UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FRANCISCO LEONARDO BATISTA MARTINS

ALGORITMOS EXATOS PARA O PROBLEMA DE COLORAÇÃO DE GRAFOS

QUIXADÁ

2018

FRANCISCO LEONARDO BATISTA MARTINS

ALGORITMOS EXATOS PARA O PROBLEMA DE COLORAÇÃO DE GRAFOS

Monografia apresentada no curso de Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Ciência da Computação. Área de concentração: Computação.

Orientador: Prof. Dr. Wladimir Araujo Tavares

QUIXADÁ

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M343a Martins, Francisco Leonardo Batista.
Algoritmos exatos para o problema de coloração de grafos / Francisco Leonardo Batista Martins. – 2018.
29 f.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2018.
Orientação: Prof. Dr. Wladimir Araujo Tavares.

1. Coloração de grafos. 2. Backtracking . 3. Algoritmos. I. Título.

CDD 004

FRANCISCO LEONARDO BATISTA MARTINS

ALGORITMOS EXATOS PARA O PROBLEMA DE COLORAÇÃO DE GRAFOS

Monografia apresentada no curso de Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Ciência da Computação. Área de concentração: Computação.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

Prof. Dr. Wladimir Araujo Tavares (Orientador)
Universidade Federal do Ceará – UFC

Prof. Dr. Arthur Rodrigues Araruna
Universidade Federal do Ceará - UFC

Prof. Me. Francisco Erivelton Fernandes de Aragão
Universidade Federal do Ceará - UFC

Aos meus pais Francisco e Carmem.

AGRADECIMENTOS

Agradeço a Deus em primeiro lugar, pois ele acreditou em mim mesmo quando eu não acreditei. A toda a minha família que serviu de apoio e suporte sempre que precisei, principalmente ao meu pai, Francisco, e a minha mãe, Carmem, que fizeram o possível para que eu tivesse as oportunidades que eles nunca tiveram.

Agradeço aos meus amigos da turma dos Computeiros: Alex, Arthur, Décio, Enoque, Fransa, João Henrique, João Pedro, Juliana, Leo, Lucas, Marcelo, Olímpio, Pedro Henrique, Robson, Segundo, Sávio, Thomas, Vinicius e Yuri, por todos os bons momentos de descontração que vocês me proporcionaram. E serem a turma mais unida do campus e me ajudarem sempre que possível. Além de fazerem os melhores rachas de qualidade questionável que eu já vi.

Agradeço aos meus amigos que frequentam a capela de São Francisco da minha comunidade. Por me ajudarem a manter a paz de espirito que eu precisava durante essa fase bem complicada da minha vida. E a todos que me colocaram em suas orações.

Agradeço aos meus amigos que são como irmãos : Lukas e Wesley que me apoiaram desde criança, e durante esses quatro anos não foi diferente. Espero dá ainda muitas aulas de futebol a vocês.

Queria agradecer ao Prof. Paulo de Tarso pela paciência e por sua dedicação como professor e Coordenador de curso. Queria agradecer ao Prof. Wladimir por se mostrar um grande amigo além de orientador e professor. Queria agradecer a todos os docentes e servidores da UFC de Quixadá por fazerem um grande trabalho e engrandecerem nosso campus. E queria agradecer a todos os professores que me ajudaram a chegar até aqui. E queria agradecer a todos que me ajudaram em minha formação, e contribuiram pra essa vitória.

"E sabemos que todas as coisas contribuem juntamente para o bem daqueles que amam a Deus, daqueles que são chamados segundo o seu propósito."

(Apóstolo Paulo)

RESUMO

Neste trabalho são apresentadas algumas das principais estratégias para reduzir o tempo do algoritmo de *Backtracking* para a coloração de grafos. No algoritmo de *Backtracking* podemos alterar a estratégia de ramificação e os limites inferior e superior que são usados pelo algoritmo original. São apresentados dois algoritmos de coloração, o *BTSL* e o *BTDSATUR* que foram elaborados a apartir de algumas alterações no algoritmo de *Backtraking*. Nessas alterações são apresentadas propostas para a estratégia de ramificação do *Backtracking*: uma forma estática baseada na ordenação *SL* e uma forma dinâmica baseada no algoritmo *DSATUR*. E em relação ao limite superior ele será indicado pelo número de cores utilizadas pelo algoritmo de coloração sequencial com a ordem baseada na ordem *SL* para o *BTSL* e o número de cores utilizadas pelo algoritmo de coloração *DSATUR* no caso do Algoritmo *BTDSATUR*. Além disso um novo algoritmo será proposto para esse o problema de coloração de grafos, o *BTH*, que será um algoritmo híbrido que combinará os dois algoritmos (*BTSL* e *BTDSATUR*), dividindo uma porcentagem de vértices que será colorido por cada algoritmo. Por fim, é feita uma análise de desempenho dos algoritmos baseada no tempo e no número de nós gerados pra cada grafo de entrada nos experimentos que foram propostos. E a partir dos experimentos podemos notar que a proposta utilizada, mesmo sendo melhor que a do algoritmo *BTSL*, não consegue superar o *DSATUR*.

Palavras-chave: Coloração de Grafos. Backtracking. Algoritmos.

