Apresentado e publicado nos Anais do XXXVII SBPO 2005 – GRAMADO, pg. 2497-2504

METAHEURISTICA BUSCA TABU PARA O PROBLEMA DE COLORAÇÃO DE GRAFOS

Felipe Neves de Aguiar**
Gustavo de Sá Carvalho Honorato**
Haroldo Gambini Santos*
Luiz Satoru Ochi*

Instituto de Computação – Universidade Federal Fluminense
Rua Passo da Pátria, 156, Bloco E, Niterói, RJ, 24210-240
e-mail: {faguiar, ghonorato, satoru, hsantos}@ic.uff.br

Resumo

Este artigo apresenta uma nova heurística Busca Tabu para a solução do Problema de Coloração Vértices de Grafos. Resultados computacionais empíricos com instâncias da literatura indicam que a heurística proposta é muito eficiente apresentando um desempenho médio tão bom quanto as melhores heurísticas da literatura e freqüentemente apresentando um desempenho superior em termos da qualidade da solução gerada.

Abstract

This paper presents a new Tabu Search heuristic to solve the Graph Coloring Problem. Empirical results on instances from the literature suggest that the proposed heuristic is very efficient algorithm, performing at least as well as other heuristics from the literature, and often is better in terms of solution quality.

1. Introdução

Considere um grafo não-orientado $G = (V, E)$, onde V representa o conjunto de nós e E o conjunto de arestas. Considere ainda um conjunto finito de cores C tal que $|C| = |V|$ e uma função $c:V \rightarrow C$ que associa cada nó a uma cor de C. O problema de coloração de vértices de grafos consiste em determinar c tal que se uma aresta $(v, w) \in E$ então $c(v) \neq c(w)$, ou seja, a mesma cor não pode estar associada a dois vértices adjacentes. Neste caso dizemos que c é uma *coloração de G* e se $|c(V)| = k$ dizemos que c é uma k-coloração de G. O problema de coloração de grafos se divide em duas versões. Uma é determinar se existe uma k-coloração de G (problema de decisão) e outra é determinar um número mínimo k tal que exista uma k-coloração de G (problema de otimização). Essa última pode ser vista como a aplicação sucessiva da primeira, sendo que em cada aplicação o número k é reduzido até não ser mais encontrada uma solução que satisfaça as restrições do problema.

Note que se G é k-partido então existe uma k-coloração de G. Neste caso é sabido que tal número k é mínimo e esse número é chamado de número cromático de G, denotado por $\chi(G)$. Assim alguns casos especiais devem ser considerados. Se o grafo é completo, ou seja, todos os nós são adjacentes, $\chi(G) = |V|$. Se o grafo é planar $\chi(G) = 4$. Essa última situação pode representar o problema de coloração de mapas de países, onde cada país tem uma determinada cor e países vizinhos nunca têm cores iguais. A coloração de grafos pode representar alguns problemas da vida real, como os seguintes exemplos:

- *Problemas de Programação de Horários (Timetabling Problems) onde por exemplo se deseja a determinação do número mínimo de dias para se realizar um período de provas em determinada instituição de ensino, onde duas provas não podem ser realizadas na mesma hora se é o mesmo professor que irá aplicá-las, ou se será aplicada na mesma turma, ou se será aplicada na mesma sala.*
- *Alocação de faixas de freqüência para rádios FM, onde dois transmissores não podem transmitir na mesma faixa de freqüência se estão muito pertos um do outro.*
- *Alocação de registradores num processador, onde dado um conjunto de variáveis e um conjunto de registradores disponíveis, deve-se alocar as variáveis de um determinado escopo do programa aos registradores de modo a reduzir o número de acessos à memória.*

Este problema de compreensão fácil é, no entanto, de difícil resolução por métodos de busca

---

* Orientadores ; **: bolsistas de Iniciação Científica do CNPq.

por enumeração. Por ser o problema de coloração de grafos um problema *NP*-difícil [6], tais métodos requerem um tempo de processamento exponencial, sendo que somente para instâncias pequenas o tempo de processamento é viável. Por isso, vários algoritmos aproximativos ou heurísticos têm sido desenvolvidos para se produzir soluções próximas da ótima, num intervalo de tempo razoável. Neste artigo propomos uma nova heurística para o Problema de Coloração de Grafos utilizando conceitos da metaheuristica Busca Tabu [13].

