X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

ALGORITMO HEURÍSTICO BASEADO EM COLÔNIA DE FORMIGAS ARTIFICIAIS *COLORANT*$_2$ COM BUSCA LOCAL APLICADO AO PROBLEMA DE COLORAÇÃO DE GRAFO

Carla Négri Lintzmayer, Mauro Henrique Mulati, Anderson Faustino da Silva
Departamento de Informática – Universidade Estadual de Maringá (UEM)
{carla0negri, mhmulati}@gmail.com, anderson@din.uem.br

Resumo – O problema de coloração de grafo é $\mathcal{NP}$-difícil e é utilizado em aplicações práticas, como escalonamento de tarefas e alocação de registradores. Para obter soluções para este problema em tempo aceitável, a investigação reportada no presente trabalho utiliza a meta-heurística *Ant Colony Optimization*, cujo funcionamento é baseado no comportamento de formigas durante a busca por alimento em um ambiente. Com base nesta meta-heurística é apresentado o algoritmo *ColorAnt*$_2$, que utiliza Busca Tabu como busca local. Os experimentos realizados reportam que *ColorAnt*$_2$ é uma opção promissora para encontrar boas aproximações para grafos *geométricos* e/ou *le450* em um tempo de execução aceitável, como também para minimizar a quantidade de conflitos, o principal problema da coloração de grafo com um número fixo de cores.

Palavras-chave – Meta-heurística, Problema de Coloração de Grafo, Otimização por Colônia de Formigas Artificiais, *Ant Colony Optimization* e *ColorAnt*$_2$.

Abstract – The graph coloring problem is $\mathcal{NP}$-hard and it is used in many practical applications, such as task scheduling and register allocation. To obtain solutions for this problem in acceptable time, the investigation reported in this paper uses the Ant Colony Optimization metaheuristic whose operation is based on ant's behavior during the search for food in an environment. Based on this metaheuristic, it is presented the *ColorAnt*$_2$ algorithm, which uses Tabu Search as local search. The experiments demonstrated that *ColorAnt*$_2$ is a promising option to find good approximations for *geometric* and/or *le450* graphs in an acceptable runtime, but also minimizing the amount of conflict, the main problem of graph coloring with a fixed number of colors.

Keywords – Meta-heuristic, Graph Coloring Problem, Ant Colony Optimization and *ColorAnt*$_2$.

1 INTRODUÇÃO

Obter uma solução para o problema de coloração de grafo (PCG) consiste basicamente em 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, porém, 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 [1]. Assim, a menos que $\mathcal{P} = \mathcal{NP}$, não existem algoritmos exatos em tempo polinomial que possam resolver grandes instâncias [2], sendo necessárias técnicas alternativas para obter soluções satisfatorias. O caráter $\mathcal{NP}$-difícil do PCG tem levado à realização de trabalhos que exploram meta-heurísticas e algoritmos heurísticos [3–5]. E ainda, a importância do PCG pode ser verificada por suas apliações, como escalonamento de tarefas e alocação de registradores.

Dentre as meta-heurísticas, o presente trabalho destaca a Otimização por Colônia de Formigas Artificiais, ou *Ant Colony Optimization* (ACO), que se embasa no comportamento apresentado por formigas durante a busca por alimento em um ambiente [6]. Dentre os algoritmos ACO, o *Ant System* (AS) [7] foi o primeiro a surgir, sendo aplicado originalmente ao Problema do Caixeiro Viajante. Outros algoritmos ACO foram criados, como o *Max-Min Ant System* e o *Ant Colony System* [6], e obtiveram bom desempenho para alguns tipos de problemas. Diversos trabalhos propõem o uso de algoritmos ACO para a resolução de problemas [6], tais como: roteamento de veículos, atribuição de frequência, agendamento e coloração de grafo. Este último aparece em diversos problemas onde é necessário particionar um conjunto de elementos em vários grupos com determinadas características compatíveis entre os membros [8].

