RT-ColorAnt: Um Algoritmo Heurístico Baseado em Colônia de Formigas Artificiais com Busca Local para Colorir Grafos

Carla Négri Lintzmayer, Mauro Henrique Mulati, Anderson Faustino da Silva
Departamento de Informática – Universidade Estadual de Maringá (UEM)
Avenida Colombo, 5790 – 87020-900 – Bloco C56 – PR – Brazil
{carla0negri, mhmulati}@gmail.com, anderson@din.uem.br

RESUMO

O problema de coloração de grafos é $\mathcal{NP}$-difícil e é usado em diversas aplicações práticas, como escalonamento de tarefas e alocação de registradores. Estas características fazem surgir interesses no desenvolvimento de algoritmos heurísticos para tentar resolver este problema. Este artigo apresenta o algoritmo *RT-ColorAnt*, um algoritmo heurístico baseado em colônia de formigas artificiais com busca local projetado para o problema de coloração de grafos. Os experimentos realizados demonstraram que o algoritmo proposto é capaz de obter soluções satisfatórias, além de ser uma boa opção quando o tempo de execução é um fator crítico, e não necessariamente a qualidade da solução.

PALAVRAS CHAVE. Grafo, Problema de Coloração de Grafos, Otimização por Colônia de Formigas Artificiais

ÁREA PRINCIPAL. Meta-Heurísticas.

ABSTRACT

The graph coloring problem is $\mathcal{NP}$-hard and it is used in many practical applications, such as task scheduling and register allocation. Such properties raise interests in developing heuristic methods to try to solve this problem. This article presents the RT-ColorAnt algorithm, an heuristic algorithm based on artificial ant's colony with local search designed for the graph coloring problem. The experiments demonstrated that the proposed algorithm is able to obtain satisfactory solutions, besides being a good option when the execution time is a critical factor, and not necessarily the solution quality.

KEYWORDS. Graph, Graph Coloring Problem, Ant Colony Optimization.

MAIN AREA. Metaheuristics.

1. Introdução

Segundo Dorigo e Krzysztof (2006), o campo de pesquisa de inteligência coletiva (inteligência por enxame, *swarm intelligence*) atua sobre algoritmos baseados no comportamento de enxames e na sua coletividade. Ainda segundo esses autores, enxames são compostos por indivíduos (abelhas, formigas ou cupins, por exemplo) que realizam o trabalho interagindo entre si coordenadamente sem uma forma de controle centralizado.

Ant System (AS) é um algoritmo heurístico que utiliza colônia de formigas artificiais como ferramenta de otimização e foi proposto por Dorigo *et al.* (1991). Por sua vez, a *Ant Colony Optimization* (ACO) foi formalizada como sendo uma meta-heurística de otimização combinatorial, que, de acordo com Dorigo *et al.* (2006), é um conjunto de conceitos algorítmicos que podem ser aplicados à diversos problemas.

Devido às características presentes nas colônias de formigas artificiais, diversos trabalhos propõem o uso de algoritmos ACO para a resolução de diferentes problemas, tais como os citados por Dorigo e Stützle (2004): roteamento de veículos, atribuição de frequência, agendamento e coloração de grafos. Este último aparece em diversos problemas onde, de acordo com Bondy e Murty (2008), é necessário particionar um conjunto de elementos em vários grupos com determinadas características compatíveis entre os membros.

O interesse no problema de coloração de grafos (PCG) é, de forma simples, encontrar uma quantidade k de cores que possam ser atribuídas aos vértices de forma que não existam vértices adjacentes com a mesma cor. Trivialmente, se um grafo G possui n vértices, então basta escolher $k = n$ cores. Mas o objetivo é encontrar o valor mínimo de k que respeite a restrição do problema, denominado número cromático do grafo e denotado por $\chi(G)$.

Encontrar o número cromático de um grafo é um problema $\mathcal{NP}$-difícil, conforme apresentado por Karp (1972). Assim, a menos que $\mathcal{P} = \mathcal{NP}$, não há algoritmos exatos em tempo polinomial, segundo Cormen *et al.* (2009), que possam resolver grandes instâncias, sendo necessárias técnicas alternativas para obter soluções satisfatorias. Isso tem resultado em trabalhos que exploram algoritmos heurísticos e meta-heurísticas, como os de Plumettaz *et al.* (2010), Galinier e Hertz (2006), Laguna e Martí (2001), Galinier e Hao (1999) e Johnson e Trick (1996).

Este artigo apresenta um algoritmo baseado no comportamento de colônias de formigas artificiais aplicado ao problema da coloração de grafos e os resultados obtidos com sua utilização. Ele encontra-se dividido da seguinte forma: a Seção 2 descreve o referencial teórico desta pesquisa; a Seção 3 apresenta alguns trabalhos relacionados; a Seção 4 descreve o algoritmo proposto; a Seção 5 apresenta alguns resultados preliminares; e por fim, a Seção 6 apresenta as considerações finais e os trabalhos futuros.

2. Referencial Teórico

O presente trabalho se utiliza de pesquisas sobre colônia de formigas artificiais, sobre as definições envolvendo o PCG e explora a aplicação de algoritmos ACO ao PCG.