2. Algoritmos heurísticos e metaheurísticos para Coloração de Grafos

Atualmente, podemos encontrar na literatura muitas heurísticas e metaheurísticas desenvolvidas para o problema de coloração de grafos. No caso das heurísticas, elas geralmente pertencem a uma dessas duas classes: *heurísticas construtivas* e *heurísticas de busca local*.

As heurísticas construtivas partem de uma solução inicial vazia (nenhum vértice colorido) e seqüencialmente atribuem cores aos nós, mantendo do satisfatibilidade do problema, até que todos os nós do grafo estejam coloridos. Algumas das técnicas mais conhecidas são a heurística de Brelaz [12] e *Recursive Largest First* (**RLF**) proposta por Leighton [8]. Johnson et al. [11] propôs uma versão randomizada do RLF, chamada **XRLF**, que melhora a performance desta heurística simples.

Em relação a métodos de refinamento de uma solução, a maioria das heurísticas de busca local encontradas na literatura parte de uma solução inicial com conflitos de cores (solução inviável) e iterativamente tentam reduzir o número de conflitos até que este seja zero, ou até que um número máximo de iterações seja alcançado. Elas são guiadas por uma função objetivo que conta o número de conflitos de um determinado esquema de cores. Sendo $V_1, \ldots, V_k$, partições do conjunto V, onde k é o número de cores e cada conjunto V_i contém os nós associados a cor i, define-se $E(V_i) \subseteq E$ como o conjunto de arestas cujos nós terminais pertencem ao conjunto V_i.

$$E(V_i) = \{\ (x,\ y)\ \in\ E \mid x \in\ V_i,\ y\ \in\ V_i\}$$

Assim o número de conflitos existentes num determinado esquema de cores é determinado pela fórmula:

$$f(s) = \sum_{i=1}^{n} |E(V_i)|$$

Quando $f(s) = 0$, concluímos que uma solução viável foi encontrada. Essas heurísticas de busca local são geralmente usadas da seguinte maneira: Após uma coloração inicial viável for produzida por um algoritmo. Então o procedimento tenta reduzir o número de cores uma a uma. A cada cor removida, se a remoção resultar numa solução com conflitos (*inviável*), é aplicada a busca local para encontrar um esquema de cores livre de conflitos. O processo termina quando a busca local não for capaz de produzir um esquema de cores válido. Neste artigo foi utilizada essa abordagem substituindo a fase de busca local tradicional por uma nova heurística utilizando conceitos de busca tabu [13].

Metaheurísticas como GRASP, *Simulated Annealing*, Algoritmo Genético e Busca Tabu também têm sido propostas para este problema. Chams, Hertz e Werra [3] apresentaram resultados da aplicação de *Simulated Annealing* para coloração de grafos. Duas abordagens são apresentadas, uma com *Simulated Annealing* puro e outra combinando XRLF com *Simulated Annealing*. XRLF é usado para construir parcialmente uma solução, colorindo o grafo apenas com um determinado número de nós, então o *Simulated Annealing* é aplicado para colorir o restante do grafo. Hertz e de Werra [9] foram os primeiros a propor uma Busca Tabu para este problema.