No presente trabalho, é reportada a investigação da aplicação ao PCG do algoritmo heurístico ACO *ColorAnt*$_2$, que se baseia no algoritmo *Colorant*$_1$ [9] (proposto originalmente como *RT-ColorAnt*, nome que já contempla a busca local), porém com diferenças na maneira de atualizar o feromônio. O *ColorAnt*$_1$ foi feito com embasamento no algoritmo *ANTCOL* [10] também com modificações no tratamento do feromônio. Ambos utilizaram a busca local React-Tabucol (RT) [11], que tem como base os conceitos de Busca Tabu. Os resultados obtidos por *ColorAnt*$_2$ demonstram que ele é uma boa opção para encontrar boas aproximações para grafos aleatórios e/ou geométricos com até 500 vértices. Outro diferencial de *ColorAnt*$_2$ é a sua capacidade de minimizar a quantidade de conflitos, o principal problema da coloração de grafo com um número fixo de cores.

O texto encontra-se organizado da seguinte forma: a Seção 2 descreve o PCG; a Seção 3 apresenta a meta-heurística ACO e as maneiras de aplicá-la ao PCG; a Seção 4 apresenta alguns trabalhos relacionados; a Seção 5 descreve o algoritmo proposto; a

Seção 6 apresenta resultados dos experimentos; e por fim, a Seção 7 apresenta as conclusões e os trabalhos futuros.

2 O PROBLEMA DE COLORAÇÃO DE GRAFO

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* [8] 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, de 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.

O problema da k-coloração descrito como um problema de decisão (k-PCG) é da seguinte forma [12]: 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 [8]. 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, a menos que $\mathcal{P} = \mathcal{NP}$) algoritmos exatos que processem grafos com mais de 100 vértices [3].

Uma aplicação real do k-PCG é vista na alocação de registradores [13]. 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.

3 OTIMIZAÇÃO POR 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*" [6]. 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 [14]. 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 tendência é que a colônia percorra 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 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 [15].

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" [6]. São exemplos de meta-heurísticas: *tabu search* [16] e *simulated annealing* [17].

A execução de um algoritmo ACO geralmente é composta por vários ciclos. 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 [15]. A informação heurística depende de cada problema.

Diferentes tipos de métodos baseados em colônias de formigas artificiais foram desenvolvidos para o PCG. Eles são classificados em três classes [18]: (1) constituído 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; (2) composto 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 (3) aqueles nos quais as formigas são algoritmos de busca local, onde, partindo-se de um grafo já colorido, cada formiga encontra uma solução vizinha, que normalmente é a atribuição de uma nova cor a um vértice conflitante da solução atual.

As duas últimas classes se diferenciam de forma significativa da idéia original de um algoritmo ACO e existem divergências com relação ao fato de poder considerar tais algoritmos como sendo "baseados em colônias de formigas artificiais" [18]. 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. Os algoritmos *ColorAnt*$_1$ e *ColorAnt*$_2$ encontram-se na Classe 1.

4 TRABALHOS RELACIONADOS

No primeiro trabalho sobre coloração de grafos com colônia de formigas, o *ANTCOL* [10], cada formiga tenta colorir o grafo com o menor valor de k possível, utilizando os métodos construtivos *RLF* (*Recursive Large First*) [19] e *Dsatur* [20]. Uma matriz $M_{|V|\times|V|}$ armazena a experiência das construções (feromônio). A trilha entre dois vértices não-adjacentes coloridos com a mesma cor é reforçada com o inverso do número de cores encontradas na solução de uma formiga. Os resultados do *ANTCOL*

não foram os melhores, mas foram bons o suficiente para influenciar novos estudos. O tratamento da matriz de feromônio no *ColorAnt*$_2$ é o mesmo do *ANTCOL*. Porém, a diferença está no uso da probabilidade, que envolve o feromônio e a informação heurística: no *ANTCOL* é utilizada para escolher um novo vértice a ser colorido, e no *ColorAnt*$_2$ é utilizada para escolher a cor que irá colorir um vértice.

