Rainbow Version of Dirac's Theorem: An Algorithmic Approach

Nathan Luiz Bezerra Martins
Willian Miura Mori

Orientadora: Yoshiko Wakabayashi

Departamento de Ciência da Computação,
Instituto de Matemática e Estatística, Universidade de São Paulo

Resumo

Dada uma coleção $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ de grafos de ordem $n \geq 3$, definidos sobre o mesmo conjunto de vértices e que satisfazem a condição de Dirac para cada G_i, existe um $\mathcal{G}$-transversal que forma um circuito hamiltoniano, também conhecido como Circuito Hamiltoniano Rainbow. Neste trabalho, desenvolvemos um algoritmo eficiente que encontra um Circuito Hamiltoniano Rainbow. Fizemos implementações tanto em `C++` quanto em `Python` e realizamos testes de desempenho para comparar as duas versões. Utilizamos a biblioteca `manim` para fazer uma animação gráfica do algoritmo.

Conceitos básicos

Definições principais:

· Um grafo simples G é um grafo não direcionado sem laços e sem arestas múltiplas, onde $V(G)$ e $E(G)$ são definidos, respectivamente, o conjunto de vértices e o conjunto de arestas desse grafo.

· Um **circuito hamiltoniano** de G é um circuito que visita cada vértice de G exatamente uma vez.

· Definimos $\deg(v)$ como o número de vizinhos de um vértice de v em $V(G)$, e $\delta(G) \coloneqq \min\{\deg(v) : v \in V(G)\}$ como o grau mínimo de um vértice em G.

Teorema de Dirac (1952): Se um grafo simples G com $n \geq 3$ vértices tem grau mínimo $\delta(G) \geq \frac{n}{2}$, então G contém um circuito hamiltoniano.

Versões rainbow de problemas clássicos

Definição:

A versão rainbow de um problema na teoria dos grafos é uma variação que adiciona a restrição de cores à solução desejada. Nesse contexto, o termo rainbow (arco-íris) refere-se a estruturas em um grafo que utilizam elementos provenientes de diferentes subconjuntos, ou arestas com diferentes rótulos ou cores, garantindo que não haja repetições.

Teorema de Dirac (*Versão Rainbow*):
Dada uma coleção $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ de grafos de ordem $n \geq 3$, definidos sobre o mesmo conjunto de vértices e que satisfazem a condição de Dirac para cada G_i, existe um $\mathcal{G}$-transversal que forma um circuito hamiltoniano, também conhecido como Circuito Hamiltoniano Rainbow. Cada grafo G_i pode ser enxergado como se suas arestas fossem coloridas com a cor i.

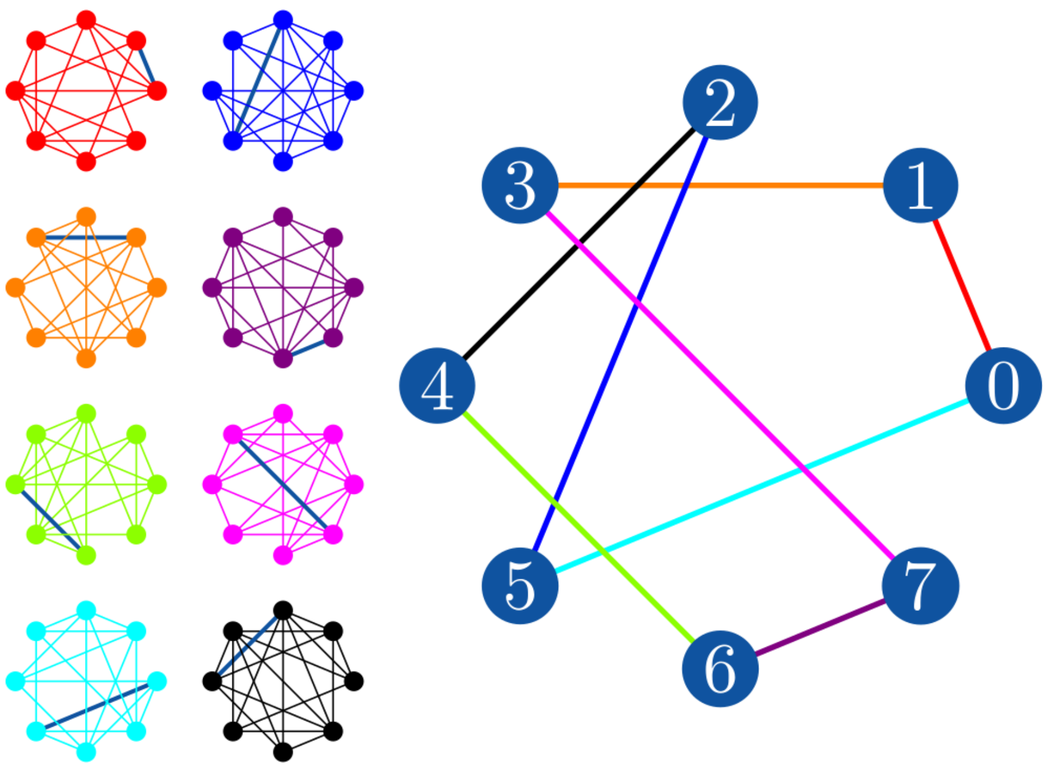

Figura 1: Exemplo de um Circuito Hamiltoniano Rainbow para uma coleção de grafos com 8 vértices.

Mais exemplos clássicos:

Floresta Geradora Mínima Rainbow: Dado um grafo G e uma coleção de cores, encontre uma floresta geradora mínima que utilize exatamente uma aresta de cada cor.

Emparelhamento Perfeito Rainbow: Dado um grafo bipartido G e uma coleção de cores, encontre um emparelhamento perfeito que utilize exatamente uma aresta de cada cor.

