Impacto de Técnicas de Pré-Processamento em Algoritmo de *Branch-and-Price* para o Problema de Coloração de Grafos

Ieremies V. F. Romero[1], **Rafael C. S. Schouery**[1]

[1] Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Av. Albert Einstein, 1251 – Cidade Universitária, Campinas – SP, 13083-852

{ieremies,rafael}@ic.unicamp.br

Abstract. *This work investigates the vertex coloring problem, which has applications in scheduling and resource allocation. We describe a preprocessing step that uses separation and reduction techniques (k-core) to reduce the search space without compromising valid lower bounds. Experimental results with DIMACS instances demonstrate that preprocessing increases the number of solved instances, although in some cases, the order in which graph components are processed may hinder the ability to solve them. We conclude with suggestions for future optimizations, such as incorporating clique separators.*

Resumo. *Este trabalho investiga o problema de coloração de vértices, o qual possui aplicações em escalonamento e alocação de recursos. Descrevemos um pré-processamento que utiliza técnicas de separação e redução (k-core) para diminuir o espaço de busca sem comprometer limites inferiores válidos. Resultados experimentais com instâncias do DIMACS demonstram que o pré-processamento aumenta a quantidade de instâncias resolvidas, apesar de, em alguns casos, a ordem com que as partes do grafo são resolvidas por comprometer a capacidade de resolvê-lo. Concluimos com sugestões para trabalhos futuros, como a incorporação de separadores por clique.*

1. Introdução

O *problema de coloração de vértices*, frequentemente referido como o problema de coloração de grafos, é um problema clássico da teoria dos grafos que há décadas desperta o interesse dos pesquisadores. O objetivo do problema é atribuir cores aos vértices de um grafo de modo que nenhum par de vértices adjacentes compartilhe a mesma cor, minimizando ao mesmo tempo o número total de cores utilizadas.

O problema de decidir se um grafo admite uma coloração com k cores, para qualquer $k > 2$, é NP-difícil, e sua versão de otimização, ou seja, encontrar o menor valor de k para o qual existe uma coloração com k cores, é NP-completo [Garey and Johnson 1976]. Apesar dos avanços contínuos no desenvolvimento de algoritmos e no poder computacional, instâncias com apenas 100 vértices ainda representam desafios significativos para algoritmos exatos, evidenciando a complexidade persistente do problema. Compreender e resolver o problema de coloração de vértices não é apenas uma questão de interesse teórico. Ele surge naturalmente em diversas áreas, incluindo escalonamento de tarefas, alocação de recursos, atribuição de frequências e alocação de registradores em compiladores [Gamache et al. 2007, de Werra 1985, Caprara et al. 2007, Chow and Hennessy 1990, Smith et al. 1998].

Uma estratégia fundamental para reduzir significativamente o esforço computacional consiste em diminuir o espaço de busca, dividindo inteligentemente o problema em subproblemas menores e refinando-o com base em limites previamente obtidos. Esses métodos permitem tornar instâncias difíceis mais tratáveis, seja por meio da decomposição em subproblemas menores, seja pela simplificação da estrutura do grafo sem perder suas propriedades essenciais. A separação e a redução de instâncias são cruciais para mitigar o crescimento exponencial do tempo computacional inerente a problemas NP-difíceis, como a coloração de grafos.

Seja $G = (V, E)$ um grafo simples não direcionado, tal que $E \subseteq V \times V$. Dois vértices u e v são chamados de *adjacentes* se existir uma aresta $\{v, u\} \in E$. Uma *coloração própria* é uma atribuição de rótulos, chamados de *cores*, aos vértices, de modo que nenhum par de vértices adjacentes receba a mesma cor. Referimo-nos a uma coloração própria simplesmente como uma coloração. O *número cromático* de G, denotado por $\chi(G)$, é o menor número de cores necessário para obter uma coloração própria de G. Denominamos *classe de cor* o conjunto de vértices que recebem a mesma cor. Uma k-coloração de G é uma coloração própria de G com k classes de cor.

Para um grafo $G = (V, E)$, um conjunto $S \subseteq V$ é *independente* em G se, e somente se, não houver arestas em E conectando dois elementos de S. Observe que uma classe de cor é um conjunto independente em G. Um *subgrafo induzido* $H = (V_H, E_H)$ de $G = (V, E)$ é um grafo tal que $V_H \subseteq V$ e $E_H = \{\{u, v\} \in E : u, v \in V_H\}$. Em outras palavras, H é um subgrafo obtido pela seleção de um subconjunto dos vértices de G e pela inclusão de todas as arestas entre esses vértices que estão presentes em G. Denotamos por $G[V']$ o subgrafo induzido pelo conjunto $V' \subseteq V$.

O complemento de um grafo $G = (V, E)$ é o grafo $\overline{G} = (V, \overline{E})$, onde

$$\overline{E} = \{\{u, v\} : \{u, v\} \notin E,\ u, v \in V \text{ e } v \neq u\}.$$

Ou seja, dois vértices são adjacentes em $\overline{G}$ se, e somente se, não forem adjacentes em G.