2.1. Colônia de Formigas Artificiais

Colônias de formigas naturais são organizadas e apresentam comportamento que permite a realização de diversas tarefas que não seriam possíveis por uma única formiga. A comunicação indireta que as coordena e orienta se dá por alterações que elas realizam no ambiente, em um processo denominado "*stigmergy*", como explicado por Dorigo e Stützle (2004). Na maioria dos casos, essa comunicação se dá pelo depósito da substância química *feromônio* na superfície, formando trilhas que guiam o caminho de cada formiga.

Quanto maior a concentração de feromônio em um caminho, maior a probabilidade da formiga escolhê-lo. Tal comportamento é chamado de autocatalítico: um processo que reforça a si mesmo causando convergência, de acordo com Dorigo *et al.* (1996). Como o feromônio evapora com o

tempo e caminhos mais curtos são percorridos mais rapidamente (incentivando as formigas a passarem mais vezes por eles), a concentração de feromônio nestes caminhos tenderá a ser maior. Portanto, em algum momento a colônia tenderá a estar percorrendo o menor caminho possível entre dois pontos, característica que atraiu a atenção para o fato de que o comportamento das formigas poderia ser mapeado e utilizado computacionalmente. A técnica foi bastante explorada e principalmente aplicada ao problema do caixeiro viajante (PCV). Desde então, estudos vêm sendo realizados para mapear o comportamento das formigas para outros problemas de otimização, como o PCG, conforme citam Dorigo e Krzysztof (2006).

ACO é uma meta-heurística de otimização combinatória que se baseia no comportamento das formigas artificiais. Uma meta-heurística "é um conjunto de conceitos algorítmicos que podem ser usados para definir métodos heurísticos aplicáveis a um largo conjunto de diferentes problemas", segundo Dorigo e Stützle (2004). São exemplos de meta-heurísticas: *tabu search* apresentada por Glover (1989), *simulated annealing* por Kirkpatrick *et al.* (1983) e *iterated local search* por Lourenço *et al.* (2003).

A execução de um algoritmo ACO geralmente é composta por vários ciclos, onde atua a colônia. Conforme descrito por Dorigo e Krzysztof (2006), cada formiga é geralmente um método construtivo e seu comportamento no algoritmo é percebido principalmente quando, para decidir para onde ela deve dar o próximo "passo", usa-se uma probabilidade calculada com base em dois fatores: a trilha de feromônio e a informação heurística (desejabilidade ou atratividade) relacionadas ao item. A informação heurística depende de cada problema.

2.2. Problema de Coloração de Grafos

Uma k-coloração de um grafo $G = (V, E)$ é uma atribuição de k cores aos seus vértices, ou seja, um mapeamento $c : V \rightarrow \{1..k\}$. Outra abordagem é o particionamento de V em k conjuntos independentes ou *classes legais* de cores $s = \{C_1, C_2, ..., C_k\}$. A coloração é *própria*, como citado por Bondy e Murty (2008), quando ela não possui nenhuma *aresta conflitante*, ou seja, não existem vértices adjacentes com a mesma cor. Um grafo é *k-colorível* se possuir uma k-coloração própria. O valor mínimo de k que permite a um grafo ser k-colorível é o *número cromático* do grafo e é representado por $\chi(G)$.

O PCG consiste, portanto, em encontrar o menor valor de k tal que o grafo seja k-colorível. Deseja-se minimizar a função objetivo que, para o PCG, é o número de cores da solução.

Shawe-Taylor e Zerovnik (2001) apresentam o problema da k-coloração descrito como um problema de decisão (k-PCG) da seguinte forma: dado um grafo G e um número fixo inteiro k de cores possíveis, G é k-colorível? Para o k-PCG, também deseja-se minimizar a função objetivo, que, neste problema, é o número de arestas conflitantes da solução.

Alguns grafos com características específicas possuem χ conhecido e fixo, como grafos bipartidos (2-coloríveis) e grafos planares (4-coloríveis). Para determinar se um grafo é 2-colorível existem algoritmos em tempo polinomial, conforme Bondy e Murty (2008). Para outros casos não especiais, o PCG necessita de técnicas que o resolvam de maneira satisfatória, já que não se conhecem (e provavelmente não existem) algoritmos exatos que processem grafos com mais de 100 nós, como descreve Plumettaz *et al.* (2010).

Uma aplicação real do k-PCG é vista na alocação de registradores como descrita por Appel (1998). Neste problema, a solução não se restringe apenas a verificar se um grafo é k-colorível, mas também deve utilizar alguma heurística que possibilite "eliminar" as arestas conflitantes da melhor forma possível, já que é obrigatório colorir o grafo apenas com k cores.

2.3. Aplicando Colônia de Formigas a Coloração de Grafos

Diferentes tipos de métodos baseados em colônias de formigas artificais foram desenvolvidos para o PCG. Eles são classificados por Hertz e Zufferey (2010) em três classes:

Classe 1 constituída de algoritmos nos quais cada formiga é um método construtivo que reforça a trilha de feromônio entre pares de vértices não adjacentes quando eles recebem a mesma cor;

Classe 2 composta por algoritmos nos quais as formigas caminham pelo grafo nem sempre previamente colorido e tentam modificar a cor dos vértices de forma a reduzir o número de conflitos existentes; e

Classe 3 aqueles nos quais as formigas são algoritmos de busca local: partindo de um grafo já colorido, cada formiga procura por uma solução vizinha, que normalmente é dada pela atribuição de uma nova cor a um vértice conflitante existente na solução atual.