Uma abordagem diferente trabalha com cada formiga se movendo, em uma iteração, com certa probabilidade, para um vértice adjacente que tenha o maior número de arestas conflitantes [21]. Nesse vértice, 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. Essa abordagem se encaixa na Classe 2 e não se assemelha com o que é realizado pelo *ColorAnt*$_2$.

Há um algoritmo para o k-PCG no qual cada formiga é um procedimento iterativo que tenta minimizar o número de conflitos [12]. A trilha de feromônio é modelada com base em um grafo G', inicialmente igual a G, ao qual vão sendo adicionadas arestas caso muitas formigas atribuiam diferentes cores a nós não-adjacentes. Foi avaliado com poucas instâncias, que são coloridas otimamente por algoritmos exatos em segundos. Pertence à Classe 3 e, portanto, também não é semelhante ao *ColorAnt*$_2$.

Outro algoritmo aplicado ao k-PCG trabalha com cada formiga colorindo um único vértice, de forma que a colônia inteira encontra apenas uma solução [22]. Uma cor entre as k é atribuida a cada formiga e k formigas ficam em cada vértice. Um procedimento baseado no *Dsatur* escolhe um vértice e atribui a cor de uma das formigas nele existentes (que minimize os conflitos). Com base na informação heurística e na trilha de feromônio, as formigas andam pelo grafo. Comparado ao *Dsatur*, *ANTCOL* original, *Tabucol* [4] e ao algoritmo genético híbrido *GH* [4], considerado a melhor heurística de coloração, o algoritmo supera apenas o *Dsatur* e o *ANTCOL* com *Dsatur* para grandes instâncias. Também da Classe 2, não se assemelha ao *ColorAnt*$_2$.

Um algoritmo mais recente, *ALS-COL* (*Ant Local Search*) [3], é o único que compete com os melhores algoritmos de coloração de grafo. Nele, cada formiga é uma busca local derivada da Busca Tabu para o k-PCG, que modifica colorações parciais: $C_1, \ldots, C_k$ classes legais e uma classe C_{k+1} de vértices não coloridos. A solução vizinha surge movendo um vértice $v \in C_{k+1}$ para alguma classe C_c e movendo os vizinhos de v que estão em C_c para C_{k+1}. O movimento (v, c) é escolhido em dois passos: um com base na informação heurística e outro com base no valor de feromônio (tratado quase como no *ANTCOL*). Foi comparado com o *PartialCol* [11], *Tabucol*, *GH*, *MOR* [23] e *MMT* [24], encontrando o número cromático ou o melhor valor conhecido de várias instâncias testadas e se aproximando bastante das outras. Ficou pior em algumas delas quando comparado ao *MMT* e ao *GH*, mas com pouca diferença. Sendo da Classe 3, este algoritmo é bem diferente do *ColorAnt*$_2$.