2. Pré-processamento

Uma ideia fundamental é que, ao (re)introduzir vértices no grafo, o número cromático não pode ser reduzido. Isso significa que um limite inferior calculado em um subgrafo induzido ainda é um limite inferior válido para o grafo original. Portanto, seguindo essa lógica, ao remover vértices, podemos calcular limites inferiores válidos para o número cromático em um grafo menor.

Lema 2.1. Seja G um grafo e G' um subgrafo de G. Então, $\chi(G) \geq \chi(G')$.

2.1. Separação

Uma *componente conexa* de um grafo é um subgrafo maximal conexo, ou seja, um subgrafo em que há um caminho entre quaisquer dois vértices e que não está contido em nenhum subgrafo conexo maior. Uma observação simples é que o número cromático de um grafo G é igual ao maior número cromático entre suas componentes conexas, ou seja,

$$\chi(G) = \max\{\chi(H) : H \text{ é uma componente conexa de } G\}.$$

Isso significa que podemos separar o grafo em suas componentes conexas, resolver cada uma individualmente e então combinar as soluções.

Essa ideia pode ser explorada ainda mais ao alterar a ordem em que cada componente conexa é resolvida, com o objetivo de aumentar a eficácia das reduções descritas adiante (ver Seção 2.2). Ao priorizar componentes com mais vértices, buscamos obter limites inferiores que simplifiquem a resolução de componentes menores. Por outro lado, se começarmos pela menor componente conexa, podemos obter algum limite inferior — possivelmente menor do que obteríamos de outra forma —, mas acelerar a solução de componentes maiores, que geralmente são as mais difíceis. Testamos essas ideias mais adiante em nossos experimentos.

Outro lema útil decorre das componentes conexas do complemento do grafo.
Lema 2.2. Seja G um grafo, e sejam $\overline{G}_1, ..., \overline{G}_k$ as componentes conexas de $\overline{G}$. Então, $\chi(G) = \sum_{i=1}^{k} \chi(G_i)$.

Isso decorre do fato de que, para quaisquer duas componentes conexas $\overline{G}_i$ e $\overline{G}_j$ de $\overline{G}$, todos os vértices $v \in \overline{G}_i$ são adjacentes a todos os vértices $u \in \overline{G}_j$ no grafo original G. Portanto, nenhuma cor usada em uma componente conexa de $\overline{G}$ pode ser usada em outra componente conexa.

Por fim, observamos que essas separações podem ser aplicadas múltiplas vezes: ao separar o grafo por estar desconexo, ainda podemos verificar a condição de Lema 2.2 e repetir esse processo até que nenhuma outra separação seja possível. Ao reconstruir o grafo original, precisamos aplicar corretamente a regra adequada para unir as soluções das diferentes partes, de acordo com o critério utilizado para separá-las.

2.2. Redução

Inicialmente proposto por [Seidman 1983], o conceito de k-core pode ser aplicado para reduzir o número de vértices considerados ao resolver o problema de coloração.
Definição 2.3. Seja G' o subgrafo resultante da remoção iterativa de vértices com grau menor que k do grafo G. Cada componente conexa de G' é chamada de k-core de G.

O conceito de k-core está relacionado à *degenerescência* $\delta^*(G)$ de um grafo, definida como o maior valor de k tal que G contém uma k-core não vazia. Recomendamos ao leitor a consulta de [Malliaros et al. 2019] para um levantamento de algoritmos que aplicam essa ideia.

É importante notar que esse processo pode resultar em múltiplas componentes (k-cores), mas, como aplicamos as regras de separação explicadas anteriormente, consideramos o resultado como um único subgrafo conexo. Dado um limite inferior k para o número cromático de G, podemos encontrar a k-core de G em tempo polinomial.
Teorema 2.4. [Malliaros et al. 2019] Seja G um grafo e k um limite inferior para $\chi(G)$. Uma coloração de sua k-core pode ser estendida para uma coloração válida do grafo original com o mesmo número de cores em tempo polinomial.

Por fim, existem duas reduções bem conhecidas para o problema estudado. Primeiro, quando um vértice é *universal*, ou seja, adjacente a todos os outros vértices, ele pode ser removido e posteriormente atribuído a uma nova cor. Segundo, para um par de vértices não adjacentes u, v, se $N(u) \subseteq N(v)$, dizemos que o vértice v domina u. Quando isso ocorre, podemos remover o vértice u e posteriormente colori-lo com a mesma cor atribuída a v.

3. Experimentos

Para testar estas regras, implementamos a abordagem de *branch-and-price* proposta por [Mehrotra and Trick 1996] e o algoritmo de [Xiao et al. 2021] como base para a precificação. A implementação foi feita em C++17, compilado com GCC 11.4.0. Os experimentos foram realizados em um Intel Xeon CPU E5-2630 v4 @ 2.20GHz com 68Gb de RAM, rodando Ubuntu 22.04.3. Como resolvedor de programação linear inteira, utilizamos o Gurobi 11.0.3 [Gurobi Optimization, LLC 2024]. Todas as instâncias tiveram 1 hora como tempo limite.

