Em direção a uma nova abordagem para colorir os vértices de um grafo usando Busca Monte Carlo em Árvore

Raquel M. de Souza[1], **Fabiano de S. Oliveira**[1], **Paulo Eustáquio D. Pinto**[1], **Valmir C. Barbosa**[1,2]

[1]Programa de Pós-graduação em Ciências Computacionais
Universidade do Estado do Rio de Janeiro (UERJ) – Rio de Janeiro – RJ – Brasil

[2]Programa de Engenharia de Sistemas e Computacão
Universidade Federal do Rio de Janeiro (UFRJ) – Rio de Janeiro – RJ – Brasil

raquel.souza@pos.ime.uerj.br, fabiano.oliveira@ime.uerj.br,

pauloedp@ime.uerj.br, valmircbarbosa@gmail.com

Abstract. *This paper introduces a new approach to vertex coloring in graphs. This approach consists of randomly adding edges to the graph, aiming to find a supergraph whose chromatic number is close to that of the original graph and requires low computational time to be computed. To guide the random process, we employ Monte Carlo Tree Search. We have found preliminary evidence in favor of the proposed approach.*

Resumo. *Este artigo apresenta uma nova abordagem para coloração de vértices em grafos. Essa abordagem consiste em adicionar arestas ao grafo aleatoriamente, na expectativa de encontrar um supergrafo cujo número cromático seja próximo ao do grafo original e requeira tempo computacional baixo para ser computado. O processo aleatório é guiado pela Busca Monte Carlo em Árvore. Apresentamos evidências preliminares em favor dessa abordagem.*

1. Introdução

O problema da coloração de vértices em grafos é um dos problemas NP-difíceis mais conhecidos. Em sua versão de decisão, consta da famosa lista dos primeiros 21 problemas NP-completos identificados [Karp 1972]. Dado um grafo G, o problema pede que se atribua uma *cor* (e.g., um inteiro positivo) a cada vértice, de modo que a dois vértices adjacentes nunca seja atribuída a mesma cor e que o número total de cores seja o menor possível. O número de cores $\chi(G)$ que resolve esse problema de otimização é o *número cromático* de G. Uma coloração de G com $\chi(G)$ cores é chamada de *ótima*. Além de ser NP-difícil, esse problema também é de difícil aproximação com razão de no máximo $n^{\frac{1}{5}-\epsilon}$, para qualquer $\epsilon > 0$, desde que NP $\neq$ coRP [Bellare et al. 1998].

2. Proposta

Nossa proposta tem como fundamento a premissa a seguir.

Premissa. *Dado um grafo G, existem um conjunto E' de arestas e um algoritmo $\mathcal{A}$ que recebe G como entrada e encontra $\chi(G)$ ou que fornece um limite superior para $\chi(G)$, tais que: (i) adicionando E' a G, o grafo G' resultante é tal que $\mathcal{A}(G') = \chi(G') = \chi(G)$;*

(ii) se $\mathcal{A}$ resolve o problema de forma exata, o tempo computacional para determinar $\mathcal{A}(G')$ é muito menor que aquele de $\mathcal{A}(G)$.

A aplicação dessa premissa é direta: para obter $\chi(G)$ utilizando $\mathcal{A}$, encontramos o conjunto E' e o adicionamos a G. A expectativa é poder transferir a dificuldade teórica do problema de coloração para a dificuldade prática de encontrar E'. A seguir, primeiro discutimos a validade da premissa e, em seguida, como procurar pelo conjunto E'.