5 O ALGORITMO *COLORANT*$_2$

O algoritmo proposto, *ColorAnt*$_2$, utiliza como método construtivo para cada formiga o algoritmo *ANTCOL* modificado, que tenta colorir um grafo G com k cores fixas [10], chamado aqui de k-ANTCOL. Seu pseudocódigo 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;                // conjunto de vértices ainda não coloridos
2  cor_i = 0  ∀i ∈ V;     // vetor cor mantém um mapeamento vértice-cor
3  while ncoloridos < |V| do
4      v = arg max{gsat(v') | v' ∈ NC};
5      escolher uma cor c ∈ {1..k} com probabilidade p(s, v, c) dada pela Equação 1;
6      cor_v = c;
7      NC = NC\{v};
8      ncoloridos++;
9  return cor;            // retornar solução dada pelo vetor cor
```

A cada passo, o k-ANTCOL deve escolher um vértice v ainda não colorido e uma cor c para colorí-lo na solução s. É escolhido o vértice que possui o maior grau de saturação, $gsat(v)$, que 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, apresentada na Equação 1, que é calculada com base na trilha de feromônio τ, apresentada na Equação 2, e na informação heurística η, 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 a influência dos valores associados a eles na equação, e

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

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

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

A trilha de feromônio, armazenada na matriz $F_{|V|\times|V|}$, é inicializada com 1 para as ligações entre os vértices não-adjacentes e 0 para as ligações entre os vértices adjacentes. Sua atualização envolve a persistência por um fator ρ da trilha atual ($1-\rho$ é a taxa de evaporação) e o reforço da mesma por meio da experiência obtida nas soluções que as formigas geraram. A evaporação é dada pela Equação 4 e a forma geral de depósito de feromônio é mostrada na Equação 5.

$$F_{uv} = \rho F_{uv} \quad \forall u, v \in V \qquad (4) \qquad\qquad F_{uv} = F_{uv} + \frac{1}{f(s)} \quad \forall u, v \in C_c(s) \mid (u, v) \notin E, c = 1..k \qquad (5)$$

onde s é uma solução, $C_c(s)$ é o conjunto de vértices coloridos com a cor c na solução s e f é a função objetivo, que retorna o número de arestas conflitantes da solução.

Este trabalho apresenta uma diferença em relação à trilha de feromônio quando comparado ao k-ANTCOL original e ao *ColorAnt*$_1$: apenas a solução da melhor formiga da colônia no ciclo após a busca local (s') e a solução da melhor formiga encontrada durante toda a execução até o momento (s^*) reforçam a trilha. O *ColorAnt*$_1$ utilizava, além destas, todas as soluções encontradas pela colônia.

O algoritmo *ColorAnt*$_2$ utiliza então o k-ANTCOL como descrito no Algoritmo 1, sendo que para melhoria das soluções é utilizada como busca local a busca tabu reativa *React-Tabucol* (RT) [11]. O *ColorAnt*$_2$ é descrito no Algoritmo 2 com a denominação *ColorAnt*$_2$-RT, para contemplar a busca local sendo utilizada.

Algoritmo 2 Pseudocódigo do *ColorAnt*$_2$-RT.