ABSTRACT

In this work we present some of the main strategies to reduce the time from the *Backtracking* algorithm for coloring graphs. In the Backtracking algorithm we can change the branching strategy and the lower and upper bounds that are used by the original algorithm. Two coloring algorithms, *BTSL* and *BTDSATUR*, are presented, which were elaborated from some changes in the Backtracking algorithm. In these changes proposals are presented for the Backtracking branching strategy: static form based on the ordering *SL* and dynamic form based on the *DSATUR* algorithm. With regard to the upper limit, it will be indicated by the number of colors used by the sequential coloring algorithm, with order based on the SL order for the BTSL, and the number of colors used by the DSATUR coloring algorithm, in the case of the *BTDSATUR* algorithm. In addition, a new algorithm will be proposed for this graph coloring problem, the *BTH*, that will be a hybrid algorithm that will combine the two algorithms (*BTSL* and *BTDSATUR*), dividing a percentage of vertices that will be colored by each algorithm. Finally, we provide a algorithm performance analysis based on the time and number of nodes generated for each input graph in the experiments that were proposed. From the experiments, we can notice that the proposed strategy used, despite being better than the *BTSL* algorithm, does not overcome *DSATUR*.

Keywords: Graph Coloring. Backtracking. Algorithms.

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Um grafo é um par (V,E) em que V é um conjunto arbitrário e E é um subconjunto de V tal que $V \neq \emptyset$. Os elementos de V são chamados vértices e os de E são chamados arestas (FEOFILOFF; KOHAYAKAWA; WAKABAYASHI, 2011). Dado um grafo $G = (V,E)$, uma k-coloração dos vértices é uma função $c : V \rightarrow \{1,2,...,k\}$, tal que $c(v) \neq c(u)$ para $(u,v) \in E$. Um grafo G é k-colorível se existe uma k-coloração de G. O menor inteiro k tal que G é k-colorível representa o número cromático de G, e pode ser denotado por $\chi(G)$. O *Problema da Coloração Mínima* (PCM) de G consiste em encontrar uma $\chi(G)$-coloração de G.

Existem vários problemas que podem ser resolvidos usando coloração de grafos:

- Alocação de registradores (APPEL, 2004).
- Escalonamento de horários escolares (SILVA; SILVA, 2010).
- Tráfego aéreo (BARNIER; BRISSET et al., 2004).
- Alocação de comprimento de ondas em redes ópticas *WDM* (ZANG et al., 2000).

Em geral, problemas envolvendo eventos conflitantes (que não podem ocorrer simultaneamente) podem ser modelados usando coloração em grafos (SILVA et al., 2016).

O *PCM* é um problema amplamente estudado, porém, por ser um problema *NP*-Completo (GAREY; JOHNSON, 1979), não existe teoricamente um algoritmo que o resolva em tempo polinomial a menos que $P = NP$. E esse caráter *NP*-Completo faz com que esse problema se torne custoso para grandes instâncias.

Existem duas principais abordagens para o *PCM*: heurística e exata. As heurísticas se caracterizam por encontrar soluções sub-ótimas para o problema de maneira mais eficiente em relação ao tempo, e podem ser utilizadas para encontrar limites superiores para os algoritmos exatos. Os algoritmos exatos conseguem encontrar uma solução demonstradamente ótima.

Dentre os algoritmos heurísticos, podemos destacar as heurísticas de coloração gulosa sequencial: *SL* (*smallestlast*) (MATULA; BECK, 1983), *LF* (WELSH; POWELL, 1967) e *DSATUR* (BRÉLAZ, 1979). Também existem os baseados em meta-heurísticas: *Busca TABU* como *TABUCOL* apresentado em (GLOVER, 1989), têmpera simulada (JOHNSON et al., 1989), algoritmo híbrido evolucionário (GALINIER; HAO, 1999) e algoritmo de colônia de formigas (BUI et al., 2008).

Algoritmos de *branch-and-cut* (MÉNDEZ-DÍAZ; ZABALA, 2006) e *branch-and-price* (MEHROTRA; TRICK, 1996) são empregados tentando atacar o problema da simetria. Vários outros algoritmos exatos surgem a partir de algoritmos *Backtracking* com estratégia de

poda (BROWN, 1972; BRÉLAZ, 1979; PEEMÖLLER, 1983).

Neste trabalho, vamos apresentar e analisar as principais estratégias que visam reduzir o tempo do algoritmo exato utilizando *Backtracking*. No algoritmo de *Backtracking* podemos alterar a estratégia de ramificação e os limites inferior e superior que são usados pelo Algoritmo 1.

O restante deste documento está organizado do seguinte modo: O Capítulo 2 apresenta uma base teórica para os principais conceitos apresentados neste trabalho. O Capítulo 3 mostra alguns dos principais trabalhos da literatura que se relacionam com a coloração exata de vértices. O Capítulo 4 apresenta os algoritmos elaborados que foram implementados para os experimentos. O Capítulo 5 apresenta e comenta os resultados. Por fim, o Capítulo 6 apresenta as considerações finais deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Ao longo deste trabalho usaremos alguns termos próprios da teoria dos grafos e análise de algoritmos, que são essenciais para a apresentação do conteúdo. Serão apresentados a seguir alguns desses termos, com a intenção de fundamentar posteriores análises.

