Algoritmos para o Problema de Coloração de Grafos

Marcelo Ferreira Rego, Haroldo Gambini Santos
PPGCC - Programa de Pós-Graduação em Ciência da Computação
UFOP - Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brasil
email: marcelofr@gmail.com, haroldo.santos@gmail.com

***Resumo*—Este artigo apresenta um algoritmo metaheurístico para resolver o problema de coloração de grafos. O algoritmo proposto tem em sua estrutura duas etapas. Na primeira é gerado uma solução inicial a partir de um algoritmo guloso. Na segunda é feita uma busca local para refinar a solução inicial. O algoritmo é executado para algumas intâncias reais, com grafos de várias densidades e número de vértices. A ordem de complexidade do algoritmo apresentado é de $O(|V|^2 * |E|)$.**

***Keywords*-palavras chaves, algoritmos, complexidade.**

I. Introdução

O problema de coloração de grafos teve sua origem em 1852. Na ocasião, Frederick Guthrie, da universidade de Londres, apresentou para seu professor, o famoso matemático Augustus De Morgan, uma conjectura que daria origem a um dos mais famosos problemas de teoria dos grafos. Segundo ele seria necessário no máximo quatro cores para colorir as regiões de qualquer mapa, evitando que duas regiões adjacente tivessem a mesma cor. Fato que deu origem ao Teorema das Quatro Cores.

Em 1879, Alfred Kempe publicou um artigo provando o Teorema das Quatro Cores, entretanto, esta prova foi questionada em 1890, por Percy John Heawood, que postulou o Teorema das Cinco Cores, mas finalmente em 1976, Kenneth Appel e Wolfgang Haken demonstraram com a ajuda de um computador que o Teorema das Quatro Cores proposto por Kempe estava correto [1].

O Teorema das Quatro Cores diz que qualquer um grafo planar é 4-colorível. Entretanto, um outro problema que surge é como estabelecer uma valor mínimo de cores para colorir um grafo qualquer. Este problema é conhecido como Problema de Coloração de Grafos.

A definição formal para o Problema de Coloração de Grafos é: dado um grafo não-direcionado $G = (V, E)$, com um conjunto V de vértices e um conjunto E de arestas que incidem sobre os vértices de G, o problema de coloração de grafos consiste em atribubir k-cores para os vértices de G, de modo que dois vértices adjacentes não tenham a mesma cor. Assim a coloração pode ser considerada como uma função $c : V(G) \rightarrow \mathbb{N}$ (onde $\mathbb{N}$ é o conjunto de inteiros positivos) tal que, $c(u) \neq c(v)$ se u e v são adjacentes em G [2]. Caso k seja o número mínimo de cores necessárias para colorir o grafo, dizemos que k é o número cromático de G denotado por $\chi(G)$. Então, um grafo é k-colorível se e somente se, $\chi(G) \leq k$.

O Problema de Coloração de Grafos pode ser formulado de duas formas. Primeiro como problema de decisão, em que o objetivo é identificar se um grafo é k-colorível, neste caso ele é um problema de decisão. Na segunda formulação o objetivo é encontrar um valor de k que seja mínimo. Deste ponto em diante, toda vez que o Problema de Coloração de Grafos for mencionado, será uma referência a segunda formulação.

O estudo do problema de coloração de grafos é de grande importância teórica pois é um clássico problema NP-Completo [3]. Progressos na solução dessa classe de problemas considerados intratáveis motivam diversas pesquisas em computação, matemática e pesquisa operacional.

Do ponto de vista prático, o estudo do problema de coloração de grafos é muito importante pois sua estrutura é muito utilizada para modelar problemas de diversas áreas, como: *scheduling* [4], *timetabling* [5], otimização de alocação de registradores [6], *train platforming* [7], *frequency assignment* [8], redes de comunicação [9].

Um algoritmo heurístico consiste em um algoritmo de sentido geral para a otimização de um problema que é baseado em alguma estratégia para percorrer o conjunto de soluções viáveis. Em geral, heurísticas não são capazes de provar a qualidade da solução. Por isso a eficiência de uma heurística é medida através de testes empíricos.

Metaherísticas são algoritmos baseados em herísticas dotadas da capacidade de escapar de ótimos locais e eventualmente encontram o ótimo global. Para isso, as metaheuristicas, além de percorrem o espaço de busca das soluções viáveis permite movimentos que gere soluções inviáveis.