Testamos a validade da premissa empiricamente. Como instâncias do algoritmo $\mathcal{A}$, usamos um algoritmo para encontrar $\chi(G)$ [Trick 1994] e o clássico algoritmo guloso para encontrar um limite superior para $\chi(G)$ [Brélaz 1979]. Este último consiste em escolher uma ordem aleatória de $V(G)$ e, para cada vértice nessa ordem, escolher a menor cor diferente das cores já utilizadas em sua vizinhança. Quanto ao conjunto E', partimos do pressuposto de ser conhecida uma coloração ótima $f\colon V(G) \to \{1, 2, \ldots, \chi(G)\}$ e definimos $E' = E^f$ como o conjunto de não-arestas ij de G (i.e., $ij \notin E(G)$) tais que $f(i) \neq f(j)$, resultando em $G' = G^f = (V(G), E(G) \cup E')$. Note que $\chi(G) = \chi(G^f)$.

Como instâncias de G, utilizamos os chamados grafos rainha. Um grafo *rainha* R_n modela os movimentos de uma rainha em um tabuleiro de xadrez $n \times n$. Cada casa do tabuleiro é representada por um vértice, e cada possível movimento de uma rainha entre duas casas em uma única jogada (i.e., entre casas em uma mesma linha, coluna ou diagonal) é representado por uma aresta [Campbell 1977]. Essa classe de grafos é de solução exata particularmente difícil. Sabe-se que $\chi(R_n) = n$ para todo n que não seja múltiplo de 2 nem de 3. Entretanto, em geral não é verdade que $\chi(R_n) = n$; por exemplo, $\chi(R_9) = 10$. O menor n para o qual não se conhece $\chi(R_n)$ é $n = 27$ [Vasquez e Vimont 2018].

Tabela 1. Teste da premissa com o algoritmo exato [Trick 1994] e grafos rainha.

Grafo $G = (V, E)$	$\chi(G)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert E'\rvert$	Grafo G	Grafo G^f
R_8	9	64	728	1 090	1,45 s	0,04 s
R_9	10	81	1 056	1 893	21,45 min	0,1 s
R_{12}	12	144	1 804	7 700	$>$ 24 h	0,1 s
R_{14}	14	196	2 912	14 924	$>$ 24 h	0,1 s
R_{15}	15	225	3 605	20 020	$>$ 24 h	0,1 s
R_{16}	16	256	4 400	26 320	$>$ 24 h	0,1 s
R_{24}	24	576	15 272	143 704	$>$ 24 h	0,1 s

Tabela 2. Teste da premissa com o algoritmo guloso e grafos rainha.

				Grafo G		Grafo G^f	
Grafo $G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert E'\rvert$	Mín.	Máx.	Mín.	Máx.
R_8	64	728	1 090	10	16	9	9
R_9	81	1 056	1 893	11	17	10	10
R_{12}	144	1 804	7 700	13	19	12	12
R_{14}	196	2 912	14 924	15	22	14	14
R_{15}	225	3 605	20 020	16	23	15	15
R_{16}	256	4 400	26 320	18	24	16	16
R_{24}	576	15 272	143 704	26	35	24	24

Mostramos na Tabela 1 os tempos resultantes do teste da premissa com o algoritmo exato. De fato, eles são muito menores após a adição de E^f a G. A Tabela 2 refere-se aos testes com o algoritmo guloso. Agora são tabulados os intervalos onde foram encontrados os limites superiores para $\chi(G^f)$ em um milhão de tentativas. Pela Tabela 1, de fato foram encontrados os valores corretos para $\chi(G^f)$.

3. Adição de arestas guiada por Busca Monte Carlo em Árvore

A construção de G^f toma como conhecida uma coloração ótima de G. No caso geral, no qual $\chi(G)$ é desconhecido, determinamos o conjunto E' através da Busca Monte Carlo em Árvore (BMCA) [Coulom 2007].