2.1 Grafos

Formalmente, podemos definir um grafo G como um par ordenado $G = (V,E)$ composto por um conjunto finito V, cujos elementos são denominados vértices, e por um conjunto $E \subset \{(u,v) : u, v \in V, u \neq v\}$, cujos elementos são denominados arestas. Para todo grafo G, denotamos por $V(G)$ e $E(G)$, respectivamente, os conjuntos de vértices e arestas de G. Abaixo estão listados alguns termos relacionados a grafos que serão utilizados neste trabalho:

- **Vizinhança**: $N(v) = \{ u \in V : (v, u) \in E \}$ é a vizinhança de um vértice v em G, cujos elementos são denominados *vizinhos* de v.
- **Grau**: O grau é denotado por $deg(v)$, e corresponde ao número de vizinhos de v no grafo G.
- **Número Cromático**: O menor número k para o qual existe uma k-coloração do grafo G é chamada de número cromático do grafo G e denotada por $\chi(G)$.
- **Clique**: É um subconjunto de vértices $S \subset V(G)$ onde o grafo $G[S]$ é completo.
- **Grau de Saturação**: É o número de cores diferentes que foram usadas para colorir os vértices que pertencem à vizinhança ($N(v)$) de v.

2.1.1 Coloração de Grafos

A coloração de grafos consiste em atribuir cores aos vértices ou arestas de determinado grafo. Como iremos tratar apenas de coloração de vértices, a partir desta seção sempre que nos referirmos a uma coloração de um grafo, fica subentendido que se trata de uma coloração de vértices. Além disso, podemos definir o problema abordado neste trabalho como encontrar o número de cores utilizadas na coloração ótima de um grafo, denominado número cromático. Onde o número cromático de um grafo $\chi(G)$ representa o menor número de cores necessária para colorir um grafo G, tal que vértices adjacentes não tenham a mesma cor (ALVES et al., 2015)

2.2 Algoritmos Exatos Para o Problema de Coloração de Grafos

Como citado anteriormente, os algoritmos exatos conseguem encontrar uma solução demonstradamente ótima. Dentre os algoritmos exatos para esse problema temos algoritmos baseados em programação linear, como *Branchandbound* e *Branchandprice*, e algoritmos de enumeração implícita *Backtracking*. Neste trabalho iremos abordar especificamente algoritmos baseados em *Backtracking*. Nas próximas seções iremos apresentar as definições e/ou algoritmos que serviram como base para os algoritmos propostos neste trabalho.

2.2.1 Backtracking

O *Backtracking* é um método que gera uma árvore com todas as possíveis soluções candidatas a partir da instância de entrada. Em geral, o *Backtracking* utiliza como estratégia de busca a busca em profundidade. Cada nó da árvore pode ser representado pela seguinte tripla (C,U,q), sendo que $C \subseteq V$ representa os vértices coloridos, $U \subseteq V$ representa os vértices não coloridos e q representa a quantidade de cores da melhor coloração encontrada até o momento. Um nó da árvore (C,U,q) pode ser podado, quando o número de cores utilizadas para colorir C até o momento for maior ou igual a q. Em alguns casos, pode ser usada uma estimativa para o número máximo de cores que ainda restam para colorir U como critério de poda (LIMA, 2017). Esse método é utilizado principalmente em algoritmos *NP*-Completos como último recurso (PRIESTLEY; WARD, 1994), por ser um método de força bruta.

Este trabalho usará como algoritmo base uma implementação do *Backtracking*, como podemos observar no Algoritmo 1. O Algoritmo 1 recebe três parâmetros: o grafo G, um vetor de cores que armazena as cores de cada vértice e um inteiro k representando o número de vértices já coloridos. Se o número de vértices coloridos for igual $|V|$, significa que encontramos uma coloração de G. A priori, todas as cores estão disponíveis para o vértice não colorido, exceto as cores utilizadas pelos vértices na sua vizinhança. Depois, precisamos gerar todas as opções de cores para o vértice não colorido. Observe que, inicialmente, precisamos encontrar o número máximo de cores utilizadas na coloração parcial atual.

Algoritmo 1: BACKTRACKING

```
Entrada: G, cores, k
início
    se k == |V| então
        retorna número de cores utilizadas
    fim
    senão
        v ← primeiro vértice não colorido
        mColor ← maior cor utilizada no vetor de cores
        disponivel[i] ← true  ,∀i ∈ [1...|V|]
        para u ∈ V(G) faça
            se {v,u} ∈ E(G) então
                se cor[u] ≠ −1 então
                    disponivel[cor[u]] ← false
                fim
            fim
        fim
        para i = 1 até mColor + 1 faça
            se disponivel[i] == true então
                cor[v] ← i
                Backtracking(G, cores, k + 1)
                cor[v] ← −1
            fim
        fim
    fim
fim
```

Fonte – Produzido pelo próprio autor

2.3 Algoritmos Heurísticos Para o Problema de Coloração de Grafos

Como o problema da coloração de grafos é *NP*-Completo, é fácil ver que para entradas muito grandes o problema se torna inviável de se resolver de forma exata em tempo hábil, então existe uma grande demanda por algoritmos heurísticos, que mesmo não obtendo sempre uma solução ótima podem devolver uma solução sub-ótima em tempo polinomial. Além disso, nesse trabalho eles serão usados como auxiliares para uma coloração exata da seguinte maneira: O número de cores utilizados poderão servir como entrada para os algoritmos exatos na forma de limite superior, e a ordem em que os vértices serão coloridos será usada no processo de ramificação do algoritmo de *Backtraking*. Dentre esses algoritmos heurísticos, podemos destacar os dois que serão utilizados neste trabalho, que são: A *coloração gulosa sequencial* e o algoritmo *DSATUR*.