Em 2002, Gonzáles -Velarde e Laguna [1] apresentaram uma Busca Tabu com o mecanismo de *vizinhança composta* (*Ejection Chain*) para coloração de grafos. Enquanto numa estrutura de *vizinhança mais simples*, onde uma solução vizinha é obtida apenas mudando a cor de um nó, o mecanismo de *vizinhança composta* analisa uma estrutura de vizinhança onde uma composição de até três movimentos consecutivos é avaliada. Nessa estrutura, a mudança de um nó i_1 com a cor j_0 para a cor j_1 implica na mudança de um nó adjacente i_2 da cor j_1 para a cor j_2 (que pode ser igual ou diferente de j_0) e transitivamente implicando na mudança de um nó i_3, adjacente a i_2, com a cor j_2 para a cor j_3. O procedimento mantém uma lista com as *cores preferidas de cada nó*. A cor preferida de cada nó é

definida como aquela cor que mais reduzirá o número de conflitos de uma solução, caso ela seja atribuída ao nó. Com isso a quantidade de movimentos analisados pela estrutura de vizinhança composta é bastante reduzida, uma vez que as atribuições de cores analisadas somente envolvem as cores preferidas. Chiarandini e Stützle [10] apresentaram uma *Busca Local Iterativa* que alterna as fases de *busca local* e *perturbação* repetitivamente até que um critério de parada seja satisfeito. Para acelerar a avaliação da vizinhança de soluções, é usada uma matriz de dimensões *n.k* que diz, para cada nó quanto será a variação no número de conflitos, caso cada uma das *k* cores lhe seja atribuída.

Neste artigo, são propostas inicialmente três heurísticas de refinamento denominadas respectivamente por*: Busca Local Descendente, Randômico Não-Ascendente* e *Busca Local Não-Ascendente* adaptadas ao problema de coloração de grafos. Em seguida, é proposta uma nova heurística Busca Tabu utilizando conceitos das três heurísticas anteriores.

3. Heurísticas de Refinamento

Inicialmente são propostas três heurísticas de refinamento usando diferentes formas de movimento. Para todas as heurísticas, dada uma solução *s*, é definida uma estrutura de vizinhança *N*(*s*) na qual uma solução vizinha de *s* é obtida escolhendo-se um nó e atribuindo-lhe uma cor diferente da atual.

3.1 Busca Local Descendente (BLD)

A *Busca Local Descendente* (BLD) é uma heurística que analisa toda a vizinhança de soluções, escolhendo como nova solução somente uma solução na qual o número de conflitos seja menor do que na atual. O procedimento termina no primeiro ótimo local encontrado, ou seja, quando não há mais movimentos que reduzam o número de conflitos de cores ou então quando todos os conflitos forem removidos.

```
BLD(G, s)
início
.   V ← {s' ∈ N(s)|f(s') < f(s)};
.   enquanto |V| > 0 faça
.       s ← s' ∈ V tal que f(s') = min{ f(s')|s' ∈ V };
.       V ← {s' ∈ N(s)|f(s') < f(s)};
.   fim enquanto;
fim BLD;
```

Figura1: Pseudocódigo da Busca Local Descendente

3.2 Método Randômico Não-Ascendente (MRNA)

Este método consiste em escolher aleatoriamente uma solução vizinha à solução corrente, aceitando este vizinho somente se o número de conflitos não for aumentado. O procedimento termina quando um determinado número de iterações sem melhora, passado como parâmetro, é alcançado.

```
RNA(G, s, maxit)
início
.   contador ← 0;
.   enquanto contador < maxit faça
.       s' ← rand{s' ∈ N(s)|f(s') ⩽ f(s)};
.       se f(s') < f(s) então
.           contador ← 0;
.       senão
.           contador ← contador + 1;
.       s ← s';
.   fim enquanto;
fim RNA;
```

Figura 2: Pseudocódigo do Método Randômico Não-Ascendente

3.3 Busca Local Não-Ascendente (BLNA)

Este método é uma variação da Busca Local Descendente (BLD), na qual também são aceitos movimentos laterais [15]. Assim como o Método Randômico Não-Ascendente, este método recebe como parâmetro para critério de parada, um número máximo de iterações sem melhora.

4. Busca Tabu

A metaheuristica Busca Tabu é composto de um conjunto de conceitos e práticas que são usadas para resolução de problemas de otimização combinatória e foi proposta independentemente por Glover [13] e Hansen[14][1]. Estes conceitos deixam claro o uso de memória, primordialmente para evitar movimentos cíclicos. O procedimento deve manter estruturas de memória com informações sobre o histórico da busca. O tipo de estrutura mais utilizado é uma *lista tabu* que contém os movimentos realizados mais recentemente, para que estes não sejam imediatamente desfeitos caminhando a busca para soluções já visitadas, dando assim origem ao nome do procedimento.