As duas últimas classes se diferenciam de forma significativa da idéia original de um algoritmo ACO e, ainda segundo Hertz e Zufferey (2010), existem divergências com relação ao fato de poder considerar tais algoritmos como sendo "baseados em colônias de formigas artificiais". Em geral, os algoritmos que simulam a trilha de feromônio o fazem de maneira semelhante: vértices adjacentes não possuem valor na trilha e vértices não adjacentes que recebem a mesma cor de alguma formiga têm seu feromônio reforçado.

3. Trabalhos Relacionados

O primeiro trabalho sobre coloração de grafos com colônia de formigas foi o *ANTCOL*, descrito por Costa e Hertz (1997). Neste trabalho, cada formiga constrói uma coloração para o grafo, utilizando os métodos construtivos *RLF* (*Recursive Large First*), de Brélaz (1979) e *Dsatur*, de Leighton (1979), em busca do menor k possível (não-fixo) para colorir o grafo propriamente. Uma matriz $M_{|V|\times|V|}$ armazena a experiência das construções anteriores (feromônio). O inverso do número de cores encontradas na solução de uma formiga é somado à trilha entre dois vértices, reforçando-a, quando tais vértices são não-adjacentes e coloridos com a mesma cor. Simula-se, também, a evaporação do feromônio. Os resultados do *ANTCOL* não foram os melhores, mas foram bons o suficiente para abrir caminho para novos estudos. Ahn *et al.* (2003) apresentam uma versão melhorada trocando o método *RLF* pelo *XRLF* (*eXtended RLF*).

Comellas e Ozón (1998) descrevem uma abordagem diferente. Em uma iteração cada formiga se move, com certa probabilidade, para um nó adjacente que tenha o maior número de arestas conflitantes. Nesse nó, a formiga substitui, também com certa probabilidade, a cor atual por uma nova que minimize os conflitos. O algoritmo utiliza a experiência dos eventos passados, porém não mantém uma matriz ou lista para tal. Os resultados apresentados apenas comparam o algoritmo com o *ANTCOL*, mostrando-se melhor.

Shawe-Taylor e Zerovnik (2001) apresentam um algoritmo para o k-PCG. Neste, cada formiga é um procedimento iterativo que tenta minimizar o número de conflitos. A trilha de feromônio é modelada com base em um grafo G', que começa igual ao G, ao qual arestas vão sendo adicionadas caso muitas formigas atribuam cores diferentes a nós não-adjacentes. Este método foi avaliado com poucas instâncias e estas são coloridas otimamente por algoritmos exatos em segundos.

Outro trabalho é apresentado por Hertz e Zufferey (2006), no qual cada formiga contribui colorindo um único vértice, de forma que uma colônia inteira realiza apenas uma solução. O problema abordado é o k-PCG. O método sempre atribui uma cor (entre as k) para cada formiga e k formigas para cada vértice. Então é utilizado um procedimento baseado no *Dsatur*, que escolhe um vértice e atribui uma cor que tenha pelo menos uma formiga representante e que minimize os conflitos. As formigas vão mudando de posição baseadas na informação heurística e na trilha de feromônio. Este algoritmo foi comparado com o *Dsatur*, *ANTCOL* original, *Tabucol* modificado de Galinier e Hao (1999) e o algoritmo genético híbrido *GH* de Galinier e Hao (1999), considerado como a melhor heurística de coloração, e superou apenas o *Dsatur* e o *ANTCOL* com *Dsatur*, para grandes instâncias.

O algoritmo mais recente, chamado *ALS-COL* (*Ant Local Search*) e apresentado por

Plumettaz *et al.* (2010), é também o único que consegue competir com os melhores algoritmos de coloração de grafos. Neste algoritmo, cada formiga é uma busca local derivada da Busca *Tabu* para o k-PCG. A busca utilizada modifica colorações parciais, que são os vértices particionados em $k+1$ classes de cores: $C_1, \ldots, C_k$ classes legais e uma classe C_{k+1} de vértices não coloridos. A solução vizinha surge com as ações de mover um vértice $v \in C_{k+1}$ para alguma classe C_c e mover os vizinhos de v que estão em C_c para C_{k+1}. O movimento (v, c) é escolhido da seguinte forma: a formiga escolhe um conjunto de movimentos não-*tabu* com os valores mais altos de informação heurística (no caso, a informação heurística de (v, c) é o inverso do número de vizinhos de v em C_c). Nesse conjunto, a formiga escolhe o movimento com o maior valor de feromônio (que é tratado aproximadamente como no *ANTCOL*). O *ALS-COL* é comparado com o *PartialCol* de Blöchliger e Zufferey (2008), *Tabucol*, *GH*, *MOR* de Morgenstern (1996) e *MMT* de Malaguti *et al.* (2008). É capaz de encontrar o número cromático ou o melhor valor conhecido de várias instâncias testadas, e se aproxima bastante das outras. Ficou pior em algumas delas quando comparado ao *MMT* e ao *GH*, mas ainda assim com pouca diferença.

4. O Algoritmo *RT-ColorAnt*

O algoritmo proposto, chamado *RT-ColorAnt*, utiliza como base o algoritmo *ANTCOL* modificado para colorir um grafo G com k cores fixas, brevemente citado por Costa e Hertz (1997), chamado aqui de k-ANTCOL.