2.3.1 Coloração Gulosa sequencial

Dado um grafo G e uma sequência de vértices $K = (v_1, v_2, ..., v_n)$ que representa uma ordem de coloração dos vértices. O algoritmo de coloração Gulosa Sequencial em cada iteração seleciona um vértice na ordem da sequencia da entrada K, e colore esse vértice com a cor que apresenta o menor índice possível (Algoritmo 2).

Algoritmo 2: Coloração Gulosa Sequencial

```
Entrada: G, K
início
    para v ∈ K(Na ordem de K) faça
        cor[v] ← Cor com menor indice e não utilizada por vizinhos
    fim
fim
```

Fonte – Produzido pelo próprio autor

2.3.2 DSATUR

O algoritmo *DSATUR* (BRÉLAZ, 1979) diferentemente do guloso sequencial, tem sua ordem definida dinamicamente, pois apenas o primeiro vértice colorido é definido préviamente como sendo o de maior grau. A ordem dos demais vértices a serem coloridos é definida pelo grau de saturação. O vertice v que tiver o maior grau de saturação é colorido com a cor de menor índice não utilizada por vizinhos. Em caso de empate o vértice entre os que empataram que tiver o maior grau é colorido, e se o empate persistir, o vértice entre os que empataram é escolhido de forma aleatória (Algoritmo 3).

Algoritmo 3: DSATUR

```
Entrada: G, cores, k
início
    ordena vertices V(G) em ordem decrescente de graus
    v_0 ← vertice com maior grau
    cor[v_0] ← 0
    para v_i ∈ {V − {v_0}} faça
        v_i ← vertice com maior grau de saturação
        cor[v_i] ← menor cor disponível
    fim
fim
```

Fonte – Versão baseada em (BRÉLAZ, 1979)

3 TRABALHOS RELACIONADOS

Brown (1972) apresentou um algoritmo exato que utiliza o *Backtracking*. Nesse algoritmo, a partir de uma ordenação inicial $(v_1, v_2, ..., v_n)$, os vértices serão coloridos sequencialmente. Dado k como limitante superior global para uma coloração de G e q um limitante inferior para cada nova coloração parcial da árvore de soluções tal que $q < k$, quando uma coloração completa utiliza menos do que k cores, então o valor de k é atualizado. Se em uma coloração parcial i não é possível colorir o subgrafo induzido pelos vértices $\{v_1, v_2, ..., v_i\}$ com menos do que k cores, então não é necessário continuar a respectiva coloração. Dessa maneira o algoritmo enumera implicitamente todas as soluções de uma instância do problema.

Brélaz (1979) desenvolveu uma heurística de coloração chamada *DSATUR*, que é exato para grafos bipartidos. Além disso, ele apresenta uma modificação do método para coloração exata a partir do trabalho de Brown (1972). Esse algoritmo, *Randall-Brown's Modified Algorithm (RBMA)*, se difere do algoritmo de Brown (1972) principalmente na ordem dos vértices a serem coloridos, onde a ordem é alterada dinamicamente do seguinte modo: O vértice i a ser colorido é o que contém o maior grau de saturação, em caso de empate o vértice que tiver maior número de vizinhos é escolhido e se persistir o empate um dos que empataram é colorido aleatoriamente.

Peemöller (1983) expõe dois erros que podem acontecer ao algoritmo *(RBMA)* Brélaz (1979) e afetar a eliminação de uma solução ótima. Além disso, ele apresenta uma versão correta do algoritmo e prova sua exatidão.

Segundo (2012) descreve um novo algoritmo baseado no *DSATUR*, onde propõe uma nova estratégia de desempate para o *DSATUR*, que é muito menos dispendiosa computacionalmente do que *SEWELL* (SEWELL, 1996), mas tem efeito de poda semelhante, referido como a regra *PASS*. Na regra de *PASS* se escolhe o vértice que compartilha mais cores disponíveis com os outros vértices que estão empatados.

Neste trabalho, iremos implementar algumas versões do algoritmo de *Backtracking* baseado no trabalho de Brown (1972) com as modificações de Brélaz (1979) e as correções de Peemöller (1983). Nesse algoritmo base *(BT)*, foi implementado o *Backtracking* simples de maneira recursiva e foi a partir desse algoritmo que foram implementados os aprimoramentos.

4 LEVANTAMENTO DE ALGORITMOS EXATOS DE COLORAÇÃO

Usando o *Backtracking* podemos resolver o problema de coloração de grafos, porém, como o método é de força bruta e o problema é NP-Completo, não podemos encontrar uma solução ótima para o problema com um tempo razoável. Com isso, teremos que utilizar outros artifícios para reduzir o espaço de busca para as entradas maiores. Mais precisamente temos algumas estratégias tipicamente utilizadas no estado da arte desse problema, como: O limite superior, o limite inferior e a ordem em que os vértices serão coloridos pelo algoritmo. Nesse trabalho, nos limitamos a trabalhar com o limite superior e a ordenação dos vértices, que, com base em estudos anteriores, possuem resultados mais eficientes.

