Proposta e Avaliação de Novas Heurísticas Para o Problema de Coloração de Vértices

Thalyson Gomes Nepomuceno da Silva, Leonardo Sampaio Rocha, Gerardo Valdisio Rodrigues Viana
Universidade Estadual do Ceará
Av. Dr. Silas Munguba, 1700, Campus do Itaperi, Fortaleza, CE - Brasil - 60714-903
thalyson.uece@gmail.com, leonardo.sampaio@uece.com,
valdisio.viana@uece.br

RESUMO

O problema de Coloração de Vértices consiste em colorir os vértices de um grafo de tal forma que vértices adjacentes possuam cores distintas e que o número de cores utilizadas seja o menor possível. É um problema extensivamente estudado e com diversas aplicações práticas, especialmente em problemas de escalonamento. Sabe-se que, a menos que P = NP, não existem algoritmos polinomiais exatos ou aproximativos a um fator constante para o problema, motivando o uso de heurísticas para instâncias de aplicações. Neste trabalho são apresentadas novas heurísticas para o problema, baseadas em uma técnica para divisão de grafo e no uso de uma nova estratégia, inspirada no conceito de b-coloração, para reduzir o número de cores utilizadas. Nos testes computacionais, as heurísticas se mostraram competitivas em relação ao tempo de processamento e qualidade de solução, conseguindo soluções tão boas quanto a de algoritmos existentes e com um tempo de processamento significantemente inferior.

PALAVRAS CHAVE. Coloração de grafos. Heurísticas. Coloração gulosa. b-coloração.

TAG (Teoria e Algoritmos em Grafos)

ABSTRACT

The vertex coloring problem consists in coloring the vertices of a graph in such a way that adjacent vertices have different colors and the number of colors used is minimum. It is an extensively studied problem with diverse practical applications, specially in scheduling problems. It is known that, unless P = NP, there are no polynomial exact or constant-factor approximation algorithms to the problem, what motivates the use of heuristics to applications of the problem. This work presents new heuristics for the problem, based on a graph division technique and a new strategy inspired on the concept of b-coloring, to reduce the number of colors. In computational tests, the heuristics proved to be competitive with respect to the processing time and solution quality, obtaining solutions as good as the ones from existent algorithms and running in a significantly reduced time.

KEYWORDS. Graph Coloring. Heuristics. Greedy coloring. b-coloring.

TAG (Theory and Graphs Algorithms)

1. Introdução

Dado um grafo $G(V, E)$, uma *coloração própria* c, ou simplesmente *coloração*, é uma partição $S = \{s_1, s_2, \ldots, s_k\}$ em que s_i é um conjunto independente, para $1 \leq i \leq k$, denominado *classe de cor*. De forma equivalente, é uma atribuição $c : V \rightarrow \{1, \ldots, k\}$ de modo que quaisquer dois vértices adjacentes em G não compartilhem uma mesma cor. Os valores na imagem são denominados *cores*. Sendo k cores são utilizadas, c é denominada uma k-coloração. O *número cromático* $\chi(G)$ é o menor inteiro k tal que G possui uma k-coloração. O problema de coloração de grafos consiste em determinar o número cromático de G.

Em geral, problemas em que objetos precisam ser agrupados de modo que objetos conflitantes não estejam no mesmo grupo podem ser modelados utilizando coloração de grafos. Na modelagem, os vértices representam objetos, e as arestas representam os conflitos existentes entre os objetos. A coloração é um particionamento dos objetos em conjuntos, de tal forma que objetos conflitantes não pertençam ao mesmo conjunto. Diversos problemas práticos podem ser modelados como um problema de coloração em grafo:

- Problema de grade de horários de professores [de Werra 1985];
- Alocação de tempo e frequência em sistemas de comunicação [Gamst 1986];
- Alocação de registradores em um processador [Chaitin et al. 1981];
- Controle de fluxo aéreo [Barnier e Brisset 2004].

O problema de coloração faz parte da lista dos 21 problemas NP-Completos apresentados por Karp em 72 [Karp 1972]. Além disso, Lund e Yannakakis [Lund e Yannakakis 1994] e, posteriormente, Zuckerman [Zuckerman 2007], investigaram a existência de algoritmos aproximativos para o problema, mostrando em particular que o problema não admite algoritmo aproximativo de fator constante, a menos que P=NP. Algoritmos exatos, de tempo exponencial, foram propostos para solucionar o problema [Lucet et al. 2006, Bodlaender e Kratsch 2006, Koivisto 2006], sendo viáveis apenas em instâncias de tamanho reduzido. Tais fatos justificam o uso de métodos heurísticos para o problema. As primeiras abordagens heurísticas utilizadas foram heurísticas gulosas, que apresentam um bom desempenho para grande variedade de grafos, a exemplo de DSATUR [Brélaz 1979] e RLF [Leighton 1979]. Apesar das heurísticas gulosas serem consideradas muito eficientes para grafos gerais, podem encontrar soluções de baixa qualidade em algumas famílias de grafos [Ruiz 2012]. Várias heurísticas de busca local baseadas em meta-heurísticas vêm sendo propostas [Douiri e Elbernoussi 2015, Abbasian et al. 2011, Aguiar et al. 2005].