Uma vez que métodos exatos tem se mostrado incapazes de fornecer soluções em tempo computacional aceitável, para problemas de alta complexidade, tipicamente, problemas reais, técnicas metaheurísticas têm se mostrado extremamente eficazes para resolver problemas complexos, especialmente de natureza combinatória [10].

Portanto, o presente trabalho propõe a utilização de um modelo computacional baseado em técnicas metaherísticas para solucionar o problema de coloração de grafos.

O principal objetivo deste trabalho é apresentar e implementar um algoritmo baseado em técnicas metaheurís-

ticas para resolver o problema de coloração de grafos e realizar uma análise comparando os resultados obtidos pela implementação aqui realizada com os resultados encontrados na literatura, com a finalidade de verificar a eficiência do algoritmo apresentado. Outro objetivo é realizar a análise de complexidade dos algoritmos apresentados.

O artigo está organizado da seguinte forma. A Seção II contém uma breve resumo de alguns trabalhos relacionados, A Seção III descreve a metodologia empregada no trabalho, em seguida a Seção IV apresenta uma analise de complexidade do algoritmo desenvolvido, enquanto que a Seção V mostra os experimentos realizados e por fim a Seção VI discute os resultados obtidos.

II. Trabalhos Relacionados

Um das primeiras abordagens para o Problema de Coloração de Grafos utilizando metaheurísticas que podemos encontrar na literatura foi feita por Chams *et al.* (1987) [11]. O trabalho propôs o uso da metaheurística *Simulated Annealing* que é largamente utilizada em problema de otimização combinatória. O algoritmo proposto explora o conjunto de k-coloração de uma solução, que, não necessariamente, é viável, com o objetivo de minimizar o número de conflitos em cada aresta. O conceito de movimento consiste em mudar a cor de um vértice da solução corrente. Esses movimento são realizados com base na heurística *Simulated Annealing*. Na fase de preprocessamento várias classes de cores são criadas usando um algoritmo guloso. Na fase de busca local o algorimo *Simulated Annealing* é aplicado. O artigo mostra que a fase de preprocessamento permite melhorar os resultados obtidos pela metaheurística. Esta constatação é verificada principalmente quando o algoritmo é aplicado a grafos densos e com grade quantidade de vértices [11] *apud* [12].

Um algoritmo baseado na metaheuristica Busca Tabu foi proposto por Hertz & de Werra (1987) [13] chamado TABUCOL. O algoritmo constroi uma solução inicial utilizando uma quantidade k classe de cores. Na fase de busca local, o objetivo é eliminar os conflitos que são arestas com dois vértices incidentes que possuem a mesma cor. O movimento consiste em mudar a cor de um vértice. A função de avaliação que se deseja minimizar é o número de conflitos da coloração corrente. O algoritmo realiza apenas movimentos em que o vértice envolvido possua algum conflito. A Busca Tabu funciona a partir de uma lista chamada lista tabu que é responsável por armazenar a atribuição de cores aos vértices. Quando uma atribuição é realizada ela fica proibida de se repetir por um certo número de iterações. Este processo é chamado movimento tabu. Com isso, o algoritmo garante que uma solução gerada anteriormente, dentro do limite do tamanho da lista, não seja gerada novamente.

Uma abordagem utilizando a metaheurística GRASP foi proposta por Laguna & Martí (2001) [14] GRASP é um procedimento de busca adaptativa gulosa e randômica que consiste em criar uma lista chamada de lista restrita de candidatos, nesta lista estão os l primeiros elementos, ordenados a partir de uma função guia. A escolha de um elemento da lista é feito de forma aleatória. A fase de construção do algoritmo de Laguna & Martí é feita de maneira iterativa, sendo que a função guia para a criação da lista restrita de candidatos são os vértices de maior grau.