A cada passo, o k-ANTCOL escolhe um vértice v para colorir e uma cor c para colorí-lo na solução s. O vértice será aquele que possui o maior grau de saturação, $gsat(v)$, entre os vértices não coloridos. Grau de saturação de um vértice não colorido é o número de cores diferentes que já foram atribuídas aos seus vértices adjacentes. A cor $c \in \{1..k\}$ é escolhida com probabilidade p (como apresentado na Equação 1), que é calculada com base na trilha de feromônio τ (como apresentado na Equação 2) e no fator de desejabilidade η (como apresentado na Equação 3).

$$p(s, v, c) = \frac{\tau(s, v, c)^{\alpha} \cdot \eta(s, v, c)^{\beta}}{\sum_{i \in \{1..k\}} \tau(s, v, i)^{\alpha} \cdot \eta(s, v, i)^{\beta}} \tag{1}$$

onde α e β são parâmetros passados ao algoritmo e controlam quão importante são os valores associados a eles na equação.

$$\tau(s, v, c) = \begin{cases} 1 & \text{se } C_c(s) = \{\} \\ \dfrac{\sum_{u \in C_c(s)} F_{uv}}{|C_c(s)|} & \text{caso contrário} \end{cases} \tag{2}$$

onde F_{uv} é a trilha de feromônio entre os vértices u e v, explicada a seguir, e $C_c(s)$ é a classe de cor c na solução s, ou seja, o conjunto de vértices já coloridos com a cor c naquela solução.

$$\eta(s, v, c) = \frac{1}{|N_{C_c(s)}(v)|} \tag{3}$$

onde $N_{C_c(s)}(v)$ são os vértices $x \in C_c(s)$ vizinhos de v na solução s, ou seja, os vértices adjacentes a v já coloridos com a cor c naquela solução.

A trilha de feromônio (armazenada na matriz $F_{|V| \times |V|}$) é inicializada com 0 para todos os vértices adjacentes e 1 caso contrário. Sua atualização envolve a persistência por um fator ρ da trilha atual (a evaporação é dada por $1 - \rho$) e o reforço da mesma por meio da experiência obtida nas soluções que as formigas do ciclo geraram. Este trabalho incluiu uma diferença em relação à trilha de feromônio quando comparado ao k-ANTCOL original: além de cada formiga da colônia reforçar

a trilha, a solução da melhor formiga da colônia após a aplicação da busca local (s') e a solução da melhor formiga encontrada durante toda a execução até o momento (s^*) também reforçam a trilha. A evaporação da trilha é dada pela Equação 4 e a forma geral de depósito de feromônio é mostrada na Equação 5.

$$F_{uv} = \rho F_{uv} \quad \forall u, v \in V \tag{4}$$

$$F_{uv} = F_{uv} + \frac{1}{f(s)} \quad \forall u, v \in C_c(s) \mid (u, v) \notin E, c = 1..k \tag{5}$$

onde ρF_{uv} indica a persistência da trilha atual, $C_c(s)$ é o conjunto de vértices coloridos com a mesma cor c na solução s e f é a função objetivo, que retorna o número de arestas conflitantes da solução. O pseudocódigo do k-ANTCOL é descrito no Algoritmo 1.

Algoritmo 1 Pseudocódigo do k-ANTCOL.

K-ANTCOL($G = (V, E)$, k) // V é o conjunto de vertices e E o de arestas
1 $NC = V$; // vértices ainda não coloridos
2 $cor_i = 0 \quad \forall i \in V$; // vetor cor mantém um mapeamento vértice-cor
3 **while** $ncoloridos < |V|$ **do**
4 $v = \arg\max\{gsat(v') \mid v' \in NC\}$;
5 escolher uma cor $c \in \{1..k\}$ com probabilidade $p(s, v, c)$ dada pela Equação 1;
6 $cor_v = c$;
7 $NC = NC \backslash v$;
8 $ncoloridos{+}{+}$;
9 **return** cor;

O *RT-ColorAnt* é um algoritmo que utiliza uma implementação modificada do algoritmo k-ANTCOL como método construtivo, utiliza a busca tabu reativa *React-Tabucol* apresentada por Blöchliger e Zufferey (2008) para melhorar as soluções encontradas, e é descrito no Algoritmo 2.

Algoritmo 2 Pseudocódigo do *RT-ColorAnt*.

RT-COLORANT($G = (V, E)$, k) // V é o conjunto de vertices e E o de arestas
1 $F_{uv} = 1 \quad \forall (u, v) \notin E; \qquad F_{uv} = 0 \quad \forall (u, v) \in E;$
2 $s^*.num\_conflitos = \infty$;
3 **while** $tempo < max\_tempo$ **and** $f(s^*) \neq 0$ **do**
4 $\Delta F_{uv} = 0 \quad \forall u, v \in V$;
5 $s'.num\_conflitos = \infty$;
6 **for** $a = 1$ **to** $nformigas$ **do**
7 $s =$ K-ANTCOL(G, k);
8 $\Delta F_{uv} = \Delta F_{uv} + \frac{1}{f(s)} \qquad \forall u, v \in C_c(s) \mid (u, v) \notin E, c = 1..k$;
9 **if** $f(s) == 0$ **or** $f(s) < f(s')$ **then**
10 $s' = s$;
11 **if** $f(s') \neq 0$ **then**
12 REACT_TABUCOL(s');
13 **if** $f(s') < f(s^*)$ **then**
14 $s^* = s'$;
15 $F_{uv} = \rho F_{uv} + \Delta F_{uv} \quad \forall u, v \in V$;
16 $F_{uv} = F_{uv} + \frac{1}{f(s')} \quad \forall u, v \in C_c(s') \mid (u, v) \notin E, c = 1..k$;
17 $F_{uv} = F_{uv} + \frac{1}{f(s^*)} \quad \forall u, v \in C_c(s^*) \mid (u, v) \notin E, c = 1..k$;

