de Floyd Warshall e de Dijkstra em C#. Para isso, a biblioteca de grafos foi utilizada para a geração dos grafos que serviram de entrada para os algoritmos implementados. Desta forma, foi considerado que o peso de uma Aresta refere-se à distância entre seu nó de origem de seu nó final.

4.2. Resultados

Foram criados grafos manualmente, primeiramente sua representação gráfica através do diagrama de pontos, e posteriormente criados usando a biblioteca. Estes grafos foram criados no intuito de que fosse possível aferir a assertividade do resultado dos algoritmos. O grafo ilustrado na Figura 4 foi criado no código usando a biblioteca de grafos executado em ambos os algoritmos, Dijkstra e Floyd-Warshall. Foi tomado como nó inicial o A e o nó final sendo o H, os dois algoritmos resultaram na distância mínima de 6 e o caminho apresentado na Figura 5. Os resultados correspondem ao esperado, indicando que, os algoritmos foram implementados corretamente.

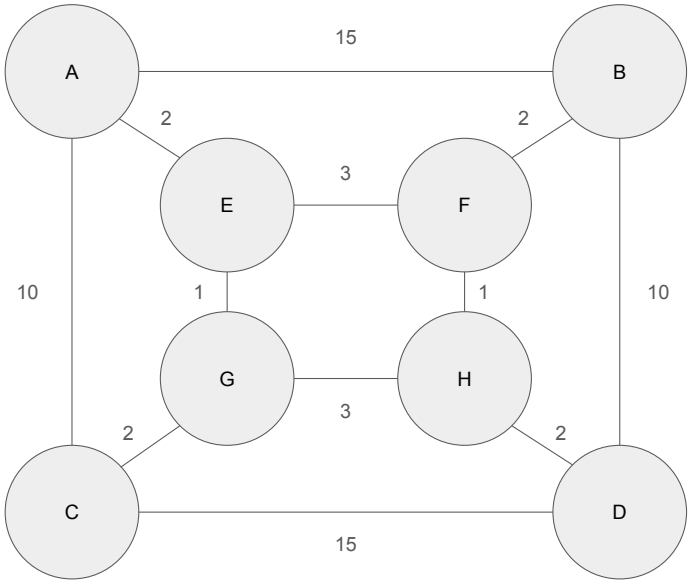

Figura 4. Grafo criado manualmente.

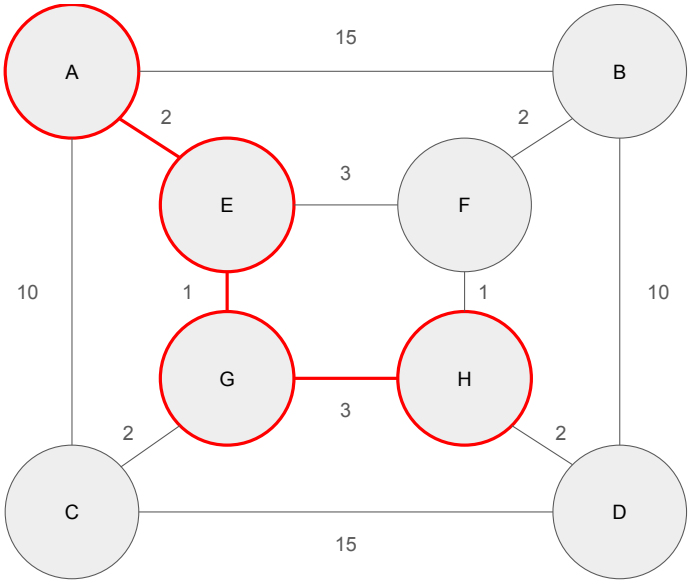

Figura 5. Resolução do caminho mínimo do grafo criado manualmente.

5. Conclusão e Perspectivas Futuras

O presente estudo abordou o problema do caminho mínimo, um problema central na teoria dos grafos, com aplicações práticas em diversas áreas, como logística e redes de comunicação. Os algoritmos de caminho mínimo desempenham um papel fundamental em otimizar rotas, reduzir custos e aumentar a eficiência em diferentes contextos.

No desenvolvimento do trabalho, foram implementados os algoritmos de Dijkstra e Floyd-Warshall, utilizando uma biblioteca própria desenvolvida em C#. Esses algoritmos foram testados tanto em grafos manuais quanto em grafos gerados aleatoriamente, o que permitiu uma análise detalhada de seus desempenhos e a validação dos resultados.

O trabalho contribuiu com uma biblioteca funcional em C# para a manipulação

de grafos e a implementação dos algoritmos estudados. Futuros estudos poderiam incluir a implementação de outros algoritmos, como Bellman-Ford e A*, desta forma seria possível uma comparação mais informativa sobre os diferentes algoritmos de caminho mínimo. Além disso, pretende-se uma análise estatística aperfeiçoada de resultados de desempenho.

Referências

Alicia Cavasso de Oliveira, A. K. G. M., Carvalho, G. S., and Giusti, R. (2020). Aplicação do conceito de caminho mínimo em uma empresa de pequeno porte através do algoritmo de dijkstra. Acesso em: 24 set. 2024. Disponível em: https://fateclog.com.br/anais/2020/APLICA%C3%87%C3%83O%20DO%20CONCEITO%20DE%20CAMINHO%20M%C3%8DNIMO%20EM%20UMA%20EMPRESA%20DE%20PEQUENO%20PORTE%20ATRAV%C3%89S%20DO%20ALGORITMO%20DE%20DIJKSTRA.pdf.

André Luiz Gonçalves Larrosa, E. H. M. C. (2021). Grafos aplicados ao problema do caixeiro viajante. *IFPR*. Acesso em: 24 de setembro de 2024.

Cardoso, L. (2021). Floyd-warshall. Acesso em: 24 de setembro de 2024. Disponível em: https://noic.com.br/materiais-informatica/curso/menor-caminho/floyd-warshall/.

Carvalho, M. A. M. (2024). Bcc204 - teoria dos grafos. Acesso em: 24 de setembro de 2024. Disponível em: http://www.decom.ufop.br/marco/site_media/uploads/bcc204/07_aula_07.pdf.

Escola ALBK (2023). O que é algoritmo de caminho? Acessado em: 22 out. 2024. Disponível em: https://escolalbk.com.br/glossario/o-que-e-algoritmo-de-caminho/.

Junior, C. M. (2021). Algoritmo a* (a estrela). Acesso em: 24 de setembro de 2024. Disponível em: https://www.tabnews.com.br/CristianoMafraJunior/algoritmo-a-a-estrela.

Júnior, E. (2024). Algoritmo de dijkstra: entendendo o caminho mínimo em grafos ponderados. Acesso em: 24 de setembro de 2024. Disponível em: https://elemarjr.com/clube-de-estudos/artigos/algoritmo-de-dijkstra-entendendo-o-caminho-minimo-em-grafos-ponderado

NEGRI, M. A. S. (2017). Caminhos em um grafo e o algoritmo de dijkstra. *UFSC*. Acesso em: 26 de setembro de 2024. Disponível em: https://repositorio.ufsc.br/xmlui/handle/123456789/183409.

Penna, P. H. (2021). Algoritmo de dijkstra. Acesso em: 24 de setembro de 2024. Disponível em: http://desenvolvendosoftware.com.br/algoritmos/grafos/algoritmo-de-dijkstra.html.