Paquete e Stutzle (2002) [15] propuseram a metaheurística *Iterated Local Search* é baseada na ideia simples de que um procedimento de busca local pode ser melhorado gerando-se novas soluções de partida, as quais são obtidas por meio de pertubações na solução ótima local. Este trabalho concentrou seus esforços, principalmente, na escolha da busca local e no operator de pertubação. Na fase inicial foram testados dois algoritmos, um que gera solução aleatória e outro que gera solução gulosa. A fase de busca local também teve dois algoritmos. O primeiro, escolhe aleatoriamente um entre todos os vértice com conflito para realizar um movimento, que consiste em trocar a cor de um vértice. O segundo, verifica todas as possibilidades de cores para cada vértice, e o movimento escolhido é o que fornece o menor número de conflitos. Foram analisados quatro tipos de perturbação. 1) Perturbação aleatória, que consiste em atribuir uma cor para um vértice escolhido aleatoriamente. 2) Perturbação de adição de aresta, consiste em adicionar arestas aos grafo no início da busca local, promovendo uma mudança na estrutura original da instância, e removendo estas aresta ao final do processo. 3) Perturbação de conflitos de vértices, atribui-se cores escolhidas aleatoriamente somente a vértice com conflito. 4) Perturbação de diversificação direcionada, neste método foi realizado uma combinação de dois tipos de movimentos aleatórios, um que tinham uma probabilidade w_p de acontecer, e outro com movimentos baseados em busca tabu com probabilidade de $1 - w_p$ de acontecer.

Uma estratégia baseada em algoritmos genético foi proposta por Galinier e Hao (1999) [16]. Algoritmo genético é uma técnica de busca baseada numa metáfora do processo biológico de evolução natural. Esse algoritmo considera que uma solução para o problema é um indivíduo e o conjunto de soluções é uma população. O algoritmo começa gerando uma população inicial. Cada indivíduo dessa população é obtido a partir do algoritmo guloso denominado DSATUR proposto por Brélaz (1979) [17]. O DSATUR colore primeiro os vértices de maior saturação, ou seja, aquele que tenha o maior conjunto de classe de cores em seus vértice adjacentes. A segunda etapa do algoritmo consiste em um processo de combinação dos indivíduos aos pares obtendo um terceiro indivíduo. O operador de combinação escolhe a classe de cor com maior número de vértices para compor o descendente dos indivíduos que estão sendo combinados.

III. Métodos

O método a ser seguido neste trabalho pode ser descrito em duas fases. Primeiro será aplicado uma heurística cons-

trutiva com a finalidade de gerar uma solução inicial para o problema. Em seguida será aplicada uma metaheurística com o objetivo de refinar a solução gerada na primeira fase.

Seja S o conjunto de todas as soluções para o problema de coloração de grafos e s um dos elementos de S, ou seja, $s \in S$. A representação computaciconal de s foi feita utilizando um vetor de tamanho $|V|$, sendo que cada posição i do vetor representa a classe de cor c do vértice i. Para explorar o espaço de busca de soluções, utilizamos uma estrutura de vizinhança que aplica a cada solução s um movimento de troca, ou seja, uma vizinhança N de s é definida por:

$$N(s) = \left\{ \ s' : \quad s' = s \oplus m_t \ \right\}$$

Onde m_t consiste na aplicação do movimento de troca da cor de um vértice.

O algoritmo construtivo cria uma classe de cor c_i, em seguida percorre todos os vértice que estão sem cor e não possuem conflitos com algum vértice que esteja na classe c_i. Caso um vértice seja inserido em c_i ele fica proibido de ser inserido em outra classe de cor e seus vértices adjacentes ficam proibidos de pertencerem a classe c_i. Ao final é gerado uma solução viável.

```
ColoracaoGrafoConstrucao(G)
B = ∅
U = V
i = 1
for each vertex v ∈ V

    // Se o vértice u não foi colorido
    // e não esta na lista de bloqueados
    if v ∩ U ≠ ∅ and v ∩ B = ∅
        // Adiciona u a classe i
        c_i = c_i ∪ u
        B = B ∪ Adjacentes de u

    // Se todos os vértices estão bloqueados
    if V − B = ∅
        // cria uma nova classe
        i = i + 1
        B = ∅

```

Código 1. Pseudocódigo da Herística Construtiva

Foi implementado uma matriz para auxiliar no processo de construção da solução. Cada elemento da matriz representado por a_{ij} onde i representa cada vértice do grafo e j representa cada classe de cor. Cada elemento poderia assumir um dos três valores:

$$a_{ij} = \begin{cases} -1, & \text{se o vértice } i \text{ pode receber a cor } j \\ 0, & \text{se o vértice } i \text{ não pode receber a cor } j \\ 1, & \text{se o vértice } i \text{ recebeu a cor } j \end{cases}$$