A busca tabu utilizada pelo *RT-ColorAnt* é descrita a seguir. Dada a função objetivo f que retorna o número de arestas conflitantes, um espaço de soluções S onde cada solução é formada por k classes de cores e todos os vértices estão coloridos (contendo conflitos ou não) e dada uma solução inicial $s_0 \in S$, f deve ser minimizada sobre S. Chama-se *movimento* a troca da cor de um único vértice e ele ocorre entre duas soluções *vizinhas*. Quando ocorre, o movimento inverso é armazenado em uma *lista tabu* de forma que nas próximas t (chamado *tabu tenure*) gerações esse movimento não possa ser realizado. O que a busca faz é gerar uma sequência $s_1, s_2, \ldots$ de soluções em S, que é da forma $s_{i+1} \in N(s_i)$, com $s_{i+1} = \arg \min_{s \in N'(s_i)} f(s)$, onde $N(s)$ é o conjunto de soluções vizinhas de s e $N'(s) \subseteq N(s)$ contém apenas os movimentos que não estão presentes na lista tabu, ou seja, a próxima solução da sequência deve ser gerada por um movimento não-tabu e ela deve possuir o menor número de conflitos entre as soluções vizinhas possíveis.

5. Avaliação Experimental

O algoritmo *RT-ColorAnt* foi implementado na linguagem `C`, compilado com GCC 4.1.2 utilizando O3 e executado em um computador Intel Xeon E5620 de 2.40 GHz, 8GB RAM e sistema operacional Rocks.

5.1. Metodologia

Para a calibragem do algoritmo foram realizados experimentos com $\alpha, \beta \in \{1..10\}$, indicando que bons valores para esses parâmetros são $\alpha = 7$ e $\beta = 9$. Além disto, eles indicaram $\rho = 0,7$ e $nformigas = 180$ como valores razoáveis. A busca *React-Tabucol* foi limitada por um número máximo de 500 ciclos. A execução do *RT-ColorAnt* finaliza assim que uma solução própria é encontrada ou quanto o tempo máximo de execução de uma 1 hora é atingido.

Os experimentos foram realizados em 13 grafos do Desafio DIMACS, descrito por Johnson e Trick (1996). Os grafos escolhidos são os mesmos utilizados por Plumettaz *et al.* (2010), para que seja possível uma comparação dos resultados obtidos pelo *RT-ColorAnt* com algumas das heurísticas que tal trabalho comparou. As instâncias consideradas[1], são:

- `dsjc500.1`, `dsjc500.5`, `dsjc500.9`, `dsjc1000.1`, `dsjc1000.5`: grafos aleatórios padrão `dsjc`$n.d$ que possuem n vértices e d probabilidade de quaisquer dois vértices formarem uma aresta;
- `dsjr500.1c` e `dsjr500.5`: grafos aleatórios geométricos `dsjr`$n.d$ que foram gerados escolhendo n pontos em um quadrado e configurando arestas entre pares de vértices com distância menor que d. A letra 'c' ao final do nome do arquivo indica que o grafo é complemento do grafo geométrico;
- `flat300_28_0` e `flat1000_76_0`: grafos `flat`n_χ_0 que possuem n vértices e número cromático χ conhecido;
- `le450_15c`, `le450_15d`, `le450_25c` e `le450_25d`: grafos `le450_`χl que possuem sempre 450 vértices e número cromático χ conhecido.

Essas instâncias foram escolhidas por Plumettaz *et al.* (2010) por, segundo o autor, serem as mais desafiadoras. Algumas dessas instâncias também foram utilizadas por Blöchliger e Zufferey (2008) para avaliar a busca local *React-Tabucol*. Nos testes do k-ANTCOL foram utilizados alguns grafos aleatórios, nos quais as probabilidades variavam entre 4, 5, e 6, e a quantidade de vértices variava entre 100, 300, 500 e 1000. Não foi possível identificar exatamente quais instâncias são essas. Portanto, não será possível comparar os resultados do *RT-ColorAnt* com os de seu algoritmo base. Sabe-se apenas que os grafos `dsjc500.5` e `dsjc1000.5` fazem parte deste conjunto.

[1]Disponível em `http://mat.gsia.cmu.edu/COLOR/instances.html`, acessado em junho de 2011.

5.2. Resultados e Discussão

A Tabela 1 apresenta os resultados obtidos pelo *RT-ColorAnt* com os parâmetros mencionados anteriormente. Nesta tabela, a primeira coluna apresenta o nome do grafo e a segunda apresenta o par χ/k^* (com '?' caso χ não seja conhecido) onde k^* é o valor da melhor aproximação da coloração encontrada até o momento. O *RT-ColorAnt* foi executado 10 vezes para cada valor de k a partir de χ ou de k^*, incrementando k até que 10 execuções fossem bem sucedidas, ou seja, não houvesse arestas conflitantes. A terceira coluna apresenta os valores de k que foram avaliados. A quarta coluna apresenta o par *número de execuções bem sucedidas(Suc.)/total de execuções(Exec.)*. A quinta coluna apresenta o *tempo médio até a melhor solução encontrada*[2] em segundos. Enfim, a última coluna apresenta o número médio de conflitos dos dez experimentos para cada cor. Os resultados foram omitidos quando não houve execuções bem sucedidas, para simplificação da tabela.