Porém ao proibir movimentos para tentar evitar o retorno a uma solução já visitada, pode acontecer também de esse mesmo movimento resultar numa solução nunca antes visitada, necessitando nestes casos, a ativação de um módulo que *fiscalize* o papel das listas tabu. Este módulo é conhecido como *critério de aspiração* [13]. Um outro tipo de estrutura de memória, a qual é baseada na freqüência dos movimentos, também pode ser utilizada. Este tipo de memória pode ser utilizado, por exemplo, para proibir movimentos que tenham sido muito freqüentes no histórico da busca, dando assim, um caráter diversificado [13].

4.1. Busca Tabu Proposta

O procedimento proposto para o problema de coloração de grafos, se inicia por um algoritmo construtivo polinomial (figura 3) para gerar uma solução inicial livre de conflitos. Em seguida o número atual de cores é diminuído em um escolhendo-se uma cor e então reassociando os nós com essa cor a uma outra cor disponível. O procedimento analisa qual a cor a ser retirada preferenciando aquela com menor ocorrência no esquema. Ao reassociar os nós, é escolhido a cor que irá gerar o menor número de conflitos. Cada redução no número de cores pode resultar numa solução com conflitos, então aplicamos a Busca Tabu para tentar remover esses conflitos seguindo a função objetivo mencionada acima e usando a mesma estrutura de vizinhança descrita para as heurísticas de refinamento. Antes de aplicarmos a Busca Tabu, guardamos a solução em seu estado livre de conflitos, pois se a busca não conseguir remover os conflitos, é retornada a última solução armazenada.

```
ConstruçãoInicial(G, s)
início
.   para i = 0 até n atribui ao nó i a cor −1;
.   para i = 0 até n faça
.       para j = 0 até n faça
.           podeatribuir ← verdadeiro;
.           para l = 0 até n faça
.               se l é adjacente a i e a cor de l é j então
.                   podeatribuir ← falso;
.                   sai do para;
.               fim se
.           fim para
.           se podeatribuir então atribui ao nó i a cor j;
.       fim para;
.   fim para;
fim ConstruçãoInicial;
```

Figura 3: Algoritmo Construtivo

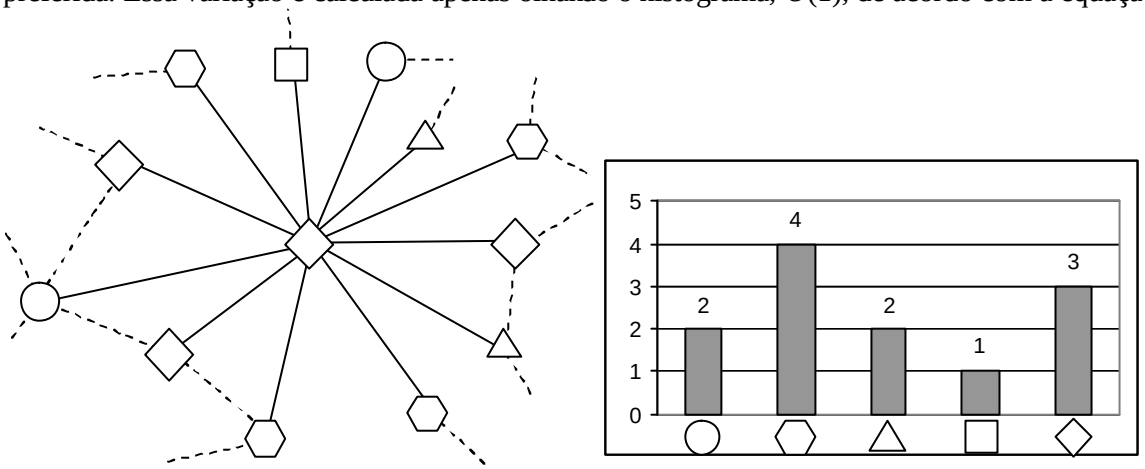
O procedimento descrito pela figura 5 mantém duas estruturas de memória de curto prazo contendo duas regras de ativação de status tabu, como em [1]: De acordo com a primeira regra, quando um nó i muda da cor j_0 para a cor j_1, o nó i fica proibido de receber a cor j_0 por um determinado número de iterações t_1. De acordo com a segunda regra, para esse mesmo movimento, o