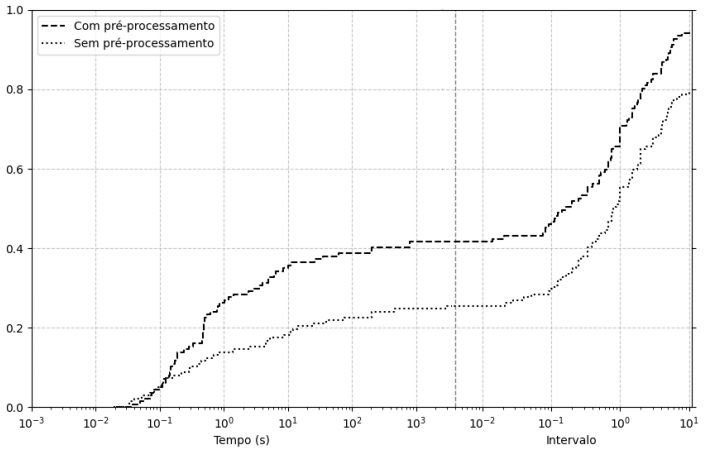

Figura 1. Distribuição acumulada de instâncias resolvidas por tempo e por intervalo (diferença entre limites superior e inferior).

Na Figura 1 pode ser observado o impacto das técnicas. Sem o pré-processamento aqui proposto, conseguimos resolver somente 36 das 137 instâncias do DIMACS [DIMACS 2002]. Já com o pré-processamento, este número aumentou para 58, uma considerável diferença.

Apesar disso, em 4 instâncias "mais difíceis", observamos ainda algumas diferenças significativas, em especial, na instância `myciel4`, onde a abordagem com pré-processamento demorou 61 segundos em contraste com os 39 segundos sem o método aqui proposto. Por fim, existem mais 2 instâncias na qual o método sem o pré-processamento conseguiu resolver a mais que o seu contraposto. Acreditamos que isso deve-se a uma questão de ordenação das componentes para obter limitantes inferiores.

4. Conclusões

Neste trabalho, apresentamos brevemente a severidade dos impactos de separação e redução quando se trata de problemas como a coloração de grafos. Observamos que o desempenho conjunto de tais técnicas nos permite atingir patamares de "performance" consideravelmente superiores, ainda mais em abordagens especialmente sensíveis à quantidade de conjuntos independentes, como a utilizada aqui.

Como trabalhos futuros, resta entender o que ocasionou esta disparidade de tempo nas instâncias onde o não pré-processamento se saiu muito melhor. Além disso, ideias como separador por clique [Tarjan 1985] podem ser incorporadas para ainda mais dividir o nosso espaço de busca.

Referências

Caprara, A., Kroon, L., Monaci, M., Peeters, M., and Toth, P. (2007). *Chapter 3 Passenger Railway Optimization*, pages 129–187. Elsevier.

Chow, F. C. and Hennessy, J. L. (1990). The priority-based coloring approach to register allocation. *ACM Transactions on Programming Languages and Systems*, 12(4):501–536.

de Werra, D. (1985). An introduction to timetabling. *European Journal of Operational Research*, 19(2):151–162.

DIMACS (2002). Graph coloring instances. `https://mat.tepper.cmu.edu/COLOR/instances.html`. [Accessed 31-03-2025].

Gamache, M., Hertz, A., and Ouellet, J. O. (2007). A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. *Computers Operations Research*, 34(8):2384–2395.

Garey, M. and Johnson, D. (1976). The complexity of near-optimal graph coloring. *Journal of the ACM*, 23(1):43–49.

Gurobi Optimization, LLC (2024). Gurobi Optimizer Reference Manual.

Malliaros, F. D., Giatsidis, C., Papadopoulos, A. N., and Vazirgiannis, M. (2019). The core decomposition of networks: theory, algorithms and applications. *The VLDB Journal*, 29(1):61–92.

Mehrotra, A. and Trick, M. A. (1996). A column generation approach for graph coloring. *INFORMS Journal on Computing*, 8(4):344–354.

Seidman, S. B. (1983). Network structure and minimum degree. 5(3):269–287.

Smith, D., Hurley, S., and Thiel, S. (1998). Improving heuristics for the frequency assignment problem. *European Journal of Operational Research*, 107(1):76–86.

Tarjan, R. (1985). Decomposition by clique separators. *Discrete Mathematics*, 55(2):221–232.

Xiao, M., Huang, S., Zhou, Y., and Ding, B. (2021). Efficient reductions and a fast algorithm of maximum weighted independent set. In *Proceedings of the Web Conference 2021*, WWW '21. ACM.