Tabela 1: Resultados obtidos pelo *RT-ColorAnt*.

Grafo	χ/k^*	k	Suc./Exec.	Tempo(s)	Conflitos
dsjc500.1.col	?/12	13	10/10	385,043	0
dsjc500.5.col	?/48	53	3/10	2204,815	3,1
		54	10/10	509,998	0
dsjc500.9.col	?/126	132	1/10	1552,924	5,9
		133	2/10	1503,506	3,1
		134	8/10	937,453	0,8
		135	10/10	202,384	0
dsjc1000.1.col	?/20	23	10/10	20,644	0
dsjc1000.5.col	?/83	98	6/10	1080,445	2,2
		99	10/10	600,418	0
dsjr500.1c.col	?/85	**85**	3/10	708,217	1,9
		86	9/10	778,647	0,2
		87	10/10	32,443	0
dsjr500.5.col	?/122	123	3/10	518,887	1,1
		124	3/10	701,903	0,8
		125	10/10	7,564	0
flat300_28_0.col	28/28	33	8/10	893,705	0,6
		34	10/10	29,934	0
flat1000_76_0.col	76/82	97	3/10	1194,426	3,4
		98	10/10	335,324	0
le450_15c.col	15/15	17	1/10	1673,505	52,2
		18	0/10	1315,475	38,3
		19	1/10	1358,056	22,8
		20	0/10	1352,943	9,8
		21	5/10	559,859	2,3
		22	10/10	11,839	0
le450_15d.col	15/15	16	1/10	1800,629	69,3
		17	1/10	2132,706	52,1
		18	3/10	1716,221	26,8
		19	6/10	1754,457	4,7
		20	3/10	1904,039	7,2
		21	7/10	656,061	1,3
		22	10/10	6,611	0
le450_25c.col	25/25	27	7/10	884,212	1,1
		28	10/10	0,879	0
le450_25d.col	25/25	27	7/10	366,113	1
		28	10/10	0,935	0

[2]Independente da solução ser própria (possuir nenhum conflito) ou não.

Os resultados apresentados na Tabela 1 demonstram que o *RT-ColorAnt* possui os melhores resultados para grafos *geométricos*. Neste caso, a distância média entre os resultados obtidos pelo *RT-ColorAnt* e as melhores aproximações encontradas até o momento é de apenas 0,41%. Para os grafos *le450*, o *RT-ColorAnt* obteve resultados que se distanciam 9% das aproximações conhecidas. Por outro lado, os resultados obtidos para os grafos *aleatórios* e *flat* foram os que mais se distanciaram, obtendo respectivamente as distâncias médias de 11,32% e 18,07%. Estes resultados demonstram os casos extremos do *RT-ColorAnt*. Ele é ideal para ser aplicado a grafos *geométricos*, e ruim para ser aplicado a grafos *aleatórios* e *flat*.

Analisando o algoritmo proposto por um prisma diferente, a partir da quantidade de vértices do grafo, é possível perceber que aumentar a quantidade de vértices (até certo limiar) melhora a qualidade dos resultados. Na média, os melhores resultados seguem a seguinte distribuição de acordo com a quantidade de vértices dos grafos: 300 (17,86% de distância das melhores aproximações), 450 (9%), 500 (4,87%) e 1000 (20,23%). Deste prisma, é possível perceber que o *RT-ColorAnt* obtém os melhores resultados para grafos com 500 vértices. Além disto, quando a quantidade de vértices diminui ou aumenta, a qualidade dos resultados tende a diminuir. Contudo, como a quantidade de vértices a partir de 500 duplica, não é possível (com estes resultados) determinar o ponto limite da curva de melhora dos resultados. Em trabalhos futuros, novos experimentos serão feitos para determinar este limite.

Analisando os resultados individuais e não mais a média obtida por um grupo, os resultados demonstram que: (1) para o grupo composto por grafos com 450 vértices, a qualidade do resultado obtido para a instância `le450_15c` se distância duas vezes mais do que daqueles obtidos para outras instâncias do mesmo grupo; e (2) para 1000 vértices o desempenho de *RT-ColorAnt* para grafos aleatórios chega a ser 2 vezes melhor do que para os grafos *flat*.

Um ponto importante a ser destacado é o fato do algoritmo obter as melhores aproximações em um tempo de execução entre 1 e 30 minutos. Os melhores resultados foram obtidos para tempos de execução mais próximos do limite inferior, enquanto os piores resultados foram obtidos para tempos de execução mais próximos do limite superior. Portanto, os resultados demonstram que a melhora da qualidade das aproximações, em *RT-ColorAnt*, não está diretamente relacionada ao tempo de execução do algoritmo.

Em síntese, os resultados demonstram que as características dos grafos, juntamente com o uso dos mesmos parâmetros para todas as instâncias, influenciaram consideravelmente na execução do algoritmo, e não necessariamente o tempo de execução. Portanto, o ideal é que o algoritmo seja *calibrado* para cada entrada, ao invés de utilizar os mesmos parâmetros para todas as instâncias.