O presente trabalho propõe novas heurísticas aplicáveis ao problema de coloração de grafos. As heurísticas são baseadas em uma técnica de divisão do grafo, bem como da aplicação de uma estratégia inspirada em b-colorações, propostas por Irving e Manlove [Irving e Manlove 1999] a partir de uma estratégia para redução do número de cores de uma dada coloração de um grafo. Em uma b-coloração, cada classe de cor utilizada na coloração contém pelo menos um b-vértice, ou seja, um vértice que apresenta todas as demais cores utilizadas na sua vizinhança. Para validação das heurísticas propostas, elas foram implementadas e confrontadas com os algoritmos propostos em [Douiri e Elbernoussi 2015, Abbasian et al. 2011, Brélaz 1979]. Nos testes computacionais, os algoritmos propostos se mostraram competitivos em relação ao tempo de processamento e qualidade de solução. Para as instâncias utilizadas, os algoritmos encontram, em média, soluções que utilizam $1,58$ à $1,83$ cores a menos que as soluções encontradas pelo algoritmo DSATUR. Além disso, para a maioria das instâncias, os algoritmos encontram soluções tão boas quanto as soluções encontradas pelos algoritmos propostos em [Douiri e Elbernoussi 2015, Abbasian et al. 2011].

O trabalho está dividido em sete seções. A Seção 2 apresenta alguns métodos algorítmicos utilizados para obter soluções para o problema de coloração de grafos. A Seção 3 apresenta conceitos de b-coloração e o funcionamento da b-estratégia. Na Seção 4, são apresentados os resultados

do uso da b-estratégia para reduzir o número de cores de soluções conseguidas com a utilização do algoritmo DSATUR. A Seção 5 apresenta as heurísticas propostas e as técnicas utilizadas em cada uma delas. A Seção 6 apresenta os resultados computacionais obtidos com o uso das heurísticas propostas. A Seção 7 apresenta as conclusões do presente trabalho.

2. Algoritmos para Coloração de Grafos

Algumas das principais abordagens algorítmicas, utilizadas para o problema de coloração são baseadas em métodos gulosos que realizam a coloração sequencial dos vértices usando uma função gulosa como guia. Um exemplo de tal abordagem é o *algoritmo guloso*, também conhecido como *algoritmo sequencial*, descrito no Algoritmo 1.

Algorithm 1 Algoritmo Sequencial (Guloso)

Entrada: Um grafo $G(V, E)$ e uma sequência S de vértices.
Saída: Número de cores utilizadas para colorir o grafo G.

for Cada elemento de $S =< v_1, v_2, \ldots, v_k >$ **do**
  Colore o vértice v_i com a menor cor que não pertença a nenhum vértice de $N(v_i)$;
end for
Retorna: A maior cor utilizada;

O problema com esse algoritmo é que ele pode apresentar um desempenho ruim dependendo da sequência dos vértices escolhida, utilizando mais cores que o número cromático do grafo. A Figura 1 mostra que o grafo P_4 pode ser colorido com duas ou três cores pelo algoritmo guloso. Os *cografos*, que são os grafos que não possuem P_4 como subgrafo induzido, são os únicos grafos para os quais o algoritmo guloso sempre encontra uma coloração ótima, independentemente da ordem tomada [Gyárfás e Lehel 1988].

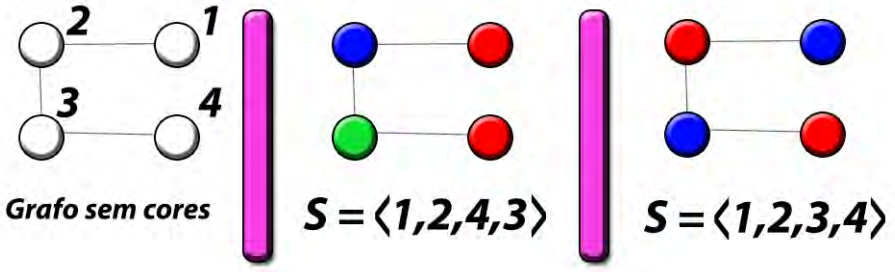

Figura 1: Colorações obtidas pelo algoritmo guloso em diferentes sequências de vértices

Vários heurísticas gulosas foram propostas na literatura. Em 1976, Welsh e Powell propuseram o algoritmo LF (Large First), que analisa os vértices em ordem decrescente de seus graus [Garey e Johnson 1976]. Em 1979, Leighton propôs o algoritmo RLF (Recursive Largest First), que é uma modificação do LF [Leighton 1979]. Ainda em 1979, Brélaz propôs o algoritmo DSATUR, que tem prioridade em colorir vértices que possuem maior quantidade de cores distintas na sua vizinhança [Brélaz 1979]. Embora as heurísticas gulosas sejam muito rápidas, estas podem gerar soluções insatisfatórias para algumas instâncias de grafos [Lü e Hao 2010]. Nas últimas décadas, os algoritmos gulosos têm sido amplamente empregados em aplicações em que o tempo de execução é priorizado e para gerar soluções iniciais em meta-heurísticas, tal como em [Laguna e Martí 2001] [Douiri e Elbernoussi 2015] [Abbasian et al. 2011] [Aguiar et al. 2005] [Ruiz 2012].