[1]Neste artigo nós mostramos apenas alguns conceitos envolvidos na Busca Tabu. Maiores detalhes podem ser encontrados nestas referências.

nó i fica proibido de receber qualquer cor diferente da atual por um número de iterações t_2. Evidentemente, a segunda regra é mais restritiva do que a primeira, por isso os valores para t_1 e t_2 são diferentes. Nos testes os valores para t_1 variaram entre 2,5k e 3,5k e para t_2, 1,25k e 2,25k.

A vizinhança de uma determinada solução s pode ser particionada em n subconjuntos, onde em cada subconjunto V_i, todas as soluções são obtidas alterando-se a cor do nó i. Então em cada um desses subconjuntos é escolhido o melhor vizinho. Para guardar essa escolha, é mantida uma lista com a cor preferida de cada nó [1] $C = (c_1, c_2, .. , c_n)$, onde c_i é a cor preferida do nó i. A lista de cores preferidas é atualizada a cada iteração. A cor preferida de cada nó é aquela que vai resultar no menor número de conflitos, caso tal cor seja atribuída ao nó.

Para a escolha da cor preferida, é gerado para cada nó, um *histograma de cores* dos nós adjacentes. Assim a cor que gerará menos conflitos será aquela com menor ocorrência no histograma. Um exemplo do uso do histograma pode ser visto na figura 4. Essa abordagem foi utilizada para reduzir a complexidade de processamento de escolha do melhor vizinho, pois o normal seria varrer todas as cores e todos os nós e analisar, em cada combinação, qual seria a melhor escolha. Tal abordagem envolveria uma complexidade de pior caso $O(n.k)$ para pesquisar toda a vizinhança, e em cada vizinho gerado, avaliar a qualidade da solução obtida, $O(n^2k)$, num total de $O(n^3k^2)$. Entretanto para calcular a lista de cores preferidas e então, o melhor movimento, a complexidade é de apenas $O((n+k)n)$ (o procedimento precisa gerar o histograma e em seguida, escolher a cor com menor ocorrência) e para se avaliar a qualidade da solução obtida pelo movimento, é consultado um vetor $\Delta F(s)$ que diz, o quanto será a variação no número de conflitos, caso se atribua, a cada nó a sua cor preferida. Essa variação é calculada apenas olhando o histograma, $O(1)$, de acordo com a equação



(a): classes de cores. (b): Histograma de cores dos nós adjacentes.
Classe de cor preferida: □

Figura 4: Exemplo de uso do Histograma.

$\Delta F_i(s) = hist_i[c_i] - hist_i[j_0]$, onde j_0 é a cor atual do nó i. De posse destes valores, computa-se uma lista contendo os nós cujos ΔF 's são iguais ao menor ΔF. A cada iteração, o procedimento precisa atualizar o vetor $\Delta F(s)$ somente nas posições que correspondem aos nós envolvidos no movimento, ou seja, o nó que teve sua cor mudada e os nós adjacentes a ele. Esses nós são mantidos numa lista de nós a pesquisar, chamada *ListadeNosaPesq*. A atualização do vetor $\Delta F(s)$ será tão rápida quanto mais esparso for o grafo.