Os resultados do *RT-ColorAnt* para as instâncias apresentadas foram também comparados com os resultados obtidos pelas seguintes heurísticas de coloração: *ALS-COL*, *Tabucol*, *MMT*, *GH* e *MOR*. Tais heurísticas não foram implementadas durante a realização desse trabalho. Embora as condições de execução sejam distintas, é possível realizar uma comparação preliminar da qualidade dos resultados obtidos por *RT-ColorAnt*.

A Tabela 2 apresenta os resultados obtidos pelo *RT-ColorAnt* e pelas outras heurísticas. Os valores das outras heurísticas são os apresentados por Plumettaz *et al.* (2010) e indicam o menor valor de k tal que pelo menos uma k-coloração própria foi encontrada. Os valores nesta tabela aparecem em negrito quando χ ou k^* são atingidos.

Como no *RT-ColorAnt*, Plumettaz *et al.* (2010) também utilizou tempo máximo de execução de 1 hora. O autor também comenta que o algoritmo *MMT* teve um tempo limite de 100 minutos, mas nenhuma menção é feita com relação aos tempos dos outros algoritmos.

Tabela 2: Valores de k para *RT-ColorAnt* (*RTC*), *ALS-COL* (*ALS*), *Tabucol* (*TC*), *MMT*, *GH* e *MOR*.

Grafo	χ/k^*	*RTC*	*ALS*	*TC*	*MMT*	*GH*	*MOR*
dsjc500.1	?/12	13	**12**	**12**	**12**	**12**	**12**
dsjc500.5	?/48	53	**48**	49	**48**	**48**	49
dsjc500.9	?/126	132	127	127	127	**126**	**126**
dsjc1000.1	?/20	23	**20**	**20**	**20**	**20**	21
dsjc1000.5	?/83	98	86	89	**83**	**83**	88
dsjr500.1c	?/85	**85**	**85**	**85**	**85**	-	**85**
dsjr500.5	?/122	123	125	126	**122**	-	123
flat300_28_0	28/28	33	29	31	31	31	31
flat1000_76_0	76/82	97	85	88	**82**	83	89
le450_15c	15/15	17	**15**	16	**15**	**15**	**15**
le450_15d	15/15	16	**15**	**15**	**15**	**15**	**15**
le450_25c	25/25	27	26	26	**25**	26	**25**
le450_25d	25/25	27	26	26	**25**	26	**25**

Embora a qualidade das aproximações encontradas por *RT-ColorAnt* seja inferior à encontra por outras heurísticas, é importante observar que para apenas três instâncias (`dsjc500.9`, `dsjc1000.5`, `flat1000_76_0`) os resultados estão distantes. Sendo assim, para a maioria das instâncias os resultados estão bem próximos. Provavelmente *calibrando* os parâmetros do algoritmo para cada instância esta distância diminuirá.

Os resultados apresentados na Tabela 2 também demonstram que a heurística *MMT* possui um custo mais elevado do que aquela implementada pelo algoritmo *RT-ColorAnt*. Embora *RT-ColorAnt* obtenha aproximações inferiores àquelas obtidas por *MMT*, *RT-ColorAnt* possui um tempo de execução consideravelmente inferior. Isto é um atrativo para contextos onde reduzir o tempo de execução é um objetivo a ser alcançado.

Conforme mencionado anteriormente, não foi possível comparar os resultados do *RT-ColorAnt* com os do k-ANTCOL. Pelo mesmo motivo, nenhum algoritmo da Classe 1, que é a classe na qual o *RT-ColorAnt* mais se encaixa, foi considerado na comparação. Dessa forma, ainda não podem ser feitas comparações mais justas em relação a outros algoritmos que utilizam colônias de formigas artificiais.

6. Conclusões e Trabalhos Futuros

Este artigo apresentou um algoritmo baseado em colônia de formigas artificiais aplicado ao problema da coloração de grafos, denominado *RT-ColorAnt*.

Embora os resultados obtidos por *RT-ColorAnt* não sejam os mesmos das melhores aproximações conhecidas, estes se aproximam em alguns casos dos melhores algoritmos propostos na literatura. Contudo, um ponto positivo do *RT-ColorAnt* é o fato de sua viabilidade de aplicação para contextos nos quais o tempo de execução seja um fator crítico, e não necessariamente a qualidade da solução.

Os resultados demonstraram a necessidade de um ajuste nos parâmetros do algoritmo para cada entrada avaliada. Além disto, os resultados demonstraram que o tempo de execução não necessariamente melhora a qualidade do resultado.

Novos experimentos serão realizadas em trabalhos futuros. Os objetivos destes experimentos são: determinar o ponto limite da curva de melhora dos resultados, e determinar quais são os melhores parâmetros para cada instância avaliada. Outros trabalhos futuros compreendem: estudar

como as características de cada grafo influenciam no algoritmo e na escolha dos seus parâmetros; implementar as outras heurísticas para avaliá-las (e compará-las) sobre as mesmas condições; e adaptar a implementação do *RT-ColorAnt* de forma que ele possa ser utilizado por um alocador de registradores. E, um trabalho mais ambicioso, compreende tornar *RT-ColorAnt* auto-adaptável às características da instância de entrada, de forma que os parâmetros sejam *calibrados* automaticamente.