3. Uso de b-estratégia para reduzir o número de cores

Dado um grafo $G(V, E)$ e uma coloração c de seus vértices, um *b-vértice* é um vértice com vizinhos em todas as demais cores de c. Caso exista uma classe de cor que não possua b-vértices,

pode-se recolorir cada um de seus vértices, atribuindo uma das cores que não aparece entre seus vizinhos. Com isso uma cor é eliminada. Pode-se repetir este processo, denominado *b-estratégia*, até que todas as classes de cor possuam pelo menos um b-vértice. Uma tal coloração é denominada *b-coloração*, e não pode ser melhorada pela b-estratégia. O *número b-cromático* de um grafo $G(V, E)$, denotado por χ_b(G), é o maior inteiro k para o qual G possui uma b-coloração com k cores. Estes conceitos foram introduzidos em 1999 por Irving e Manlove, que provaram que determinar χ_b(G) é um problema NP-Difícil para grafos gerais e polinomial para árvores [Irving e Manlove 1999].

Considere o grafo da Figura 2(a). Caso seja aplicado o algoritmo sequencial, usando como sequência para coloração $S =< 1, 2, 3, 4, 5, 6, 7, 8 >$, o algoritmo obtém como resposta a coloração apresentada na Figura 2(b). Aplicando a b-estratégia na coloração obtida, pode-se eliminar a classe de cor 1, pois ela não apresenta b-vértices. Recolorindo-se os vértices desta classe de cor, consegue-se a configuração da Figura 2(c). Perceba que não é possível reduzir mais classes de cor utilizando a b-estratégia, pois todas as classes de cor utilizadas contém um b-vértice.

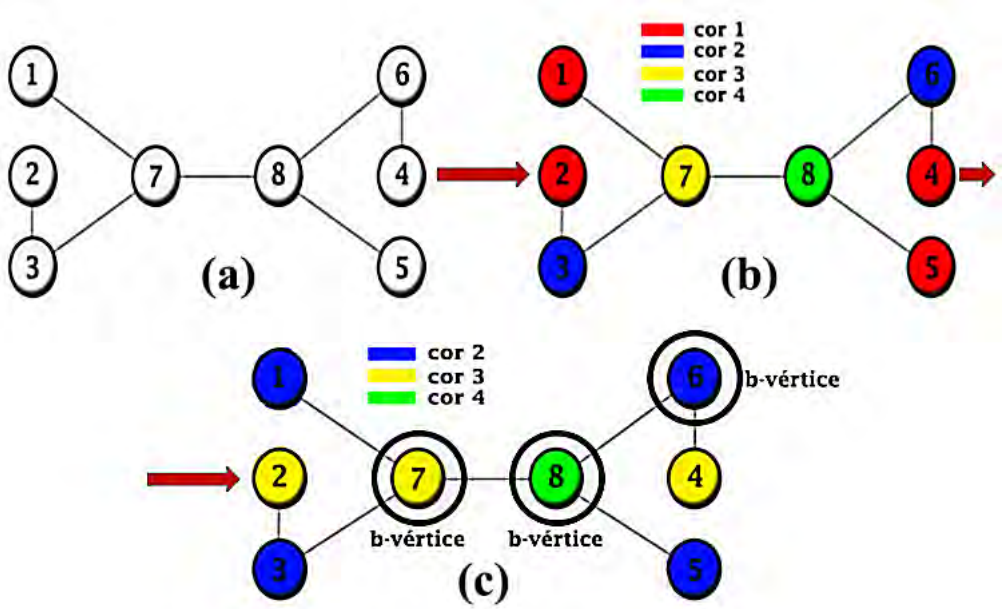

Figura 2: Exemplo de execução da b-estratégia

4. Testes Preliminares

Foi implementado o algoritmo DSATUR, tendo o mesmo sido testado em grafos da família QUEEN, fornecidos em DIMACS [Johnson e Trick 1996], no endereço `http://mat.gsia.cmu.edu/COLOR/instances.html`. Após a execução do DSATUR, também foi aplicada a b-estratégia para tentar reduzir o número de cores utilizadas nas soluções fornecidas pelo DSATUR, gerando os resultados apresentados na Tabela 1.

Ao aplicar a b-estratégia em colorações obtidas pelo DSATUR, observou-se uma pequena redução da quantidade de cores. Esse fato ocorre por conta do procedimento de construção de solução do DSATUR. O algoritmo prioriza a coloração de vértices que possuam maior grau de saturação. Ou seja, aqueles vértices que possuem maior número de cores em sua vizinhança. Isso gera uma maior quantidade de cores que possem b-vértices, reduzindo o número de cores que podem ser eliminadas até que se chegue a uma b-coloração.