```
COLORANT2-RT(G = (V, E), k)      // V é o conjunto de vertices e E o de arestas
1  F_uv = 1  ∀(u,v) ∉ E;        F_uv = 0  ∀(u,v) ∈ E;
2  s*.num_conflitos = ∞;
3  while tempo < max_tempo and f(s*) ≠ 0 do
4      s'.num_conflitos = ∞;
5      for a = 1 to nformigas do
6          s = K-ANTCOL(G, k);
7          if f(s) == 0 or f(s) < f(s') then
8              s' = s;
9      if f(s') ≠ 0 then
10         REACT_TABUCOL(s');
11     if f(s') < f(s*) then
12         s* = s';
13     F_uv = ρF_uv  ∀u, v ∈ V;          // evaporar feromônio de acordo com Equação 4
14     F_uv = F_uv + 1/f(s')  ∀u, v ∈ C_c(s') | (u, v) ∉ E, c = 1..k;
                                         // de acordo com forma geral da Equação 5
15     F_uv = F_uv + 1/f(s*)  ∀u, v ∈ C_c(s*) | (u, v) ∉ E, c = 1..k;
                                         // de acordo com forma geral da Equação 5
```

A busca tabu utilizada pelo *ColorAnt*$_2$-RT, a *React-Tabucol*, é descrita a seguir. Dados 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 (com ou sem arestas conflitantes) e uma solução inicial $s_0 \in S$, f deve ser minimizada sobre S. Um *movimento* é a troca da cor de um único vértice e ocorre entre duas soluções *vizinhas*. Quando ele ocorre, o movimento inverso é armazenado em uma *lista tabu* de forma que nas próximas t (*tabu tenure*) gerações esse movimento não possa ser realizado. A busca gera 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 não presentes na lista tabu, ou seja, a próxima solução deve ser gerada por um movimento não-tabu e ela deve possuir o menor número de conflitos entre as possíveis soluções vizinhas.

6 AVALIAÇÃO EXPERIMENTAL

ColorAnt$_1$-RT [9] e *ColorAnt*$_2$-RT foram implementados na linguagem `C` compilado com GCC 4.1.2 utilizando O3 e executados em um computador Intel Xeon E5620 de 2.40 GHz, 8GB RAM e sistema operacional Rocks.

6.1 METODOLOGIA

Os experimentos foram realizados em 13 grafos do Desafio DIMACS [5], os mesmos utilizados nos experimentos com *ALS* [3]. Desta forma, além de uma comparação do *ColorAnt*$_2$-RT com o *ColorAnt*$_1$-RT também é possível uma comparação dos resultados obtidos por estes dois 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;

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

- `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.

Nos experimentos com o ANTCOL [10] 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. Contudo, não foi possível identificar exatamente quais instâncias são essas. Portanto, não será possível comparar os resultados do *ColorAnt*$_2$ com os do algoritmo ANTCOL. Sabe-se apenas que os grafos `dsjc500.5` e `dsjc1000.5` fazem parte deste conjunto.

Um ponto importante na execução de algoritmos heurísticos é a sua calibragem dos parâmetros. Desta forma, vários experimentos foram realizados com o objetivo de encontrar possíveis parâmetros para os algoritmos desenvolvidos. Para ambas versões a calibragem foi realizada com experimentos com a instância `dsjc500.5`.

Para o *ColorAnt*$_1$-RT os valores de α e β variaram no conjunto $\{1..10\}$ e a quantidade de formigas variou no conjunto $\{20..300\}$. Tais experimentos indicaram que bons parâmetros são: $\alpha = 7$, $\beta = 9$, $\rho = 0,7$ e $nformigas = 180$. Para o *ColorAnt*$_2$-RT os valores de α e β variaram no conjunto $\{1..10\}$ e a quantidade de formigas variou no conjunto $\{20..300\}$. Estes experimentos com *ColorAnt*$_2$-RT indicaram que bons parâmetros são: $\alpha = 9$, $\beta = 10$, $\rho = 0,7$ e $nformigas = 180$. A busca tabu *React-Tabucol* foi limitada por um número máximo de 500 ciclos para *ColorAnt*$_1$-RT e 1000 ciclos para *ColorAnt*$_2$-RT. Embora os algoritmos possuam parâmetros distintos, ambos possuem em comum o fato da execução finalizar assim que uma solução própria seja encontrada ou quanto o tempo máximo de execução de 1 hora for atingido.