O limite superior funciona como um "teto" que é declarado de maneira global e é atualizado sempre que uma coloração menor que a mínima atual é encontrada. E se a coloração atual utiliza mais cores que esse "teto", então o algoritmo retorna para um vértice anterior fazendo assim a "poda" da subárvore dessa recursão. Esses limites superiores podem ser desde a quantidade de vértices de um grafo, até o número de cores utilizadas por um algoritmo de coloração heurística, entre outros meios mais complexos.

A ordem de coloração de cada vértice do grafo contribui diretamente na etapa de ramificação do algoritmo, podendo em alguns casos aumentar ou diminuir a velocidade com que o algoritmo encontra uma coloração mínima dos vértices. Essa ordem pode ser definida estaticamente, ou seja, antes da execução do *Backtracking*, ou dinamicamente durante a execução do algoritmo, definindo qual o vértice que será escolhido para ser colorido em cada interação. Nessa ordem também deve-se ter em mente que buscamos chegar em um equilíbrio de custo-benefício nesse processo, pois com uma boa ordenação é possível que o algoritmo possa chegar mais rapidamente em uma coloração mínima. Entretanto, se para obtermos tal coloração, precisarmos de um método computacionalmente custoso, talvez não obtenhamos um resultado viável, por que o custo de encontrar a ordem pode comprometer o desempenho geral do algoritmo.

A seguir serão apresentados os algoritmos que foram implementados com base no algoritmo *BTB* (Algoritmo 1), com o intuito de melhorar o desempenho do algoritmo em relação ao tempo.

- **Algoritmo (BTSL)**: O algoritmo *BTSL* é uma variação do algoritmo *BT*, em que a ordem de ramificação dos vértices seguirá uma critério de ordenação chamado smallest-last (*SL*) (Algoritmo 5) proposto em (MATULA; BECK, 1983). Nesse critério de ordenação, a

ordem dos vértices $(v_1, v_2, \ldots, v_n)$ é tal que v_n é o vértice de menor grau em $G[V]$ e v_i é o vértice de menor grau em $G[V \setminus \{v_{i+1}, \ldots, v_n\}]$. Além dessa ordem ser utilizada como critério, utilizaremos essa ordem para encontrar uma coloração inicial usando um método de coloração gulosa sequencial. Essa coloração inicial será utilizada como limite superior inicial. O número de cores encontradas na coloração gulosa é armazenado na variável $teto$ e é usado como limite superior para o algoritmo *BTSL*.

Algoritmo 4: BTSL

```
Entrada: G, ordem, cores, k
início
    mColor ← maior cor utilizada no vetor de cores
    se mColor >= teto então
        retorna para o nó superior
    fim
    se k == |V| então
        teto ← mColor
        retorna número de cores utilizadas
    fim
    senão
        v ← primeiro vértice não colorido (Seguindo a ordem SL)
        mColor ← maior cor utilizada no vetor de cores
        disponivel[i] ← true  ,∀i ∈ {1,...,|V|}
        para u ∈ V(G) faça
            se {v,u} ∈ E(G) então
                se cor[u] ≠ −1 então
                    disponivel[cor[u]] ← false
                fim
            fim
        fim
        para i = 1 até mColor + 1 faça
            se disponivel[i] == true então
                cor[v] ← i
                BTSL(G, ordem, cores, k + 1)
                cor[v] ← −1
            fim
        fim
    fim
fim
```

Fonte – Produzido pelo próprio autor

Algoritmo 5: ORDENAÇÃO SL

Entrada: G, ord
início
  $n \leftarrow |V(G)|$
  while $n > 0$ **do**
    Seja v o vértice com o menor grau em $G[V \setminus K]$
    $ord[n] \leftarrow v$
    $K \leftarrow K \cup \{v\}$
    $n \leftarrow n - 1$
  end
fim

Fonte – Versão baseada na descrição em (MATULA; BECK, 1983)

O intuito dessa ordenação é a cada iteração deixar sempre o vértice com menos restrições para a coloração (menor grau) para ser colorido por último. Um limite superior para o número de cores utilizado pela ordenação SL na heurística de coloração sequencial é:

$$SL(G) \leq \max_{i \leq n} \min_{j \leq i} deg_{G[v_1,\ldots,v_i]}(v_j) + 1 \tag{4.1}$$

onde $deg_{G[v_1,\ldots,v_i]}(v_j)$ denota o grau do vértice v_j no subgrafo induzido pelos vértices $v_1,\ldots,v_i$ e $v_1,\ldots,v_n$ é uma ordenação SL.

- **Algoritmo *BTDSATUR***:

  O algoritmo *BTDSATUR* é uma variação do algoritmo *BT*, em que a ordem de ramificação será dinâmica e baseada no grau de saturação. Para cada subproblema (C,U,q), o vértice escolhido para ser colorido será o vértice $v \in U$ com o maior grau de saturação. Esse critério de ramificação garante que o vértice escolhido terá a menor quantidade de cores disponíveis para ele. Dessa maneira, reduziremos o fator de ramificação do problema. Porém, o custo computacional para encontrar o vértice com o maior grau de saturação tem a complexidade $\mathcal{O}(V^2)$ utilizando matriz de adjacência e $\mathcal{O}(V+E)$, utilizando lista de adjacência. Para evitar recálculos desnecessários no algoritmo de Backtracking, mantemos uma matriz de saturação que relaciona os vértices e as cores que já foram utilizadas por seus vizinhos. Quando um novo vértice é colorido, essa matriz de incidência é alterada e passada como parâmetro para o novo subproblema. Com isso, buscamos evitar um esforço computacional exagerado.