Note que ao invés de se avaliar a qualidade de uma solução vizinha pela função objetivo, $f(s)$, é calculado somente o quanto será a variação na qualidade da solução para cada movimento (no caso, somente as atribuições das cores preferidas). Como esses valores são guardados, eles podem ser utilizados na próxima iteração.

```
BuscaTabu(G, s, t1, t2, maxit)
inicio
.   F ← f(s);
.   F* ← F;
.   ListaNosaPesq ← { x | x ∈ V};
.   it ← 0;
.   enquanto (ItSM < MaxIt e F > 0) faça
.       ListaMov ← {};
.       para no ∈ ListaNosaPesq faça
.           Hist_no ← geraHistogramaAdj(no);
.           c_no ← calculaCorPref(no, Hist_no, T1);
.           ΔF_no(s) ← Hist_no[c_no] − Hist_no[s(no)];
.       fim para;
.       ListaNosaPesq ← {};
.       ListaMov ← { x ∈ V | ΔF_x(s) = min{ΔF_V(s) }, (x, i) ∉ T2, i ∈ Z};
.       mno ← rand(ListaMov);
.       T1 ← T1 ⋃ { (mno, s(mno), it + rand[t1m, (t1 + 1)m]) };
.       T2 ← T2 ⋃ { (mno, it + rand[t2m, (t2 + 1)m]) };
.       F ← F + ΔF_mno(s);
.       atribuiCor(s, mno, c_mno);
.       ListaNosaPesq ← ListaNosaPesq ⋃ {mno} ⋃ {x | (mno, x) ∈ E};
.       T1 ← T1 − {(n, c, i) ∈ T1|i < it};
.       T2 ← T2 − {(n, i) ∈ T2|i < it};
.       it ← it + 1;
.       se F < F* então
.           F* ← F;
.           ItSM ← 0;
.       senão
.           ItSM ← ItSM + 1;
.       fim se;
.   fim enquanto;
fim BuscaTabu;
```

Figura 5: Pseudocódigo da Busca Tabu

5. Resultados Computacionais

Para avaliar os algoritmos aqui propostos (algoritmos de refinamento e Busca Tabu), foram utilizadas as mesmas instâncias utilizadas por González-Velarde e Laguna [1]. Elas podem ser encontradas em [5], e são divididas em quatro famílias:

- *LEI - grafos de Leighton.*
- *MYC - Grafos baseados na transformação de Mycielski.*
- *REG - Grafos baseados no problema de alocação de registradores em códigos reais.*
- *SGB - É subdividido em quatro grupos: grafos de livros, grafos de jogos, grafos de milhas e grafos de rainhas.*

As instâncias da família LEI são geradas por um procedimento proposto por Leighton [8], que gera grafos de números cromáticos conhecidos. Foram utilizadas 12 instâncias desse tipo. O conjunto MYC é composto por 5 instâncias, com ótimos conhecidos. Todas as instâncias foram utilizadas no teste. O conjunto REG contém instâncias que correspondem ao problema de alocação de registradores em códigos reais. Existem 14 instâncias e todas elas foram utilizadas.
Todas as instâncias usadas têm ótimos conhecidos. Para efeito de comparação com os outros trabalhos, no grupo das intâncias SGB os grafos de rainhas não foram utilizadas nos testes, pois nem todas têm ótimos conhecidos. Também não foi utilizada a instância *homer.col* pois nos outros trabalhos alega-se que houve um problema na leitura do arquivo. Nesta secção apresentamos os testes comparativos dos procedimentos propostos: Busca Local Descendente (**BLD**), Randômico Não-Ascendente **(RNA**), Busca Local Não-Ascendente (**BLNA**) (*as três usando o algoritmo construtivo da figura 3, inicialmente*) e a Busca Tabu Proposta (**BTP**) com os trabalhos apresentados na literatura, *Simulated Annealing* (**SA**) [3], **GRASP** [2], Algoritmo Genético (**AG**) [4] e Busca Tabu com *Ejection Chain* (**TS2**) [1]. Os testes foram feitos num PC Pentium 4 2,8GHz. Os tempos dos outros trabalhos foram ajustados segundo a medida bogoMIPS [16]. Optamos por essa medida, por ela apresentar os valores para todos os processadores.