6.2 RESULTADOS E DISCUSSÃO

A Tabela 1 apresenta os resultados obtidos por *ColorAnt*$_1$-RT e *ColorAnt*$_2$-RT, 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. Cada algoritmo foi executado 10 vezes para cada valor de k a partir de χ ou de k^*, sempre 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(S)/total de execuções(T)*. A quinta coluna apresenta o *tempo médio até a melhor solução encontrada*[2] em segundos. A sexta coluna apresenta o número médio de conflitos de cada experimento para cada cor. As próximas quatro colunas possuem as mesmas informações das anteriores, contudo para a nova versão do algoritmo. Os resultados foram omitidos quando o número de execuções bem sucedidas foi zero, para simplificação da tabela.

Os resultados apresentados na Tabela 1 demonstram que o *ColorAnt*$_1$-RT possui os melhores resultados para grafos *geométricos*. Neste caso, a distância média entre os resultados obtidos pelo *ColorAnt*$_1$-RT e as melhores aproximações encontradas até o momento é de apenas 0,41%. Para os grafos *le450* os resultados se distanciam 9%, das aproximações conhecidas. Enquanto que, para os grafos *aleatórios* e *flat* *ColorAnt*$_1$-RT obteve resultados que mais se distanciam, respectivamente 11,32% e 18,07%.

O *ColorAnt*$_2$-RT possui um comportamento similar ao *ColorAnt*$_1$-RT, quando comparado a distância média entre os resultados obtidos e as melhores aproximações já encontradas. A menor distância foi obtida para os grafos *geométricos* (0,41%), seguido pelos grafos *le450* (7,33%), depois pelos grafos *aleatórios* (9,10%), e por fim pelos grafos *flat* (13,85%). Este resultados demonstram que as alterações contidas no *ColorAnt*$_2$-RT ocasionaram: (1) uma estabilidade para grafos *geométricos*; e (2) ganho de desempenho para grafos *le450*, *aleatórios* e *flats*. Estes resultados também demonstram os casos extremos de *ColorAnt*$_1$-RT e *ColorAnt*$_2$-RT. Eles são ideais para serem aplicados a grafos *geométricos*, e ruins para grafos *flat*. Contudo, *ColorAnt*$_2$-RT tem demonstrado ser a melhor escolha.

Analisando *ColorAnt*$_1$-RT de 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 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%). A distribuição obtida por *ColorAnt*$_2$-RT segue um padrão similar, a saber: 300 (14,29%), 450 (7,33%), 500 (4,13%) e 1000 (13,03%). Estes resultados demonstram que um aumento gradativo na quantidade de vértices, a partir de 300, ocasiona uma melhora na qualidade dos resultados até um limiar de 500 vértices e que após este limite existe uma queda significativa no desempenho de ambos os algoritmos. 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 realizados para determinar este limite.

Um ponto importante a ser destacado é o fato de *ColorAnt*$_2$ possuir um menor tempo de execução, quando comparado com *ColorAnt*$_1$ para as mesmas aproximações, exceto para as instâncias *dsjc100.5* e *le450_25d*, cujo tempo de execução aumentou respectivamente 23,66% e 16,29%. Neste caso a redução do tempo de execução chega a 93%. Por outro lado, quanto é comparado o tempo de execução para as melhores aproximações encontradas, houve um aumento no tempo de execução para quatro instâncias, a saber: *dsjc1000.1* (2206,99%), *dsjc1000.5* (23,41%), *flat300_28_0* (35,88%) e *flat1000_76_6* (80,04%). Contudo, para todas estas instâncias *ColorAnt*$_2$ melhorou a qualidade das aproximações obtidas por *ColorAnt*$_1$. É interessante observar que

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

para quatro instâncias, *dsjc500.5*, *dsjc500.9*, *dsjc1000.5* e *le450_15c*, *ColorAnt*$_2$ além de reduzir o tempo de execução melhorou a qualidade das aproximações. Isto possui seus prós e contras. Quando a função objetivo é maximizar a aproximação obtida, por exemplo alocação de recursos, o aumento do tempo de execução não é necessariamente um fator negativo. Por outro lado, em algumas aplicações isto não é tolerável, por exemplo compiladores dinâmicos [25].