Conjunto Independente Rainbow: Dado um grafo G onde os vértices estão coloridos, encontre o maior conjunto independente de vértices (conjunto de vértices não adjacentes) tal que todas as cores nos vértices do conjunto sejam distintas.

Fluxograma do algoritmo

A Figura 2 mostra o fluxograma do algoritmo desenvolvido. A ideia principal é, dado um objeto, que pode ser um caminho ou um circuito, incrementar esse objeto para um objeto maior.

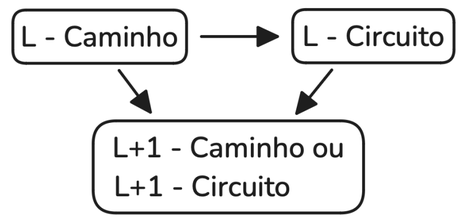

Figura 2: Fluxograma do algoritmo.

Técnicas utilizadas

Todas as técnicas utilizam fortemente a condição de Dirac para cada grafo.

Cruzamento:

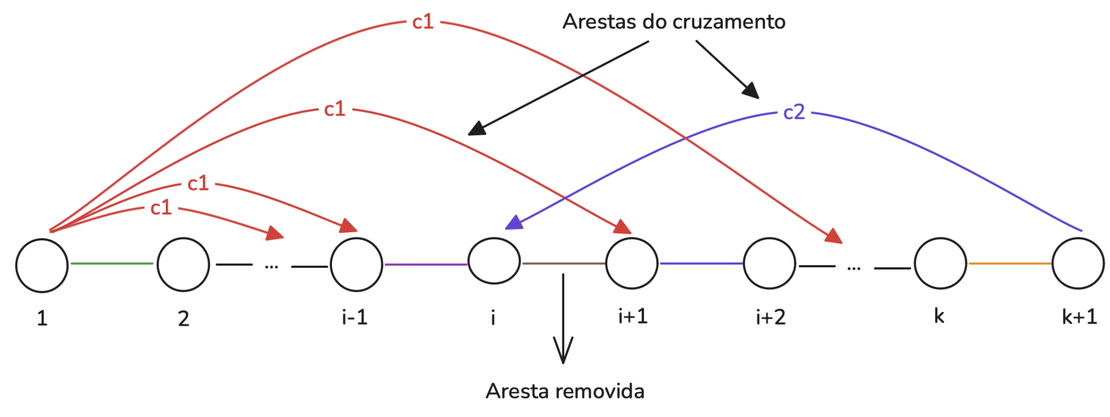

Figura 3: Construção de um circuito de tamanho $k+1$ a partir de um caminho de tamanho k.

Aresta fora do circuito:

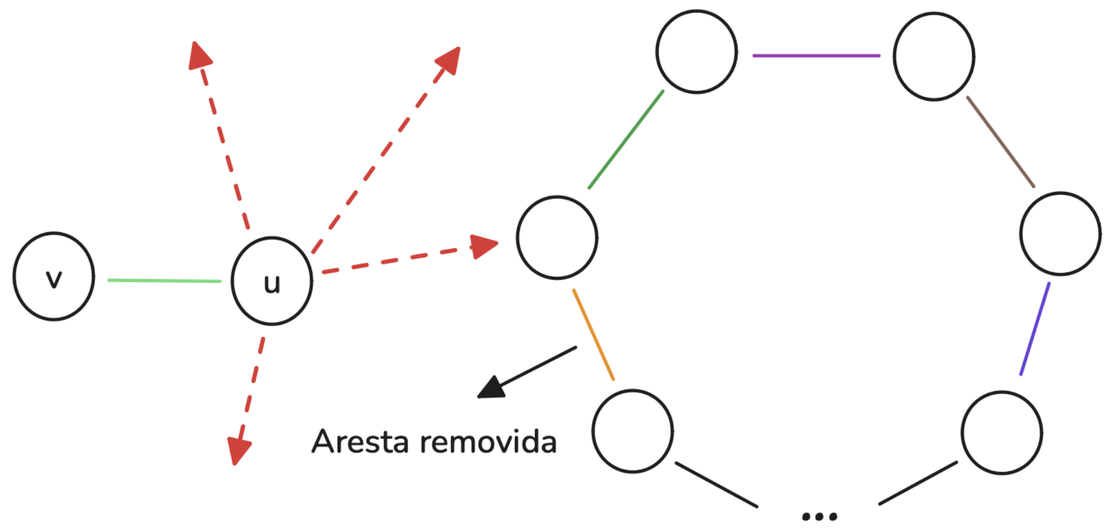

Figura 4: Construção de um caminho de tamanho $k+1$ a partir de um circuito de tamanho k.

Desafios enfrentados

Criação de uma prova construtiva:
A prova original de **Joos, Kim (2020)** é de natureza não construtiva. Em diversos momentos, o autor solicita que se considere o maior caminho ou circuito *rainbow* e, a partir disso, faz-se uma demonstração. No entanto, ele não explica como encontrar efetivamente esse caminho ou circuito.

Implementação do caso n - 1:
Esta é a parte mais difícil do algoritmo. Quase metade do código serve para lidar com esse caso, pois são muitos detalhes a serem considerados.

Implementação eficiente:
A complexidade do algoritmo é da ordem de $O(n^3)$, em que n é a ordem de G. Essa complexidade é ótima, pois existem $O(n^3)$ arestas que devem ser consideradas no input.

Animação

Animação demonstrando o algoritmo foram feitas com a biblioteca `manim` (do canal de Youtube `3Blue1Brown`) em Python. Um exemplo está disponível no QR Code ao lado.

Referências

As referências estão na página da monografia em:
`https://linux.ime.usp.br/~nathanluiz`