O Desafio do Circuito Hamiltoniano: Um Percurso Pela Complexidade

Projeto e Análise de Algoritmos (PAA)

Leonardo Nogueira Matos

Almeida Ítalo
Marcos Vinícius de Santana
Victor Caetano

PROCC
Programa de Pós-Graduação em Ciência da Computação/UFS

UNIVERSIDADE FEDERAL DE SERGIPE

Roteiro

1. Introdução: *O circuito em um grafo*
2. O Problema: *Definições e complexidade*
3. O Algoritmo: *Uso do Backtracking*
4. Exemplo Prático: *Código, execução e análise*
5. Conclusão e Aplicações

Introdução

Um circuito euleriano é caracterizado pelo fato de incluir todas as arestas de um grafo, uma única vez.

Porém, os vértices podem se repetir nesse circuito.

Já em um circuito hamiltoniano, em um circuito fechado tanto as arestas como os vértices são incluídos uma única vez.

Nota: um circuito em um grafo se dá quando todos os vértices são percorridos começando e terminando no mesmo vértice.

O problema

Definição de Grafo:

Dado um grafo G=(V,E), onde V é o conjunto de vértices e E é o conjunto de arestas.

Caminho Hamiltoniano: É um caminho que visita cada vértice $v \in V$ exatamente uma vez. (Figura 01)

Circuito Hamiltoniano: É um caminho Hamiltoniano que também é um ciclo, ou seja, existe uma aresta em E que conecta o último vértice do caminho ao primeiro. (Figura 02)

Figura 01

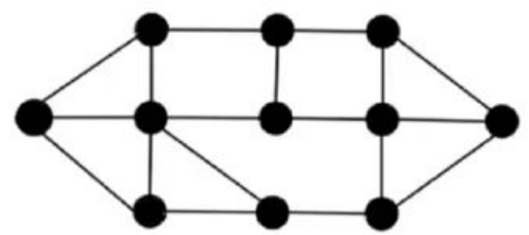
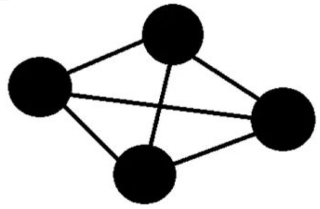


Figura 02



O problema

O grande desafio: A complexidade Computacional

O problema de decidir se um grafo possui um Circuito Hamiltoniano é NP-Completo.

Não há algoritmo eficiente conhecido: Não existe uma solução em tempo polinomial (O(nk)) para resolver todos os casos.

Explosão Combinatória: O tempo de execução de algoritmos de força bruta cresce de forma fatorial (O(n)), tornando-os impraticáveis para grafos de tamanho moderado.

Contraste: É um problema fundamentalmente "mais difícil" do que encontrar uma Árvore Geradora Mínima (que pode ser resolvida eficientemente com algoritmos gulosos).

Como Resolver o problema?

Estratégia Escolhida:

Para encontrar uma solução exata, usamos uma busca em profundidade (DFS) inteligente chamada **Backtracking.**

Ideia Central:

1. Construir: Adicione um vértice ao caminho.

2. Explorar: Avance recursivamente para um vizinho ainda não visitado.

3. Verificar: Se chegar a um beco sem saída, o caminho atual não funciona.

4. Voltar Atrás (Backtrack): Desfaça o último passo (remova o vértice do caminho) e tente uma alternativa.

5. Sucesso: Se o caminho incluir todos os vértices e puder fechar o ciclo, uma solução foi encontrada.

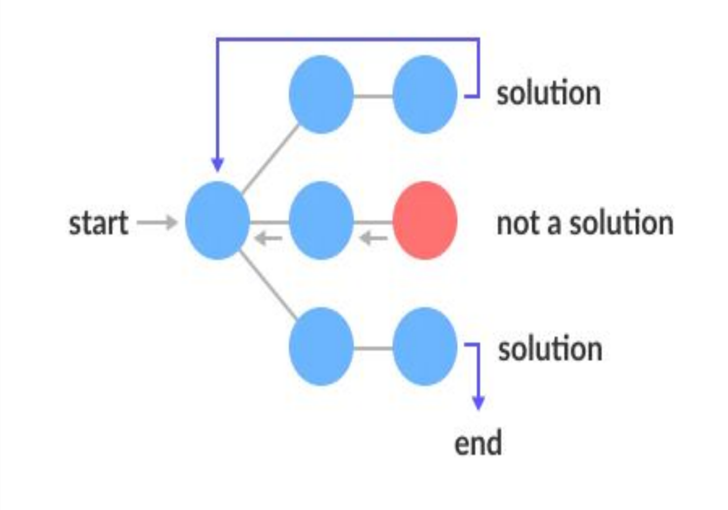


Figura 03

O Algoritmo em Ação (Demonstração Visual)

Passo a Passo

Início: Caminho = [A]

Avanço 1: Caminho = [A, B]

Beco Sem Saída: De C, o caminho para D não leva a uma solução.