	DSATUR
queen5_5	$5 \rightarrow 5$
queen6_6	$9 \rightarrow 9$
queen7_7	$11 \rightarrow 11$
queen8_8	$12 \rightarrow 11$
queen8_12	$14 \rightarrow 14$
queen9_9	$13 \rightarrow 12$
queen10_10	$14 \rightarrow 14$
queen11_11	$15 \rightarrow 15$
queen12_12	$16 \rightarrow 16$
queen13_13	$17 \rightarrow 17$
queen15_15	$21 \rightarrow 20$
queen16_16	$23 \rightarrow 22$
Média	$14{,}16 \rightarrow 13{,}83$

Tabela 1: A tabela apresenta os números de cores utilizadas pelo o algoritmo DSATUR para as instâncias QUEEN. Os valores são apresentados de forma $x \rightarrow y$, em que x representa a quantidade de cores obtidas pela heurística sem utilização da b-estratégia, e y representa a quantidade de cores utilizadas ao final da b-estratégia. Na última linha da tabela, são apresentadas as médias de cores utilizadas pelo algoritmo DSATUR nas instâncias.

5. Algoritmos Propostos

Tendo em vista as limitações observadas em heurísticas existentes, e particularmente a interação observada entre DSATUR e a b-estratégia, propomos novas heurísticas visando aprimorar o desempenho em problemas de coloração. Sobretudo aqueles em que o tempo é um recurso crítico.

A ideia base das heurísticas propostas consiste em dividir o grafo a ser colorido em subgrafos, e então colorir cada subgrafo a partir de um vértice central. A intuição que motiva esta estratégia é a de concentrar as cores maiores nos vértices das bordas dos subgrafos, tornando mais fácil a reatribuição de cores dos vértices internos por meio de uma modificação na b-estratégia. No que segue, descrevemos as 4 estratégias adotadas:

1. Coloração por Busca em Largura;
2. Divisão em raiz quadrada;
3. Prioridade de menor cor;
4. Modificação da b-estratégia.

5.1. Coloração por Busca em Largura

Para utilizar uma busca em largura na coloração de grafos, inicialmente é escolhido um vértice inicial que é adicionado em uma fila. Enquanto a fila não for vazia, o vértice do início da fila é removido, colorido com a menor cor disponível e adiciona na fila cada um dos seus vizinhos que ainda não foi adicionado na fila anteriormente. Tal algoritmo apresentado no Algoritmo 2.

Propomos a utilização de busca em largura modificada, com passos de tamanho 2, que chamamos de *busca 2-Step*. O funcionamento é semelhante à coloração por busca em largura. Porém, ao colorir um vértice, serão adicionados na fila apenas os vértices a uma distância 2 do vértice colorido. Perceba que a utilização dessa busca pode não colorir todo o grafo conexo, como ilustrado na Figura 3. Ao final da busca, os vértices não coloridos são coloridos pelo algoritmo sequencial.

Algorithm 2 Coloração por Busca em Largura

Entrada: Um grafo $G(V, E)$ e um vértice inicial v.

Inicializa uma fila F de vértices;
Adiciona v na fila;
while $|F| \neq 0$ **do**
  $v = remover(F)$;
  Colore v com a menor cor c que não é utilizada por vértices na vizinhança de v;
  for Cada vértice x na vizinhança de v **do**
    if x ainda não foi adicionado na fila F **then**
      Adiciona x na fila;
    end if
  end for
end while

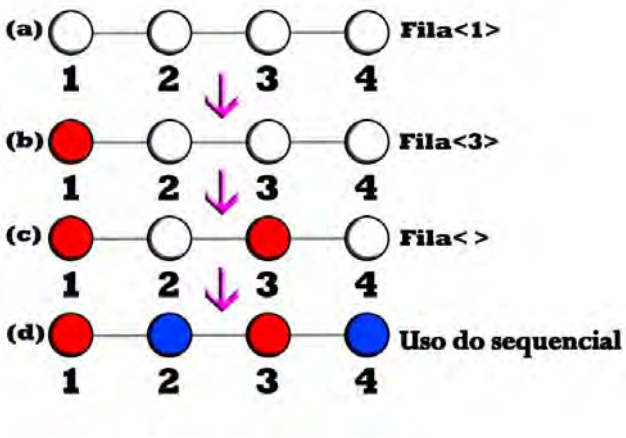

Figura 3: Execução de busca em largura com utilização de passos de tamanho dois. Nessa figura é ilustrado um grafo de 4 vértices, inicialmente, não coloridos. Em (a), o vértice 1 é adicionado na fila para coloração. Em (b), o vértice 1 é colorido e o vértice 3 é adicionado na fila para coloração. Em (c), o vértice 3 é colorido, e nenhum vértice é adicionado na fila, pois todos os vértices com distância 2 do vértice 3 já foram coloridos. Em (d), os vértices 2 e 4 são coloridos.

A utilização do 2-Step tem como objetivo maximizar o número de vértices com cores menores, pois em uma aplicação da b-estratégia em uma coloração conseguida a partir do algoritmo sequencial, só é possível recolorir um vértice com uma cor que seja maior que a sua. Isto decorre do fato de o algoritmo sequencial utilizar a cor k para um vértice apenas quando ele possuir vizinhos com as cores em $\{1, 2, \ldots, k-1\}$.