Na segunda etapa do processo, o objetivo é realizar o refinamento da solução obtida pelo algoritmo construtivo através do processo de busca local. A busca local proposta inicia reduzindo o número de cores obtidas na solução inicial k para $k-1$, através da seleção aleatoria de um classe de cor e remoção de todos elementos contidos nela. Em seguida, o algoritmo percorre os elementos que estão sem cor buscando gerar uma solução viável utilizando apenas as classes de cores utilizadas. Esse processo é repetido enquanto for possível reduzir o número de classe de cores.

```
ColoracaoGrafoBuscaLocal(G, k)
while k > 0

    // Seleciona uma classe de cor que será removida
    i = Random(k)
    // Remove a cor dos vértices pertencentes a classe i
    U = c_i
    c_i = ∅
    k = k − 1
    for each vertex v ∈ U
        // Percorre a vizinhança para gerar uma solução viável
        s_k = s ⊕ m_t

    if s_k é viável
        s = s_k
    else Retorna

```

Código 2. Pseudocódigo da Busca Local

IV. Análise de Complexidade

O problema de coloração de grafos é um clássico problema da classe NP-difícil [3], e portanto, ainda não se conhece algoritmo para resolver este problema de forma exata em tempo polinomial.

Por isso, este trabalho propôs um algoritmo baseado em técnicas metaherísticas de complexidade polinomial a fim de fornecer soluções que se mostrem de boa qualidade em tempo polinomial.

A fase de contrução possui complexidade igual a $O(|V|^2)$ no pior caso, que aconteceria se cada vértice pertencesse a uma classe de cor diferente, nesse caso, a entrada seria um grafo completo. Enquanto que a fase de busca local, tem no pior caso, complexidade $O(|V|^2 * |E|)$. Como o método executa no máximo $|V|$ vezes e em cada loop realiza a atribuição para no máximo $|V|$ vértices e a seleção de uma cor tem complexidade de no máximo $|E| - 1$ arestas que representam os vizinhos do vértice, temos uma complexidade no pior caso de $O(|V|^2 * |E|)$. Somando a complexidade de cada fase obtemos $O(|V|^2) + O(|V|^2 * |E|)$. Por fim, chegamos a complexidade do algoritmo proposto que é igual a $O(|V|^2 * |E|)$.

V. Experimentos

Tabela I
Resultados do Número de Cores

Instância	Vétices	Arestas	(%) GAP [1]	Número de Cores		
				1ª Fase [2]	2ª Fase [3]	Melhor[4]
DSJC125.1	125	736	40,00	8	7	5
DSJC125.5	125	3891	35,29	26	23	17
DSJC125.9	125	6961	15,91	56	51	44
DSJC250.1	250	3218	50,00	13	12	8
DSJC250.5	250	15668	46,43	43	41	28
DSJC500.1	500	12458	58,33	20	19	12
DSJC500.5	500	62624	45,83	72	70	48
DSJC500.9	500	112437	32,54	175	167	126
DSJC1000.1	1000	49629	50,00	31	30	20
DSJC1000.5	1000	249826	48,19	127	123	83
DSJC1000.9	1000	449449	37,50	321	308	224

Tabela II
Resultados do Tempo de Execução

Instância	Vétices	Arestas	Tempo de Execução (ms)		
			1ª Fase [2]	2ª Fase [3]	Melhor[4]
DSJC125.1	125	736	2	3	21000
DSJC125.5	125	3891	2	10	122000
DSJC125.9	125	6961	4	47	121000
DSJC250.1	250	3218	5	7	21000
DSJC250.5	250	15668	12	63	117000
DSJC500.1	500	12458	17	29	210000
DSJC500.5	500	62624	58	401	388000
DSJC500.9	500	112437	99	2322	433000
DSJC1000.1	1000	49629	78	167	260000
DSJC1000.5	1000	249826	264	2393	8407000
DSJC1000.9	1000	449449	433	18402	3234000

O testes foram realizados utilizando algumas instâncias reais de tamanhos e densidade variados de grafos, disponíveis em *The Second DIMACS Implementation Challenge* [18].

Para o experimento foram selecionadas instâncias de grafos com número de vértices que variam de 125 à 1.000 e número aleatório de arestas com três probabilidades de existir uma aresta entre dois vértices quaisquer, iguais a 0.1, 0.5 e 0.9. Convencionaremos desse ponto em diante que essas probabilidades são as densidades do grafo.

Os resultados mostram que a heurística construtiva consegue seus melhores resultados em grafos com baixa densidade, enquanto que a busca local é mais eficiente em grafos mais densos.

Os tempos de execução são muito baixos o que indicam a simplicidade da implementação. Foi possível perceber que, principalmente a busca local, pode ser melhorada para uma implementação mais sofisticada a fim de alcançar melhores resultados.