Algoritmo 6: BTDSATUR

Entrada: $G, MSaturacao, cores, k$
início
  $MSaturacaoLocal \leftarrow$
   declara uma Matriz vazia que irá guardar os graus de saturação dos vértices
  $mColor \leftarrow$ maior cor utilizada no vetor de cores
  se $mColor >= teto$ **então**
    retorna para o nó superior
  fim
  se $k == |V|$ **então**
    $teto \leftarrow mColor$
    retorna número de cores utilizadas
  fim
  senão
    $v \leftarrow$ vertice com maior grau de saturação
    $mColor \leftarrow$ maior cor utilizada no vetor de cores
    $disponivel[i] \leftarrow true \ \ ,\forall i \in \{1,\ldots,|V|\}$
    para $u \in V(G)$ **faça**
      se $\{v,u\} \in E(G)$ **então**
        se $cor[u] \neq -1$ **então**
          $disponivel[cor[u]] \leftarrow false$
        fim
      fim
    fim
    para $i = 1$ *até* $mColor + 1$ **faça**
      se $disponivel[i] == true$ **então**
        $MSaturacaoLocal \leftarrow MSaturacao\ cor[v] \leftarrow i$
        $MSaturacaoLocal \leftarrow$ *e atualizada*
         $BTDSATUR(G, MSaturacaoLocal, cores, k+1)$
        $cor[v] \leftarrow -1$
      fim
    fim
  fim
fim

Fonte – Produzido pelo próprio autor

- **Algoritmo BTH**:

  O algoritmo *BTH* é uma variação do algoritmo *BT*, onde o *BTDSATUR* e o *BTSL* são combinados. O algoritmo *BTDSATUR* é chamado primeiramente passando uma porcentagem como parâmetro, e começa o fluxo natural de sua execução, até que a porcentagem do numero de vértices coloridos se iguale ou passe pelo valor da porcentagem estipulada na entrada. Então quando isso ocorre o algoritmo solicita que o *BTSL* seja executado e termine a coloração. A ideia central nesse algoritmo é de que, como a coloração do *BTSL* é bem mais simples, ela pode custar menos que o *BTDSATUR*

quando se tem poucos vértices pra se colorir e são somente vértices com baixo grau de saturação, pois não foram coloridos pelo *BTDSATUR*.

5 RESULTADOS

Os algoritmos foram implementados na linguagem de programação $C++$ e executados em um computador com o processador *Intel(R) Core(TM) i*$5-7200U$ *CPU* @ $2.50GHz$ $2.70GHz$, 8 *GB* de memória *RAM* e *Linux 64 bit*. Neste experimento, geramos 10 grafos aleatórios para cada combinação *n* (número de vértices) e *p* (probabilidade de existência de cada aresta) considerada. Cada aresta é incluída no grafo com probabilidade (por influenciar diretamente na quantidade de arestas, também pode ser chamada de *densidade*) *p*, independentemente de todas as outras arestas. Logo, todos os grafos com *n* vértices e *m* arestas têm a mesma probabilidade de serem gerados, que é definida por:

$$p^m(1-p)^{\frac{n(n-1)}{2}-m} \tag{5.1}$$

Para os experimentos foram utilizados três variações do algoritmo *BTH*, com 50%, 70% e 90% de vértices coloridos pelo *BTDSATUR* e 50%, 30% e 10% respectivamente pelo *BTSL*. O número de vértices atribuídos ao *BTDSATUR* são bem maiores pelo fato de que ele foi melhor nos testes iniciais em relação ao *SL*.

Na Tabela 1, Cada valor apresenta a média aritmética do tempo da execução de cada algoritmo em segundos para 10 grafos aleatórios com a mesma probabilidade de geração de arestas (densidade) e tamanho. Na Tabela 2, é apresentado o tempo médio da execução de cada algoritmo em segundos para grafos com tamanhos 20, 30, 40 e 50. Para cada tamanho de grafo foram feitos testes com 9 densidades diferentes (de 10 a 90%), e com 10 grafos para cada combinação *n* (número de vértices) e *p* (probabilidade de existência de cada aresta), como mostrado na Tabela 1. A Tabela 3, mostra a quantidade de nós média utilizada na execução de cada algoritmo, utilizando o mesmo modelo da Tabela 1.

Tabela 1 – Tempo Médio de Execução dos algoritmos - Por quantidade de *vértices* e *densidade*

