Journal of the Brazilian Computer Society, 2025

Implementação de Grafos - Trabalho 01

André Luís [**Pontifícia Universidade Católica de Minas Gerais**]
Breno Pires [**Pontifícia Universidade Católica de Minas Gerais**]
Caio Diniz [**Pontifícia Universidade Católica de Minas Gerais**]
Giuseppe Cordeiro [**Pontifícia Universidade Católica de Minas Gerais**]
Vinícius Miranda [**Pontifícia Universidade Católica de Minas Gerais**]
Yan Sabarense [**Pontifícia Universidade Católica de Minas Gerais**]

Instituto de Ciências Exatas e Informática, Pontifícia Universidade Católica de Minas Gerais, Rua Dom José Gaspar, 500, Coração Eucarístico, Belo Horizonte, BH, 30535-901, Brasil.

Resumo. Este trabalho explora diferentes estratégias para identificação de ciclos em grafos não direcionados. Implementamos e comparamos duas abordagens principais: uma baseada em permutação de vértices e outra utilizando busca em profundidade (DFS). O desempenho de ambos os métodos é analisado à medida que o tamanho do grafo aumenta. Nossos resultados indicam que, enquanto a abordagem baseada em permutação se torna inviável para grafos grandes devido à sua complexidade fatorial, o método baseado em DFS se escala de forma mais eficiente, tornando-se a opção preferida para aplicações práticas.

Keywords: Teoria dos Grafos, Detecção de Ciclos, Análise de Algoritmos, Caminhamento, Busca baseada em Permutação, DFS

1 Introdução

Algoritmos baseados em grafos são utilizados em diversas áreas para resolver uma ampla gama de problemas. Um grafo pode ser definido como $G = (V, E)$, onde V representa o conjunto de vértices e E o conjunto de arestas. A Figura 1 ilustra um grafo não-direcionado simples, ou seja, sem loops e sem arestas paralelas.

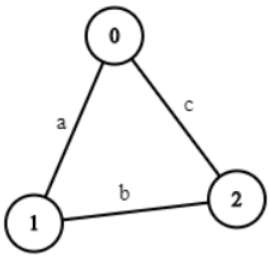


Figure 1. Exemplo de grafo simples de 3 vértices e 3 arestas.

Um dos conceitos fundamentais em grafos é o ciclo, que corresponde a um caminho fechado sem repetição de vértices. Formalmente, um ciclo é uma sequência $(v_0, a_1, v_1, a_2, v_2, \ldots, a_k, v_k)$ com $k > 2$, onde $v_k = v_0$ e os vértices intermediários são distintos dois a dois. Em grafos simples, um ciclo pode ser representado apenas pela sequência de vértices, uma vez que há no máximo uma aresta entre qualquer par de vértices.

A identificação de ciclos em grafos é uma tarefa fundamental em diversas aplicações práticas. Em redes de computadores, a detecção de ciclos pode ser utilizada para evitar loops de roteamento redundantes. Na biologia computacional, a identificação de ciclos em redes metabólicas pode fornecer informações sobre padrões de reações químicas em organismos vivos. Já na análise de redes sociais, ciclos podem indicar comunidades ou grupos interconectados de usuários. Essas aplicações evidenciam a importância de estratégias eficientes para detectar ciclos em diferentes contextos.

No entanto, a enumeração de todos os ciclos em um grafo apresenta desafios computacionais significativos. A complexidade do problema cresce rapidamente à medida que o número de vértices e arestas aumenta. Métodos ingênuos, como a permutação de vértices, sofrem com o crescimento fatorial do número de combinações possíveis, tornando a abordagem inviável para grafos grandes. Por outro lado, abordagens baseadas em busca estruturada, como a busca em profundidade (DFS), oferecem alternativas mais eficientes ao explorar caminhos de forma controlada.

O problema de enumerar todos os ciclos existentes em um grafo pode ser resolvido por diferentes abordagens. Neste trabalho, implementaremos e compararemos duas estratégias distintas para esse problema:

1. **Abordagem baseada em permutação**: Esta estratégia consiste em gerar todas as possíveis ordens dos vértices e verificar quais delas formam ciclos válidos. Apesar de garantir a identificação exaustiva de todos os ciclos, essa abordagem apresenta complexidade fatorial, tornando-a inviável para grafos grandes.
2. **Abordagem baseada em caminhamento (DFS)**: Nesta técnica, utilizamos a busca em profundidade (DFS) para explorar o grafo e detectar ciclos quando

um vértice já visitado é encontrado novamente. Essa abordagem é mais eficiente em termos computacionais e adequada para grafos de maior escala.

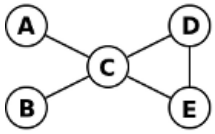
Além das implementações, será realizada uma análise comparativa entre essas abordagens para avaliar seu desempenho à medida que o tamanho do grafo aumenta.

2 Metodologia

Para comparar as abordagens de detecção de ciclos em grafos, implementamos dois métodos distintos: um baseado em permutação de vértices e outro utilizando busca em profundidade (DFS).

2.1 Estrutura de Dados

Os grafos foram representados utilizando matrizes de adjacência, onde cada posição (i, j) da matriz contém um valor binário indicando a presença ou ausência de uma aresta entre os vértices i e j. Essa escolha simplifica a manipulação e verificação de conexões durante a execução dos algoritmos.