A tabela 1 mostra um resumo das instâncias, apresentando número total de instâncias em cada grupo (**Num**), número médio de nós (**n**), número médio de arestas **(m)** e o valor médio do valor ótimo (média dos valores ótimos das instâncias consideradas no grupo), denotado por **OPT**. Esta forma de avaliação é usada por outros autores; portanto para efeito de comparação este esquema foi também usada no nosso trabalho. Também aparecem as médias de cores obtidas em cada método em comparação com os valores médios ótimos. Note que o BLNA e o RNA, ao contrário do BLD conseguiram resultados satisfatórios. Nesta tabela, os valores em negrito representam as melhores médias. Repare que entre o BLD e BLNA a única diferença é que o BLNA aceita movimentos laterais. No entanto o BLNA obteve um resultado muito melhor do que o BLD. Isso significa que os movimentos laterais foram de grande importância na busca.

Nome	Num	n	m	OPT	BLD	RNA	BLNA	SA	GRASP	TS2	AG	BTP
LEI	12	450,0	11005,5	**15,0**	24,3	17,2	17,4	18,0	16,5	18,7	***15,8***	***15,8***
MYC	5	73,4	688,4	**6,0**	***6,0***	***6,0***	***6,0***	***6,0***	***6,0***	***6,0***	***6,0***	***6,0***
REG	14	362,1	7679,6	**37,4**	***37,4***	***37,4***	***37,4***	40,1	***37,4***	***37,4***	54,5	***37,4***
SGB	10	113,9	1417,6	**22,6**	23,8	***22,6***	***22,6***	26,0	22,7	22,7	23,1	***22,6***
Total/média	41	249,9	5197,8	**20,2**	22,8	20,8	20,9	22,5	20,6	21,2	24,9	20,5

Tabela 1: Resumo do desempenho médio das instâncias com as médias de cores usadas

Ainda pelos resultados da tabela 1, observamos que o RNA e o BLNA obtiveram médias competitivas em relação aos outros métodos sendo, inclusive, melhores que o *Simulated Annealing*, o TS2 e o AG. A Busca Tabu proposta (**BTP**) obteve uma *média melhor ou igual* aos outros métodos em todas as famílias de instâncias. Um dos fatores de sucesso dos algoritmos aqui propostos: RNA, BLNA e da Busca Tabu (BTP) se deve principalmente à construção inicial (a mesma para todas as heurísticas aqui propostas) que já gera uma solução razoável e pelo alto número de iterações sem melhora tolerado nos três métodos, 100000.

As tabelas 2 e 3 mostram o número de instâncias para as quais a média de cores obtidas foi ótima em cada método, e os respectivos tempos de execução. Até mesmo o BLNA obteve uma boa quantidade de soluções ótimas, empatando com o TS2. É possível, também, perceber a performance inferior do *Simulated Annealing* e do AG ficando apenas com 16 e 18 soluções ótimas respectivamente, e tendo usado um alto tempo de execução. O GRASP foi o que obteve mais soluções ótimas, 34, seguido da Busca Tabu proposta (BTP), 33, do TS2 e do BLNA, 31, e do RNA, 29. Infelizmente os tempos utilizados pela Busca Tabu proposta foram altos comprados com os outros métodos, o que se deve ao número de iterações mencionado anteriormente. Note que o TS2 e os algoritmos propostos neste artigo usam um tempo de execução proporcionalmente pequeno para a família REG. Isso acontece porque para essa família, o construtor inicial já gera a solução ótima para todas as instâncias, então o procedimento termina imediatamente. Esse é o mesmo motivo para qual o BLD consegue chegar à solução ótima para 22 das instâncias. A verdade é que esse método praticamente não consegue melhorar uma solução que já é razoável, como a da construção inicial. Isso explica, também, o pouco tempo empregado por esse método.