N. de Vértices/Algoritmo	Densidade (%)	BT-DSATUR	BT-SL	BTH-50	BTH-70	BTH-90
	10	0,0022	0,0005	0,0008	0,0006	0,0005
	20	0,0016	0,0005	0,0008	0,0006	0,0005
	30	0,0015	0,0009	0,0011	0,0007	0,0007
	40	0,0011	0,0011	0,0019	0,0010	0,0008
20	50	0,0013	0,0009	0,0011	0,0008	0,0009
	60	0,0021	0,0009	0,0010	0,0008	0,0008
	70	0,0014	0,0014	0,0010	0,0008	0,0007
	80	0,0017	0,0020	0,0017	0,0009	0,0009
	90	0,0010	0,0006	0,0010	0,0004	0,0004
	10	0,0038	0,0020	0,0026	0,0011	0,0012
	20	0,0032	0,0031	0,0039	0,0025	0,0022
	30	0,0062	0,0070	0,0088	0,0051	0,0047
	40	0,0055	0,0055	0,0070	0,0044	0,0045
30	50	0,0178	0,0270	0,0239	0,0205	0,0204
	60	0,0113	0,0191	0,0180	0,0168	0,0170
	70	0,0081	0,0271	0,0276	0,0246	0,0215
	80	0,0077	0,0362	0,0270	0,0273	0,0391
	90	0,0032	0,0032	0,0030	0,0029	0,0039
	10	0,0083	0,0043	0,0029	0,0029	0,0045
	20	0,0147	0,0400	0,0298	0,0281	0,0337
	30	0,0244	0,0673	0,0676	0,0616	0,0667
	40	0,0213	0,1423	0,1525	0,2224	0,1572
40	50	0,1607	0,9280	1,6350	1,4207	1,0504
	60	0,1236	1,2176	0,6527	1,1275	0,5228
	70	0,0545	0,2625	0,2864	0,5289	0,2510
	80	0,0586	0,6089	0,6978	0,9617	0,5933
	90	0,0079	0,0824	0,1039	0,1401	0,1058
	10	0,0131	0,0069	0,0086	0,0135	0,0085
	20	0,0170	0,0551	0,1047	0,1207	0,0899
	30	0,1921	2,7259	1,8052	2,2673	1,4596
	40	0,2987	46,9184	65,5255	65,9251	45,2075
50	50	1,1384	36,4103	42,0774	40,2935	31,3197
	60	1,8825	30,5259	48,6310	49,4613	32,4987
	70	1,2283	128,9000	124,3290	92,1134	99,0091
	80	0,7842	28,4834	31,3547	24,9744	18,3858
	90	0,0547	4,2339	4,1412	5,1098	4,0045

Fonte – Produzido pelo próprio autor

Nota – Cada valor apresenta a média aritmética de 10 grafos aleatórios com a mesma probabilidade de geração de arestas

Tabela 2 – Tempo Médio de Execução dos Algoritmos

N. de Vértices/Algoritmo	BT-DSATUR	BT-SL	BTH-50	BTH-70	BTH-90
20	0,001544444	0,000977778	0,001155556	0,000733333	0,000688889
30	0,007422222	0,014466667	0,013533333	0,011688889	0,012722222
40	0,052666667	0,372588889	0,403177778	0,499322222	0,309488889
50	0,623222222	30,917755556	35,330811111	31,142111111	25,775922222

Fonte – Produzido pelo próprio autor

Nota – Cada linha representa a média aritmética dos resultados para grafos de todas as densidades(tabela 1) com a mesma quantidade de vértices

Tabela 3 – Número Médio de Nós - Completo

N. de Vértices/Algoritmo	Densidade (%)	BT-DSATUR	BT-SL	BTH-50	BTH-70	BTH-90
	10	48	49	50	50	50
	20	47	52	55	55	55
	30	51	53	61	61	61
	40	54	96	92	92	92
20	50	50	67	80	80	80
	60	55	70	85	85	85
	70	45	90	91	91	91
	80	48	110	111	111	111
	90	34	35	35	35	35
	10	79	79	80	80	80
	20	82	169	175	175	175
	30	137	400	377	377	377
	40	141	314	360	360	360
30	50	403	1697	1759	1759	1759
	60	235	1333	1454	1454	1454
	70	179	1759	1847	1847	1847
	80	154	2572	2244	2244	2244
	90	61	221	222	222	222
	10	116	184	174	174	174
	20	237	2001	1908	1908	1908
	30	395	3465	4065	4065	4065
	40	369	7818	9272	9272	9272
40	50	2816	44053	66773	66773	66773
	60	2153	44995	32832	32832	32832
	70	976	15529	15687	15687	15687
	80	1035	32115	32826	32826	32826
	90	134	4491	4222	4222	4222
	10	172	326	349	349	349
	20	238	2568	3645	3645	3645
	30	2690	128501	77847	77847	77847
	40	4269	2366386	2398889	2398889	2398889
50	50	16068	1694007	1584079	1584079	1584079
	60	26633	1524403	1795880	1795880	1795880
	70	17055	5292791	5112762	5112762	5112762
	80	11308	906301	907300	907300	907300
	90	771	188536	197111	197111	197111

Fonte – Produzido pelo próprio autor

Nota – Cada valor apresenta a média aritmética do número de nós gerados pelo algoritmo indicado em sua coluna para 10 grafos aleatórios com a mesma probabilidade de geração de arestas e mesmo número de vértices

6 CONCLUSÕES E TRABALHOS FUTUROS

Nesse trabalho foram implementados alguns algoritmos de *Backtracking* para o problema de coloração de Grafos, que utilizam algumas estratégias com o objetivo de aumentar o desempenho dos algoritmos em relação a tempo. Os algoritmos apresentados foram: *BTSL*, *BTDSATUR* e uma combinação dos dois algoritmos *BTH*.

A partir dos resultados obtidos nos experimentos, podemos notar que o algoritmo *BTSL* ganha (em relação ao tempo) em alguns casos específicos onde os grafos são menores e menos densos. O que parece ser intuitivo, pois ele tem sua ordem definida de uma maneira mais simples por ser estática. Entretanto para grafos maiores e mais densos pode-se notar uma grande vantagem do *BTDSATUR* em relação aos demais algoritmos tanto no tempo quanto no número de nós utilizados pelos algoritmos.