BACKTRACK: Remove D. Volta para C. Caminho = [A, B, C]

Nova Tentativa: De C, tenta o vizinho E. Caminho = [A, B, C, E] ..
(continuar até encontrar a solução)

Solução: Caminho = [A, B, C, E, D]. Verifica se D conecta com A.
Sim! Circuito encontrado.

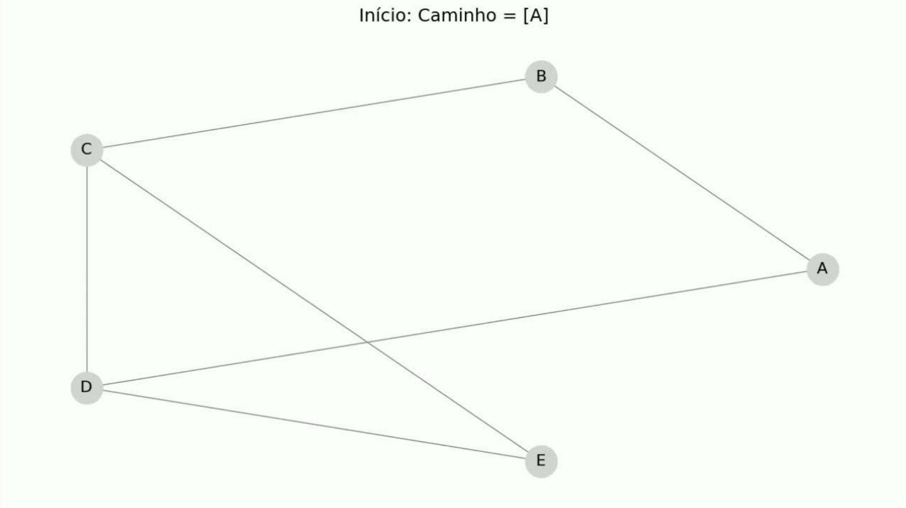


Figura 04

Da Lógica ao Código

O coração do nosso algoritmo está nesta função recursiva, a **solve_util**. Ela reflete exatamente os três passos do Backtracking que acabamos de ver:

1. **Primeiro, o Caso Base:** Onde o código verifica se já construímos um caminho completo e se ele consegue fechar o ciclo. Se sim, ele retorna **True**, indicando que encontramos a solução.
2. **Segundo, a Exploração:** Que é um loop que tenta adicionar cada vértice vizinho válido, chamando a si mesma.
3. **E o mais importante, o Backtrack:** Se uma escolha nos leva a um beco sem saída, a função 'desfaz' a última jogada e tenta a próxima alternativa no loop.

Conclusão

Para finalizar nossa apresentação, este slide resume as principais conclusões que tiramos deste projeto.

- **Validação:** Nosso algoritmo de Backtracking funcionou, encontrando a solução exata e provando que a teoria funciona na prática.
- **Custo:** A complexidade exponencial do algoritmo o torna inviável para grafos grandes, demonstrando a "explosão combinatória".
- **NP-Completo:** Este projeto mostra na prática o que é um problema NP-Completo: fácil de verificar, mas difícil de resolver.
- **Realidade:** Por isso, a indústria usa heurísticas e aproximações para obter soluções rápidas e "boas o suficiente" em problemas reais.

Referências

OLIVEIRA, Valeriano A.; RANGEL, Socorro. *Grafos Hamiltonianos* (Teoria dos Grafos). Ibilce – UNESP, 2014. Disponível em: <https://www.ibilce.unesp.br/Home/Departamentos/MatematicaAplicada/socorro4029/ghamiltoniano_rev2014.pdf >. Acesso em: 27 set. 2025.

TREVISAN, Ronaldo . *Grafos II*. Universidade Federal do Rio Grande do Sul (UFRGS). Disponível em: <http://www.mat.ufrgs.br/~trevisan/class/grafos2.pdf> . Acesso em: 27 set. 2025.

MASSUIA, Giovanny. Backtracking. *Craft & Code Club*, 25 maio 2025. Disponível em: <https://craftcodeclub.io/posts/dsa-backtracking>. Acesso em: 27 set. 2025.

MELANIE. Backtracking: What is it? How do I use it? *DataScientest*, 29 mar. 2024. Disponível em: <https://datascientest.com/en/backtracking-what-is-it-how-do-i-use-i>. Acesso em: 27 set. 2025.

KHITTHU, Haroon Ahamed. Your One-Stop Solution to Understand Backtracking Algorithm. *Simplilearn*. Última atualização: 2 dez. 2024. Disponível em: <https://www.simplilearn.com/tutorials/data-structure-tutorial/backtracking-algorithm>. Acesso em: 27 set. 2025.

Obrigado!