5.2. Prioridade de menor cor

Outra estratégia proposta é a utilização de prioridade para colorir os vértices com as menores cores primeiro. Para colorir um grafo $G(V, E)$ utilizando essa estratégia, um vértice inicial v é escolhido de forma arbitrária e colorido com a cor 1. Em seguida, enquanto existirem vértices não coloridos, um vértice u é selecionado e colorido com a menor cor não apresentada na sua vizinhança. Sendo u um vértice não colorido que possui pelo menos um vizinho colorido e que pode ser colorido com a menor cor possível.

Nas implementações, foi utilizada uma árvore binária balanceada para armazenar as menores cores possíveis para colorir os vértices não coloridos do grafo. Sendo necessário atualizar a árvore a cada vértice colorido.

Assim como a 2-Step, essa estratégia também tem como objetivo maximizar o número de vértices com classes de cor de menores identificadores.

5.3. Divisão em Raiz Quadrada

Na divisão proposta, vamos dividir o grafo $G(V, E)$ em $\sqrt{|V|}$ subgrafos, em seguida, colori-los utilizando busca em largura a partir dos vértices iniciais de cada subgrafo. A inspiração para a divisão do grafo antes da coloração vem do algoritmo de ordenação por Bucket, que divide um vetor em buckets para então ordena-lo. O algoritmo utilizado para dividir um grafo $G(V, E)$ em $\sqrt{|V|}$ subgrafos é apresentado no Algoritmo 3. Para melhor entendimento do procedimento, a Figura 4 ilustra o passo a passo da execução do algoritmo para um grafo de 9 vértices.

Algorithm 3 Divisão em Raiz Quadrada

Entrada: Um grafo $G(V, E)$.
Saída: Os subgrafos resultantes da divisão.
inteiro $raiz \leftarrow \sqrt{|V|}$;
inteiro $x \leftarrow 1$;
Inicializa uma fila F de pares (v, g);
while $x \leq |V|$ **do**
  Acrescenta o par (x, x) na fila F;
  Inicializa um subgrafo com o vértice x;
  $x \leftarrow x + raiz$
end while
while fila F não vazia **do**
  $(v, g) \leftarrow$ remove(F);
  for Cada vértice u vizinho do vértice v que ainda não foi adicionado na fila F **do**
    Acrescenta o par (u, g) na fila F;
    Acrescenta vértice u ao subgrafo iniciado em g;
  end for
end while
Retorna: Os subgrafos gerados;

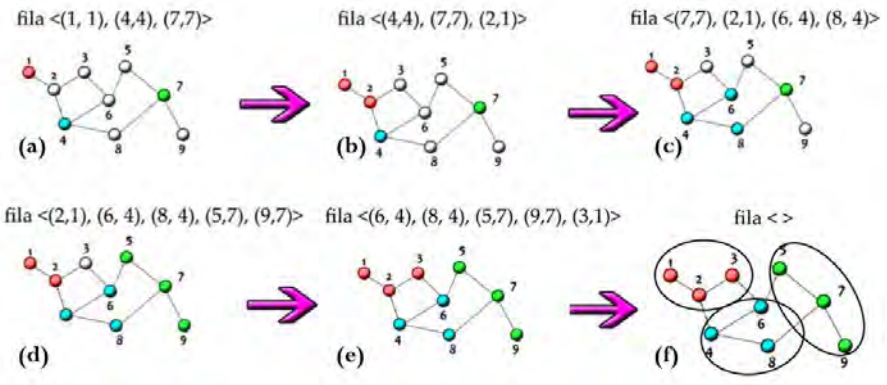

Figura 4: Execução da divisão em raiz quadrada em um grafo de 9 vértices. Na Figura (a), os vértices 1, 4 e 7 são selecionados como iniciais e adicionados na fila. Na Figura (b), o vértice 1 é removido da fila e o vértice 2 é adicionado na fila e no subgrafo iniciado em 1. Na Figura (c), o vértice 4 é removido da fila e os vértices 6 e 8 são adicionados na fila e no subgrafo iniciado em 4. Na Figura (d), o vértice 7 é removido da fila e os vértices 5 e 9 são adicionados na fila e no subgrafo iniciado em 7. Na Figura (e), o vértice 2 é removido da fila e o vértice 3 é adicionado na fila e no subgrafo iniciado em 1. Na Figura (f), não é possível adicionar outro vértice na fila, assim, todos os vértices são removidos da fila e 3 subgrafos são retornados.

Essa divisão tem como objetivo posicionar os vértices coloridos com índices de cores maiores nas bordas dos subgrafos, facilitando as alterações das cores dos vértices internos quando a b-estratégia for executada.

5.4. Modificação da b-estratégia

Ao se considerar o grafo da Figura 5(a) aplicado ao algoritmo sequencial, obtem-se a coloração ilustrada na Figura 5(b). Aplicando a b-estratégia a esta coloração, apenas a cor 1 é eliminada, restando ao final uma b-coloração com 3 cores.