[1] GAP é a relação entre o melhor resultado e o resultado encontrado.
[2] Heurística Construtiva.
[3] Busca Local.
[4] Melhor resultado encontrado na literatura [12].

VI. Conclusão

Neste trabalho foi apresentado uma metaheurística para o problema de coloração de grafos. O algoritmo proposto tem a estrutura de duas fases. Na primeira é gerado uma solução inicial utilizando um algoritmo guloso e na segunda fase é realizado uma busca local, que permite percorrer dentro do espaço de soluções inviáveis. Apesar da simplicidade na implementação o algoritmo consegue refinar a solução gulosa reduzindo o valor de k. Entretanto, o algoritmo precisaria de uma busca local mais sofisticada para se tornar um algoritmo competitivo em relação aos resultados encontrados na literatura. O problema de coloração de grafos se mostrou um problema altamente combinatório, dada uma solução com k-cores ela por ser combinada em $\begin{pmatrix} n \\ k \end{pmatrix} * k!$ soluções equivalentes.

Uma proposta de trabalho futuro é a implementação de uma busca local mais sofisticada como por exemplo busca tabu, pois a busca local desenvolvida neste trabalho é muito simples e não dispõe de mecanisco para procurar soluções promossoras no espaço de busca.

Referências

[1] L. Sousa, "O teorema das quatro cores," in *Revista do ISPV*. Millenium, 2001.

[2] G. Chartrand and P. Zhang, *Chromatic Graph Theory*, 1st ed. Chapman and Hall CRC, 2008.

[3] M. R. Gary and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1st ed. New York, NY, USA: W. H. Freeman, 1979.

[4] F. T. Leighton, "A graph coloring algorithm for large scheduling problems," *Journal of Research of the National Bureau of Standards*, vol. 84, no. 6, pp. 489–503, 1979.

[5] D. D. Werra, "An introduction to timetabling," *European Journal of Operational Research*, vol. 19, no. 2, pp. 151–162, 1985.

[6] F. Chow and J. Hennessy, "The priority-based coloring approach to register allocation," *ACM Transactions on Programming Languages and Systems*, vol. 12, no. 4, pp. 501–536, 1990.

[7] A. Caprara, L. Kroon, M. Monaci, M. Peeters, and P. Toth, "Passenger railway optimization," *Transportation, Handbook in Operations Research and Management Science*, no. 14, pp. 129–187, 2007.

[8] A. Gamst, "Some lower bounds for a class of frequency assignment problems," *IEEE Transactions on Vehicular Technology*, vol. 35, no. 1, pp. 8–14, 1986.

[9] T. Woo, S. Su, and R. Newman Wolfe, "Resource allocation in a dynamically partitionable bus network using a graph coloring algorithm," *IEEE Transactions on Communication*, vol. 39, no. 12, pp. 1794–1801, 1991.

[10] F. Glover and G. Kochenberger, *Handbook of metaheuristics*, 1st ed. Springer, 2003.

[11] M. Chams, A. Hertz, and D. D. Werra, "Some experiments with simulated annealing for coloring graphs," *European Journal of Operational Research*, vol. 32, no. 2, pp. 260–266, 1987.

[12] P. Galinier and A. Hertz, "A survey of local search methods for graph coloring," *Computers & operations research*, vol. 33, no. 9, pp. 2547–2562, 2006.

[13] A. Hertz and D. D. Werra, "Using tabu search techniques for graph coloring," *Computing*, vol. 39, no. 4, pp. 345–351, 1987.

[14] M. Laguna and R. Martí, "A grasp for coloring sparse graphs," *Computational optimization and applications*, vol. 19, no. 2, pp. 165–178, 2001.

[15] L. Paquete and T. Stützle, "An experimental investigation of iterated local search for coloring graphs," *Applications of Evolutionary Computing*, pp. 191–208, 2002.

[16] P. Galinier and J. Hao, "Hybrid evolutionary algorithms for graph coloring," *Journal of Combinatorial Optimization*, vol. 3, no. 4, pp. 379–397, 1999.

[17] D. Brélaz, "New methods to color the vertices of a graph," *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[18] D. S. Johnson and M. A. Trick, "Np hard problems: Maximum clique, graph coloring, and satisfiability," AMS, Providence, RI, 1996, http://algs4.cs.princeton.edu/66search/, visitado em 21/05/2011.