Em relação ao algoritmo *BTH* é possível observar que ele teve vantagem sobre o *BTSL* pra grande parte dos grafos, com exceção dos grafos menores e menos densos. E em comparação com o *BTDSATUR* ele ficou em desvantagem na maioria dos grafos de entrada, exceto os grafos menores e menos densos. Portanto, conjecturamos que esse resultado se deu pelo fato de que ele herdou as principais características dos algoritmos por ele utilizados. E essas características ficam bem mais explícitas quando se observa a diferença entre os algoritmos *BTH* com porcentagem diferente, quanto mais próxima do *BTDSATUR* (BTH 90%) mais rápido o algoritmo fica.

Em trabalhos futuros as instâncias podem ser maiores e mais elaboradas com densidades mais equilibradas, e outras estratégias podem ser utilizadas como: A utilização de limites inferiores e pré-colorações, além de novas propostas para a ordem de entrada para o *Backtracking* junto com a combinação de novos algoritmos.

REFERÊNCIAS

ALVES, R. P. et al. **Coloração de grafos e aplicações**. [S.l: s.n], 2015.

APPEL, A. W. **Modern compiler implementation in C**. [S.l.]: Cambridge university press, 2004.

BARNIER, N.; BRISSET, P. et al. Graph coloring for air traffic flow management. **Annals of Operations Research**, Springer, v. 130, n. 1, p. 163–178, 2004.

BRÉLAZ, D. New methods to color the vertices of a graph. **Communications of the ACM**, ACM, v. 22, n. 4, p. 251–256, 1979.

BROWN, J. R. Chromatic scheduling and the chromatic number problem. **Management Science**, INFORMS, v. 19, n. 4-part-1, p. 456–463, 1972.

BUI, T. N.; NGUYEN, T. H.; PATEL, C. M.; PHAN, K.-A. T. An ant-based algorithm for coloring graphs. **Discrete Applied Mathematics**, Elsevier, v. 156, n. 2, p. 190–200, 2008.

FEOFILOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. **Uma Introdução Sucinta à Teoria dos Grafos**. [S.l: s.n], 01 2011.

GALINIER, P.; HAO, J.-K. Hybrid evolutionary algorithms for graph coloring. **Journal of combinatorial optimization**, Springer, v. 3, n. 4, p. 379–397, 1999.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability:** a guide to the theory of np-completeness. [S.l]:, Freeman, 1979.

GLOVER, F. Tabu search—part i. **ORSA Journal on computing**, INFORMS, v. 1, n. 3, p. 190–206, 1989.

JOHNSON, D. S.; ARAGON, C. R.; MCGEOCH, L. A.; SCHEVON, C. Optimization by simulated annealing: an experimental evaluation; part i, graph partitioning. **Operations research**, INFORMS, v. 37, n. 6, p. 865–892, 1989.

LIMA, A. M. d. **Algoritmos exatos para o problema da coloração de grafos**. [S.l: s.n], 2017.

MATULA, D. W.; BECK, L. L. Smallest-last ordering and clustering and graph coloring algorithms. **Journal of the ACM (JACM)**, ACM, v. 30, n. 3, p. 417–427, 1983.

MEHROTRA, A.; TRICK, M. A. A column generation approach for graph coloring. **informs Journal on Computing**, Informs, v. 8, n. 4, p. 344–354, 1996.

MÉNDEZ-DÍAZ, I.; ZABALA, P. A branch-and-cut algorithm for graph coloring. **Discrete Applied Mathematics**, Elsevier, v. 154, n. 5, p. 826–847, 2006.

PEEMÖLLER, J. A correction to brelaz's modification of brown's coloring algorithm. **Communications of the ACM**, ACM, v. 26, n. 8, p. 595–597, 1983.

PRIESTLEY, H. A.; WARD, M. P. A multipurpose backtracking algorithm. **Journal of Symbolic Computation**, Elsevier, v. 18, n. 1, p. 1–40, 1994.

SEGUNDO, P. S. A new dsatur-based algorithm for exact vertex coloring. **Computers & Operations Research**, Elsevier, v. 39, n. 7, p. 1724–1733, 2012.

SEWELL, E. An improved algorithm for exact graph coloring. **Cliques, coloring, and satisfiability: second DIMACS implementation challenge**, American Mathematical Society: Providence, RI, v. 26, p. 359–372, 1996.

SILVA, D. J. da; SILVA, G. C. da. Heurísticas baseadas no algoritmo de coloração de grafos para o problema de alocação de salas em uma instituição de ensino superior. **Anais do XLII SBPO**, p. 2839–2849, 2010.

SILVA, T. G. N. da; ROCHA, L. S.; VALDISIO, G.; VIANA, R. Proposta e avaliação de novas heurísticas para o problema de coloração de vértices. **Simpósio Brasileiro de Pesquisa Operacional**, 2016.

WELSH, D. J.; POWELL, M. B. An upper bound for the chromatic number of a graph and its application to timetabling problems. **The Computer Journal**, Oxford University Press, v. 10, n. 1, p. 85–86, 1967.

ZANG, H.; JUE, J. P.; MUKHERJEE, B. et al. A review of routing and wavelength assignment approaches for wavelength-routed optical wdm networks. **Optical networks magazine**, v. 1, n. 1, p. 47–60, 2000.