	A	B	C	D	E
A	0	0	1	0	0
B	0	0	1	0	0
C	1	1	0	1	1
D	0	0	1	0	1
E	0	0	1	1	0

Figure 2. Representação de uma matriz de adjacência.

2.2 Algoritmos Implementados

As implementações apresentadas utilizam a matriz de adjacência como estrutura de dados para representar grafos. Essa abordagem permite verificar a existência de uma aresta entre dois vértices em tempo constante $O(1)$, embora o uso de memória seja $O(|V|^2)$, tornando-se ineficiente para grafos esparsos.

2.2.1 Abordagem baseada em Permutação

A abordagem de permutação explora todas as possíveis sequências de vértices do grafo e verifica quais delas formam ciclos válidos. Essa abordagem se baseia na geração exaustiva de todas as combinações possíveis de vértices, garantindo que nenhuma possibilidade de ciclo seja perdida. No entanto, essa técnica apresenta uma complexidade fatorial $O(|V|!)$, tornando-se rapidamente inviável para grafos com um número elevado de vértices. O algoritmo consiste nos seguintes passos:

1. Gerar todas as combinações possíveis de subconjuntos de vértices com tamanho mínimo de 3.
2. Para cada subconjunto gerado, listar todas as possíveis permutações dos vértices dentro dele.
3. Para cada permutação, verificar se a sequência de vértices forma um ciclo, garantindo que os vértices intermediários sejam distintos e que a última aresta retorne ao vértice de origem.

2.2.2 Abordagem baseada em Caminhamento

A abordagem baseada em Caminhamento (DFS - Depth-First Search) percorre o grafo recursivamente e detecta ciclos quando um vértice já visitado é encontrado novamente e não representa apenas a aresta reversa para seu pai. Essa técnica se destaca por sua eficiência em comparação com a abordagem baseada em permutação, pois evita a geração exaustiva de todas as sequências possíveis de vértices. Como resultado, a complexidade do algoritmo se mantém mais próxima de $O(|V|^2)$, tornando-o escalável para grafos de grande porte. O algoritmo segue os passos abaixo:

1. Para cada vértice do grafo, iniciar uma busca em profundidade.
2. Durante a exploração, manter um registro dos vértices visitados e verificar se um vértice já explorado é alcançado novamente sem ser o pai direto da busca, garantindo que a aresta forme um ciclo real e não apenas um retorno imediato.
3. Caso um ciclo seja encontrado, identificar corretamente seus vértices e armazená-lo, evitando a contagem duplicada de ciclos equivalentes.
4. Após a execução do algoritmo, exibir a quantidade total de ciclos distintos encontrados.

3 Resultados e Discussão

Os resultados confirmam as previsões teóricas sobre a complexidade das abordagens. Enquanto a permutação de vértices se mostrou impraticável devido à explosão combinatória, a busca em profundidade provou ser uma alternativa viável para aplicações práticas. Dessa forma, para problemas de detecção de ciclos em grafos grandes, a abordagem baseada em DFS é a melhor opção em termos de eficiência computacional.

3.1 Funcionamento da Implementação em Grafos Não Direcionados

A implementação baseada na permutação de vértices funciona corretamente para grafos não direcionados, pois a matriz de adjacência é preenchida de maneira simétrica. Sempre que uma aresta *(u, v)* é adicionada, tanto *adjMatrix[u][v]* quanto *adjMatrix[v][u]* são marcadas como verdadeiras. Dessa forma, a verificação da existência de ciclos leva em consideração a bidirecionalidade das conexões, garantindo que todos os ciclos presentes no grafo sejam corretamente identificados.

Por outro lado, a abordagem baseada na busca em profundidade (DFS) também se adapta bem a grafos não direcionados, uma vez que a exploração dos caminhos leva em conta a bidirecionalidade das arestas ao visitar os vértices. A diferença principal entre as duas implementações está na eficiência: enquanto a permutação é computacionalmente inviável para grafos grandes, a DFS apresenta um desempenho significativamente melhor para a detecção de ciclos em estruturas complexas.

4 Conclusão

Neste trabalho, exploramos diferentes abordagens para a identificação de ciclos em grafos. A utilização de matrizes de adjacência proporcionou uma implementação eficiente, permitindo representar as relações entre os vértices de forma direta. Comparando os métodos de permutação e caminhamento, observamos que a escolha da abordagem depende do tamanho do grafo e da complexidade computacional desejada.

Declaração

Contribuição dos Autores

Todos os autores contribuíram igualmente para a conceitualização, implementação e avaliação das abordagens apresentadas neste trabalho. André Luís e Vinícius Miranda implementaram a estrutura de dados de matriz de adjacência. Caio Diniz e Breno Pires desenvolveram o algoritmo de permutação. Giuseppe Cordeiro e Yan Sabarense foram responsáveis pelo caminhamento. Além disso, Vinícius Miranda ficou encarregado da documentação em LaTeX.

Disponibilidade de Dados e Materiais

O código-fonte deste trabalho pode ser acessado em `https://github.com/sabarense/GraphCycleEnumeration`. Trata-se de uma implementação desenvolvida para fins acadêmicos.