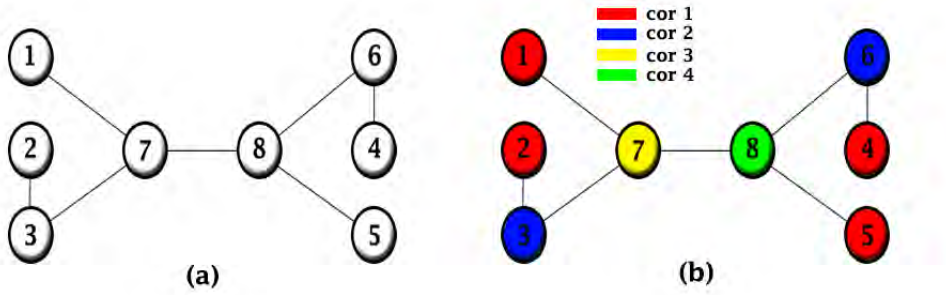

Figura 5: (a) Representa um grafo de 8 vértices, sendo o número escrito em cada vértice correspondente ao respectivo índice. (b) Coloração final do grafo (a), utilizando o algoritmo sequencial.

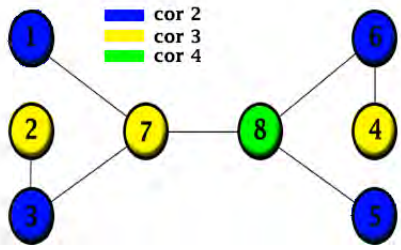

Figura 6: Configuração final aplicando b-estratégia.

A modificação que propomos é a de permitir movimentos parciais de vértices de uma classe de cor para outra. Ou seja, não é exigido que a classe de cor não possua b-vértices para que se tente recolorir os seus vértices. Cada classe de cor, a partir da primeira, é considerada, recolorindo cada um de seus vértices com a menor cor diferente da própria, sempre que isto for possível. Do contrário, o vértice permanece com a cor que já possui.

Pode-se aplicar a proposta à coloração da Figura 6. Quando a classe de cor 2 é considerada, o vértice 1 recebe cor 4, o vértice 3 recebe cor 4, o vértice 5 recebe a cor 3 e a cor do vértice 6 não é alterada, pois é um b-vértice. Isto resulta na coloração da Figura 7(a). Agora analisando a classe de cor 3, o vértice 2 recebe a cor 2, o vértice 4 recebe a cor 4, o vértice 5 recebe a cor 2 e o vértice 7 recebe a cor 2. Resultando na coloração da Figura 7(b). Por fim, a classe de cor 4 é analisada. Porém nenhum vértice pode ser recolorido. A coloração coloração resultante utiliza apenas 2 cores, uma cor a menos que o obtido pela aplicação da b-estratégia padrão.

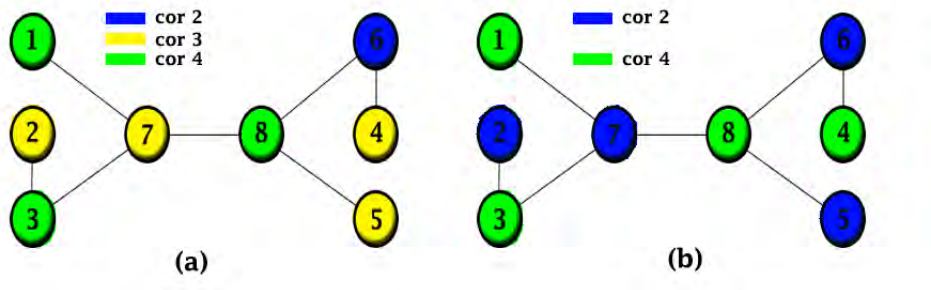

Figura 7: (a) Apresenta a configuração da coloração do grafo após a classe de cor 2 ser analisada. (b) Apresenta a configuração da coloração do grafo após a classe de cor 3 ser analisada.

5.5. Utilização das Estratégias

Com a utilização das estratégias explicadas anteriormente, elaboramos 4 heurísticas.

1. **Sqrt Start 1 (SS1)**: É utilizada a divisão por raiz quadrada, em seguida os vértices são coloridos utilizando uma busca em largura, partido do vértice central de cada subgrafo;

2. **Sqrt Start 2 (SS2)**: Semelhante à SS1, porém é utilizada uma busca com passos de tamanho dois, 2-Step;

3. **Sqrt Start Priority 1 (SSP1)**: Semelhante à SS1, porém é utilizada a prioridade de menor cor;

4. **Sqrt Start Priority 2 (SSP2)**: Semelhante à SSP1, porém é utilizada uma busca com passos de tamanho dois, 2-Step;

Em todas as heurísticas propostas, a b-estratégia modificada é executada ao final dos algoritmos.