A BMCA é um algoritmo de aprendizado por reforço, muito empregado na inteligência artificial de jogos (e.g., no programa AlphaGo [Silver et al. 2016]), no qual o resultado de um jogo é 0 ou 1 (derrota ou vitória). O algoritmo simula diversas partidas aleatórias, anotando as configurações de jogo mais promissoras, bem como as que devem ser evitadas, com base puramente em análise estatística das simulações. Cada nó i da árvore representa uma configuração de jogo, chamada de *estado*, com dois valores associados: $valor(i)$, dado pela média dos resultados obtidos por simulações a partir de i, e $visitas(i)$, dado por quantas foram essas simulações. Nem todas as configurações de jogo são representadas por um estado. Aquelas escolhidas para serem representadas, e os estados a partir dos quais são executadas as simulações, são definidos por quatro etapas que se repetem: *seleção*, *expansão*, *simulação* e *retro-propagação* (Figura 1).

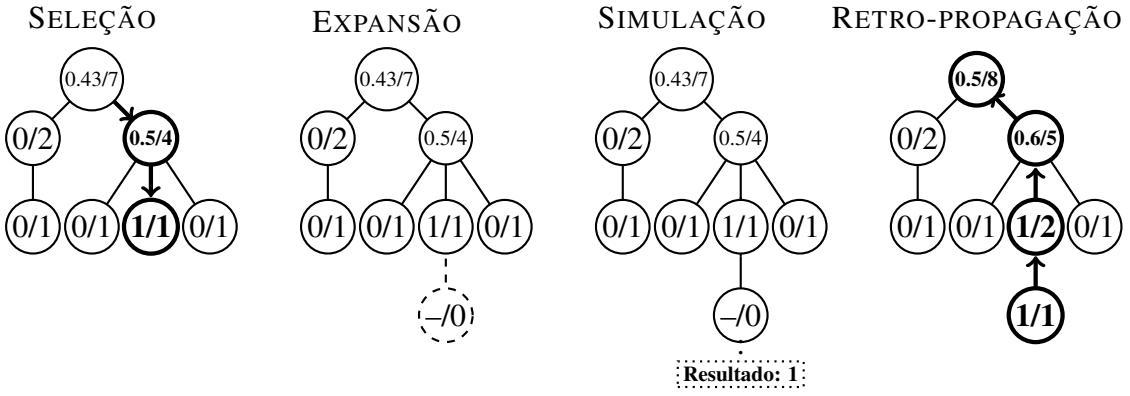

Figura 1. As quatro etapas da BMCA. O rótulo do nó i é $valor(i)$ / $visitas(i)$.

Um nó é chamado de *explorado* se todas as configurações de jogo possíveis de serem derivadas a partir daquela associada a ele já estão representadas por estados. A etapa de *seleção* escolhe a próxima configuração de jogo a ser representada por um estado. Para isso, a árvore é percorrida em profundidade a partir do nó raiz. Para cada nó explorado encontrado, seleciona-se o filho i desse nó que tenha o maior *limite superior de confiança*, denotado por $LSC(i)$ e dado por

$$LSC(i) = valor(i) + C\sqrt{\frac{\ln visitas(pai(i))}{visitas(i)}},$$

onde $pai(i)$ denota o nó explorado em questão. Há uma interpretação interessante para essa expressão: nós que possuam uma valoração maior tendem a ser mais escolhidos,

fenômeno chamado de *aproveitamento*, assim como nós com baixo número de visitas, fenômeno chamado de *exploração*. Isso indica que a BMCA mantém um equilíbrio entre aproveitar os melhores resultados e explorar novas possibilidades. A constante C é arbitrária, sendo usada para calibrar o peso da parcela de exploração. Neste trabalho, $C = 1$. Se a seleção atinge um nó não explorado, inicia-se a etapa de *expansão*. Nessa etapa, um novo nó é criado como filho do nó atual para representar uma configuração de jogo derivável da sua, iniciando-se assim a próxima etapa. Na etapa de *simulação*, jogadas aleatórias são realizadas a partir do nó expandido até que haja um resultado do jogo. Na etapa de *retro-propagação*, atualizam-se as médias dos valores e os contadores de visita de todos os nós percorridos na árvore, desde o nó folha expandido até o nó raiz.