Referências

Ahn, S.; Lee, S.; Chung, T. Modified ant colony system for coloring graphs. In: *Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing, and the Fourth Pacific Rim Conference on Multimedia*. Singapura: IEEE Singapore Communications Chapter, 2003. v. 3, p. 1849–1853.

Appel, A. W. *Modern Compiler Implementation in C*. New York, NY, EUA: Cambridge University Press, 1998. 544 p.

Blöchliger, I.; Zufferey, N. A graph coloring heuristic using partial solutions and a reactive tabu scheme. *Computers & Operations Research*, Elsevier Science Ltd., Oxford, UK, UK, v. 35, n. 3, p. 960–975, 2008.

Bondy, J. A.; Murty, U. S. R. *Graph Theory*. New York, NY, EUA: Springer, 2008. 654 p. (Graduate Texts in Mathematics, v. 244).

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

Comellas, F.; Ozón, J. An ant algorithm for the graph colouring problem. In: *First International Workshop on Ant Colony Optimization*. Berlin, Heidelberg: Springer, 1998. p. 151–158.

Cormen, T. H. *et al. Introduction to Algorithms*. 3rd. ed. Cambridge, Massachusetts: MIT Press, 2009.

Costa, D.; Hertz, A. Ants can colour graphs. *The Journal of the Operational Research Society*, v. 48, n. 3, p. 295–305, 1997.

Dorigo, M.; Birattari, M.; Stützle, T. Ant colony optimization - artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, v. 1, n. 4, p. 28–39, 2006.

Dorigo, M.; Krzysztof, S. An introduction to ant colony optimization. *IRIDIA Technical Report Series*, 2006.

Dorigo, M.; Maniezzo, V.; Colorni, A. *Ant System: An Autocatalytic Optimizing Process*. Politecnico di Milano, Itália, 1991.

Dorigo, M.; Maniezzo, V.; Colorni, A. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, v. 26, n. 1, p. 29–41, 1996.

Dorigo, M.; Stützle, T. *Ant Colony Optimization*. Cambridge, Massachusetts: MIT Press, 2004. 305 p. (Bradford Books).

Galinier, P.; Hao, J.-K. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, Springer Netherlands, v. 3, n. 4, p. 379–397, 1999.

Galinier, P.; Hertz, A. A survey of local search methods for graph coloring. *Computers & Operations Research*, v. 33, n. 9, p. 2547–2562, 2006.

Glover, F. Tabu search - part i. *ORSA Journal on Computing*, v. 1, n. 3, p. 190–206, 1989.

Hertz, A.; Zufferey, N. A new ant algorithm for graph coloring. In: *Workshop on Nature Inspired Cooperative Strategies for Optimization NICSO*. Granada, Espanha: David Alejandro Pelta and Natalio Krasnogor, 2006. p. 51–60.

Hertz, A.; Zufferey, N. Vertex coloring using ant colonies. In: Monmarché, N.; Guinand, F.; Siarry, P. (Ed.). *Artificial Ants: From Collective Intelligence to Real-life Optimization and Beyond*. France: Wiley, 2010.

Johnson, D.; Trick, M. *Cliques, coloring, and satisfiability: second DIMACS implementation challenge*. Providence, RI, EUA: American Mathematical Society, 1996. 657 p. (DIMACS series in discrete mathematics and theoretical computer science). `http://dimacs.rutgers.edu/Volumes/Vol26.html`.

Karp, R. M. Reducibility among combinatorial problems. In: Miller, R.; Thatcher, J. (Ed.). *Complexity of Computer Computations*. New York, NY, EUA: Plenum Press, 1972. p. 85–103. `http://www.cs.berkeley.edu/~luca/cs172/karp.pdf`. Acesso em: 10/03/2011.

Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. Optimization by simulated annealing. *Science*, v. 220, n. 4598, p. 671–680, 1983.

Laguna, M.; Martí, R. A grasp for coloring sparse graphs. *Computational Optimization and Applications*, Kluwer Academic Publishers, Norwell, MA, EUA, v. 19, n. 2, p. 165–178, 2001.

Leighton, F. T. A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards*, v. 84, n. 6, p. 489–506, 1979.

Lourenço, H.; Martin, O.; Stützle, T. Iterated local search. In: Glover, F.; Kochenberger, G. (Ed.). *Handbook of Metaheuristics*. New York, NY, EUA: Springer, 2003, (International Series in Operations Research & Management Science, v. 57). p. 320–353.

Malaguti, E.; Monaci, M.; Toth, P. A metaheuristic approach for the vertex coloring problem. *INFORMS Journal on Computing*, v. 20, n. 2, p. 302–316, 2008.

Morgenstern, C. Distributed coloration neighborhood search. In: Johnson, D. S.; Trick, M. (Ed.). *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*. Providence, RI, EUA: American Mathematical Society, 1996, (DIMACS Series in Discrete Mathematics and Theoretical Computer Science, v. 26). p. 335–357.

Plumettaz, M.; Schindl, D.; Zufferey, N. Ant local search and its efficient adaptation to graph colouring. *Journal of the Operational Research Society*, v. 61, n. 5, p. 819–826, 2010.

Shawe-Taylor, J.; Zerovnik, J. Ants and graph coloring. In: *International Conference on Artificial Neural Nets and Genetic Algorithms, ICANNGA'01*. Berlin, Heidelberg: Springer, 2001. p. 276–279.