6. Testes Computacionais

Foram implementadas e testadas as 4 heurísticas propostas nesse trabalho. Foram utilizadas as instâncias obtidas no repositório `http://mat.gsia.cmu.edu/COLOR/instances.html`. Todos os testes realizados foram executados em um computador com processador Intel Core i3-3220 3.30GHz, utilizando como sistema operacional o Windows 7 Ultimate.

6.1. Comparação com meta-heurísticas

As propostas foram confrontadas com os resultados obtidos pelos algoritmos propostos em [Douiri e Elbernoussi 2015] e [Abbasian et al. 2011], que são propostas que utilizam meta-heurísticas. Em [Douiri e Elbernoussi 2015] é utilizada a meta-heurística Algoritmo Genético, e em [Abbasian et al. 2011] é utilizada a meta-heurística Algoritmo Cultural.

Na Tabela 2 são apresentados os resultados obtidos para cada instância. As instâncias escolhidas foram as utilizadas pelos trabalhos utilizados para comparação. Pode-se notar que as heurísticas propostas conseguem obter semelhantes aos conseguidos pela utilização de meta-heurísticas. Apenas na instância queen7_7, todas as heurísticas propostas perdem para o algoritmo proposto em [Douiri e Elbernoussi 2015].

	SS1	SS2	SSP1	SSP2	[Douiri e Elbernoussi 2015]	[Abbasian et al. 2011]
queen5_5	**5**	**5**	**5**	**5**	**5**	**5**
queen6_6	**7**	8	8	8	**7**	**7**
queen7_7	8	8	9	8	**7**	8
anna	**11**	**11**	**11**	**11**	**11**	**11**
david	**11**	**11**	**11**	**11**	**11**	**11**
games120	**9**	**9**	**9**	**9**	**9**	**9**
miles250	9	**8**	**8**	**8**	**8**	**8**

Tabela 2: A tabela apresenta a quantidade de cores utilizadas nas colorações.

6.2. Comparação com DSATUR e Impacto da *b*-estratégia

Para estudar o impacto do procedimento de b-coloração no resultado final obtido pelos algoritmos propostos, utilizamos as instâncias da família Queen. Escolhemos as instâncias dessa família porque, apesar de serem instâncias conhecidas pela sua dificuldade, os algoritmos propostos conseguiram obter soluções comparáveis às soluções obtidas por meta-heurísticas avançadas. Também iremos utilizar a b-estratégia modificada no algoritmo DSATUR, aplicando o procedimento ao final da execução do algoritmos.

Nos experimentos foram verificadas as quantidades de cores utilizadas nas soluções encontradas por cada heurística sem a utilização da b-estratégia, e em seguida foi aplicado o procedimento em cada solução, também registrando as quantidades de cores utilizadas nas soluções finais obtidas. Os resultados obtidos nesse experimento foram compilados na Tabela 3.

	DSATUR	SS1	SS2	SSP1	SSP2
queen5_5	$5 \rightarrow 5$	$7 \rightarrow 5$	$7 \rightarrow 5$	$6 \rightarrow 5$	$6 \rightarrow 5$
queen6_6	$9 \rightarrow 9$	$9 \rightarrow 7$	$9 \rightarrow 8$	$10 \rightarrow 8$	$9 \rightarrow 8$
queen7_7	$11 \rightarrow 10$	$11 \rightarrow 8$	$12 \rightarrow 8$	$10 \rightarrow 9$	$10 \rightarrow 8$
queen8_8	$12 \rightarrow 11$	$12 \rightarrow 10$	$13 \rightarrow 10$	$13 \rightarrow 10$	$12 \rightarrow 9$
queen8_12	$14 \rightarrow 14$	$17 \rightarrow 13$	$16 \rightarrow 12$	$15 \rightarrow 13$	$16 \rightarrow 13$
queen9_9	$13 \rightarrow 12$	$13 \rightarrow 11$	$15 \rightarrow 11$	$14 \rightarrow 11$	$14 \rightarrow 11$
queen10_10	$14 \rightarrow 14$	$16 \rightarrow 13$	$17 \rightarrow 12$	$15 \rightarrow 13$	$15 \rightarrow 12$
queen11_11	$15 \rightarrow 15$	$18 \rightarrow 14$	$16 \rightarrow 14$	$17 \rightarrow 13$	$18 \rightarrow 14$
queen12_12	$16 \rightarrow 16$	$18 \rightarrow 15$	$19 \rightarrow 15$	$19 \rightarrow 15$	$18 \rightarrow 14$
queen13_13	$17 \rightarrow 16$	$21 \rightarrow 16$	$20 \rightarrow 16$	$20 \rightarrow 16$	$20 \rightarrow 15$
queen15_15	$21 \rightarrow 20$	$21 \rightarrow 19$	$22 \rightarrow 19$	$23 \rightarrow 18$	$21 \rightarrow 19$
queen16_16	$23 \rightarrow 22$	$26 \rightarrow 20$	$24 \rightarrow 20$	$26 \rightarrow 20$	$24 \rightarrow 20$
Média	$14{,}16 \rightarrow 13{,}66$	$15{,}75 \rightarrow 12{,}58$	$15 \rightarrow 12{,}5$	$15{,}6 \rightarrow 12{,}58$	$15{,}25 \rightarrow 12{,}33$

Tabela 3: A tabela apresenta o número de cores utilizadas por cada heurística em cada instância. Os valores são apresentados de forma $x \rightarrow y$, em que x representa a quantidade de cores obtidas pela heurística sem utilização da b-estratégia e y representa a quantidade de cores utilizadas ao final da b-estratégia. Na última linha da tabela são apresentadas as médias de cores utilizadas por cada uma das heurísticas para o grupo de instâncias utilizado.