Para o problema de coloração, o seguinte "jogo" é modelado: cada estado está associado a um grafo, sendo G o grafo associado ao nó raiz. Para uma ordenação $e_1, \ldots, e_{|\overline{E}|}$ das não-arestas de G, as duas possíveis "jogadas" a partir de um estado (grafo) do nível $\ell \in \{1, \ldots, |\overline{E}|\}$ consistem em escolher se o par de vértices e_ℓ será ou não unido por uma aresta no grafo atual. Nessa modelagem, a árvore é binária de altura máxima $|\overline{E}| + 1$. O "final de jogo" é dado pela escolha final do conjunto E', para o qual a simulação no caso de $G = R_n$ tem resultado

$$1 - \frac{\min\{c, \Delta(G) + 1\} - n}{\Delta(G) + 1 - n},$$

onde $c \geq \chi(G')$ é o número de cores retornado pelo algoritmo DSATUR [Brélaz 1979] para o grafo G'. Para esse algoritmo e $G = R_n$, sabemos que $n \leq c \leq \Delta(G') + 1$.

A Tabela 3 apresenta os resultados de um experimento onde a BMCA foi executada para cada instância de G até encontrar um número de cores para G', denotado por $bmca(G')$, tal que $bmca(G') = \chi(G)$, ou até atingir o tempo limite de 3 600 segundos com $bmca(G') > \chi(G)$. A tabela revela que algumas colorações obtidas chegaram ao resultado ótimo enquanto instâncias maiores têm resultado aproximado em tempos bastante reduzidos se comparados aos da Tabela 1 para o grafo G.

Tabela 3. Experimento com a BMCA valorando estados com o DSATUR.

Grafo G	Tempo	$bmca(G')$	$bmca(G') - \chi(G)$	Tempo até encontrar $bmca(G')$
R_8	34,19 s	9	0	34,19 s
R_9	280,92 s	10	0	280,92 s
R_{10}	3 600 s	12	1	0,0 s
R_{12}	3 600 s	14	2	31,0 s
R_{14}	3 600 s	17	3	3,0 s
R_{15}	3 600 s	18	3	1,0 s
R_{16}	3 600 s	19	3	77,0 s
R_{24}	3 600 s	28	4	369,0 s

4. Conclusão

O uso da BMCA para encontrar supergrafos de G para os quais um algoritmo exato determine $\chi(G)$ em tempo reduzido é uma novidade deste trabalho. Pelos resultados preliminares apresentados, vê-se que pode ser um caminho promissor para a obtenção de números cromáticos de grafos que ainda resistam à determinação por outros métodos.

Referências

Bellare, M., Goldreich, O., e Sudan, M. (1998). Free bits, PCPs, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27(3):804–915.

Brélaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256.

Campbell, P. J. (1977). Gauss and the eight queens problem: A study in miniature of the propagation of historical error. *Historia Mathematica*, 4(4):397–404.

Coulom, R. (2007). Efficient selectivity and backup operators in Monte-Carlo tree search. Em van den Herik, H. J., Ciancarini, P., e Donkers, H. H. L. M., editores, *Computers and Games 2006*, volume 4630 de *Lecture Notes in Computer Science*, páginas 72–83. Springer-Verlag, Berlin.

Karp, R. M. (1972). Reducibility among combinatorial problems. Em Miller, R. E. e Thatcher, J. W., editores, *Complexity of Computer Computations*, páginas 85–103. Plenum Press, New York. Também disponível em `https://doi.org/10.1007/978-3-540-68279-0_8`. Último acesso: 07/03/2024.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., e Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489.

Trick, M. (1994). Network Resources for Coloring a Graph. `https://mat.tepper.cmu.edu/COLOR/color.html`. Último acesso: 22/02/2024.

Vasquez, M. e Vimont, Y. (2018). On solving the queen graph coloring problem. Em Brankovic, L., Ryan, J., e Smyth, W., editores, *Combinatorial Algorithms 2017*, volume 10765 de *Lecture Notes in Computer Science*, páginas 244–251. Springer, Cham.