Nome	BLD	RNA	BLNA	SA	GRASP	TS2	AG	BTP
LEI	0	1	2	2	**6**	3	**6**	4
MYC	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
REG	**14**	**14**	**14**	3	**14**	**14**	0	**14**
SGB	3	9	**10**	6	9	9	7	**10**
Total	22	29	31	16	**34**	31	18	33

Tabela 2: Número de soluções ótimas obtidas

Nome	BLD	RNA	BLNA	SA	GRASP	TS2	GA	BT
LEI	0,01	258,91	17,43	20,34	3,94	5,18	19,73	31,39
MYC	0,00	0,00	0,00	0,02	0,11	0,00	0,98	0,00
REG	0,00	0,00	0,00	2,99	2,38	0,13	14,47	0,00
SGB	0,00	5,04	2,27	0,03	0,29	0,11	1,03	0,02
Média	0,00	65,99	4,93	5,84	1,68	1,36	9,05	7,85

Tabela 3: Tempos de execução em segundos

6. Conclusões

Este artigo apresenta como contribuições, três procedimentos de busca local (BLD, RNA e BLNA) e uma nova heurística Busca Tabu (BTP) para o problema de coloração de grafos. Os resultados computacionais ilustrados nas tabelas 1, 2, e 3 mostram que a Busca Tabu proposta neste artigo se mostra bastante competitiva em relação às outras metaheurísticas em relação a qualidade das soluções obtidas superando na média os resultados da melhor metaheuristica da literatura (GRASP) que detinha até então os melhores resultados. Os resultados promissores da BTP em parte se devem as buscas locais usadas e também pela prioridade dada no sentido de reduzir o custo de uma iteração da Busca Tabu, fazendo uso eficiente da memória. Isso também permitiu que um alto número de iterações fosse empregado no método.

7. Referências

[1] J. L. Gonzáles-Velarde e M. Laguna, Tabu Search with Simple Ejection Chains for Coloring Graphs, *Annals of Operations Research*, 117, 165-174, 2002

[2] M. Laguna e R. Martí, A GRASP for coloring sparse graphs, *Computational Optimization and Applications*, 19(2), 165-178, 2001.

[3] M. Chams, A. Hertz e D. de Werra, Some experiments with simulated annealing for coloring graphs, *European Journal of Operation Research* 32, 260-266, 1987.

[4] C. Fleurent e J.A. Ferland, Genetic and Hybrid Algorithms for Graph Coloring, *Annals of Operation Research*, 63, 437-464, 1996.

[5] M. Trick, "Graph Coloring Instances", disponível em julho 2005, em http://mat.gsia.cmu.edu/COLOR/instances.html

[6] FM. R. Garey e D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co., San Francisco, 1979

[7] C. R. Reeves, editor. *Modern heuristic techniques for combinatorial problems*. Blackwell, Oxford, 1993.

[8] F.T. Leighton, A graph coloring algorithm for large scheduling problems, *J. Res. Nat. Bur. Standards*, 84, 489-506, 1979.

[9] A. Hertz e D. de Werra, Using Tabu Search Techniques for Graph Coloring, *Computing* **39**, (1988) 345-351.

[10] M. Chiarandini e T. Stützle, An application of Iterated Local Search to Graph Coloring, *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations (D.S. Johnson, A. Mehrotra e M. Trick, editores)*, 112-125, 2002.

[11] D. S. Johnson, C. A. Aragon, L. A. Mcgeoch e C. Schevon, Optimization by Simulated Annealing: An Experimental Evaluation – Part II (Graph Coloring and Number Partitioning), *Operations Research*, 31, 378-406, 1991.

[12] D. Brelaz, New methods to color the vertices of a graph. *Communications of the ACM*, 22(4), 251–256, 1979.

[13] F. Glover e M. Laguna, *Tabu Search*. Kluwer, Boston, 1997.

[14] P. Hansen, The steepest ascent mildest descent heuristic for combinatorial programming, Congress on Numerical Methods in Combinatorial Optimization, Capri, 1986.

[15] B. Selman, H. Levesque, e D. Mitchell, A new method for solving hard satisfability problems, *Proceedings of the 10th National Conference on Artificial Intelligence*, 1992.

[16] W. van Dorst, "BogoMips mini-Howto", disponível em agosto 2005, em http://www.tldp.org/HOWTO/BogoMips/