Analisando os resultados obtidos, podemos observar que sem a utilização da b-estratégia modificada, o resultado médio do DSATUR é melhor que o das heurísticas propostas. Porém, com a utilização da b-estratégia modificada ao final, as heurísticas propostas conseguem resultados melhores que os do DSATUR. A b-estratégia modificada de fato consegue uma maior redução no número de cores das colorações obtidas pelas heurísticas propostas.

7. Conclusões

Neste trabalho um novo procedimento de inicialização de coloração, em que uma busca em largura é realizada simultaneamente a partir de $\sqrt{n}$ vértices iniciais, visitando vértices a distância 1 , no caso 1-Step, ou à distância 2, no caso 2-Step. Uma modificação da b-estratégia também foi proposta, sendo capaz de reduzir o número de cores em instâncias em que a b-estratégia tradicional não seria capaz. Diferentes utilizações destas estratégias deram origem às 4 heurísticas propostas (SS1, SS2, SSP1 e SSP2) para o problema de coloração do grafos. Os resultados computacionais ilustrados na Seção 6 mostram que as heurísticas propostas são competitivas com relação as de outros trabalhos.

Como trabalho futuro, deve-se aprimorar a escolha dos vértices iniciais na divisão em subgrafos, de modo a otimizar a performance da b-estratégia modificada, que é aplicada posteriormente. Deve-se considerar também as aplicações práticas em que há a necessidade de obter colorações em tempo muito reduzido, investigando a performance das heurísticas nestas aplicações, identificando os obstáculos e aperfeiçoando as heurísticas de modo adequado.

Agradecimentos

Agradecemos à Fundação Cearense de Amparo à Pesquisa (FUNCAP) e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela apoio financeiro ao desenvolvimento da pesquisa.

Referências

Abbasian, R., Mouhoub, M. e Jula, A. (2011). Solving graph coloring problems using cultural algorithms. In *FLAIRS Conference*.

Aguiar, F. N., Honorato, G. d. S. C., Santos, H. G. e Ochi, L. S. (2005). Metaheurística busca tabu para o problema de coloração de grafos. *Anais do XXXVII SBPO*, p. 2497–2504.

Barnier, N. e Brisset, P. (2004). Graph coloring for air traffic flow management. *Annals of operations research*, 130(1-4):163–178.

Bodlaender, H. L. e Kratsch, D. (2006). An exact algorithm for graph coloring with polynomial memory. Technical report, Technical report, Department of Information and Computing Sciences, Utrecht University.

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

Chaitin, G. J., Auslander, M. A., Chandra, A. K., Cocke, J., Hopkins, M. E. e Markstein, P. W. (1981). Register allocation via coloring. *Computer languages*, 6(1):47–57.

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

Douiri, S. M. e Elbernoussi, S. (2015). Solving the graph coloring problem via hybrid genetic algorithms. *Journal of King Saud University-Engineering Sciences*, 27(1):114–118.

Gamst, A. (1986). Some lower bounds for a class of frequency assignment problems. *Vehicular Technology, IEEE Transactions on*, 35(1):8–14.

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

Gyárfás, A. e Lehel, J. (1988). On-line and first fit colorings of graphs. *Journal of Graph Theory*, 12(2):217–227.

Irving, R. W. e Manlove, D. F. (1999). The b-chromatic number of a graph. *Discrete Applied Mathematics*, 91(1):127–141.

Johnson, D. S. e Trick, M. A. (1996). *Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993*, volume 26. American Mathematical Soc.

Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E. e Thatcher, J. W., editores, *Complexity of Computer Computations*, p. 85–103. Plenum Press.

Koivisto, M. (2006). An o*(2ˆ n) algorithm for graph coloring and other partitioning problems via inclusion–exclusion. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, p. 583–590. IEEE.

Laguna, M. e Martí, R. (2001). A grasp for coloring sparse graphs. *Computational optimization and applications*, 19(2):165–178.

Leighton, F. T. (1979). A graph coloring algorithm for large scheduling problems. *Journal of research of the national bureau of standards*, 84(6):489–506.

Lü, Z. e Hao, J.-K. (2010). A memetic algorithm for graph coloring. *European Journal of Operational Research*, 203(1):241–250.

Lucet, C., Mendes, F. e Moukrim, A. (2006). An exact method for graph coloring. *Computers & operations research*, 33(8):2189–2207.

Lund, C. e Yannakakis, M. (1994). On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981.

Ruiz, I. C. R. (2012). *Gravitational Swarm for Graph Coloring*. PhD thesis, The University of the Basque Country.

Zuckerman, D. (2007). Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(6).