Tabela 1: Resultados obtidos pelos algoritmos *ColorAnt*$_1$-RT e *ColorAnt*$_2$-RT.

		ColorAnt$_1$-RT				*ColorAnt*$_2$-RT			
Grafo	χ/k^*	k	S/T	Tempo (s)	Conflitos	k	S/T	Tempo (s)	Conflitos
dsjc500.1	?/12	13	10/10	385,043	0	13	10/10	25,899	0
dsjc500.5	?/48	53	3/10	2204,815	3,1	52	8/10	1026,574	1
		54	10/10	509,998	0	53	10/10	254,907	0
dsjc500.9	?/126	132	1/10	1552,924	5,9	130	3/10	1097,152	2,7
		133	2/10	1503,506	3,1	131	7/10	906,681	1,3
		134	8/10	937,453	0,8	132	10/10	339,419	0
		135	10/10	202,384	0	-	-	-	-
dsjc1000.1	?/20	23	10/10	20,644	0	22	10/10	476,245	0
dsjc1000.5	?/83	98	6/10	1080,445	2,2	96	8/10	1333,387	0,8
		99	10/10	600,418	0	97	10/10	762,127	0
dsjr500.1c	?/85	**85**	3/10	708,217	1,9	**85**	2/10	875,753	1,5
		86	9/10	778,647	0,2	86	6/10	264,867	1
		87	10/10	32,443	0	87	10/10	142,141	0
dsjr500.5	?/122	123	3/10	518,887	1,1	123	1/10	18,505	2
		124	3/10	701,903	0,8	124	9/10	40,122	0.1
		125	10/10	7,564	0	125	10/10	1,592	0
flat300_ 28_0	28/28	33	8/10	893,705	0,6	32	6/10	1214,337	1,7
		34	10/10	29,934	0	33	10/10	26,682	0
flat1000_76_0	76/82	97	3/10	1194,426	3,4	93	1/10	2150,411	7,7
		98	10/10	335,324	0	94	1/10	1323,365	6,3
		-	-	-	-	95	6/10	1115,152	1,4
		-	-	-	-	96	9/10	479,534	0,4
		-	-	-	-	97	10/10	165,307	0
le450_15c	15/15	17	1/10	1673,505	52,2	16	6/10	380,142	35,5
		18	0/10	1315,475	38,3	17	8/10	154,618	11,7
		19	1/10	1358,056	22,8	18	6/10	530,114	11,8
		20	0/10	1352,943	9,8	19	6/10	732,335	7,4
		21	5/10	559,859	2,3	20	6/10	580,620	2,1
		22	10/10	11,839	0	21	10/10	54,319	0
le450_15d	15/15	16	1/10	1800,629	69,3	16	8/10	480,133	18,2
		17	1/10	2132,706	52,1	17	9/10	220,356	6,3
		18	3/10	1716,221	26,8	18	10/10	165,139	0
		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	25/25	27	7/10	884,212	1,1	27	10/10	389,834	0
		28	10/10	0,879	0	-	-	-	-
le450_25d	25/25	27	7/10	366,113	1	27	10/10	425,758	0
		28	10/10	0,935	0	-	-	-	-

Outro ponto a ser destacado é a capacidade de *ColorAnt*$_2$ reduzir a quantidade de conflitos, que aumentou apenas para *dsjr500.5* (para as mesmas aproximações melhores). Para as outras instâncias a redução na quantidade de conflitos variou entre 8% e 100%. Isto demonstra que mesmo que *ColorAnt*$_2$ não melhore a qualidade das aproximações obtidas por *ColorAnt*$_1$, em geral *ColorAnt*$_2$ reduz significativamente a quantidade de conflitos. Isto é excelente no contexto de alocação de registradores, onde o objetivo das heurísticas utilizadas não é necessariamente encontrar a melhor coloração, mas sim encontrar uma coloração que não possua conflitos.

Em síntese, os resultados demonstram que as características dos grafos, juntamente com o uso dos mesmos parâmetros para todas as instâncias, influenciaram na execução de *ColorAnt*$_2$, e não necessariamente o tempo de execução. Portanto, o ideal é que *ColorAnt*$_2$ seja *calibrado* para cada instância, ao invés de utilizar os mesmos parâmetros para todas elas.

Os resultados das versões dos algoritmos *ColorAnt*$_x$-RT para as instâncias apresentadas foram também comparadas 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 *ColorAnt*$_2$-RT.

A Tabela 2 apresenta os resultados obtidos pelos *ColorAnt*$_x$-RT e pelas outras heurísticas. Os valores das outras heurísticas são os apresentados para comparação com o *ALS* [3] 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 em *ColorAnt*$_1$-RT e *ColorAnt*$_2$-RT, o *ALS* também utilizou tempo máximo de execução de 1 hora. O algoritmo *MMT* teve tempo limite de 100 minutos [3], mas nenhuma menção é feita com relação aos tempos dos outros algoritmos.

Tabela 2: Valores de k para *ColorAnt*$_2$-RT (*CA*$_2$*-RT*), *ColorAnt*$_1$ (*CA*$_1$*-RT*), *ALS-COL* (*ALS*), *Tabucol* (*TC*), *MMT*, *GH* e *MOR*.

Grafo	χ/k^*	*CA*$_2$*-RT*	*CA*$_1$*-RT*	*ALS*	*TC*	*MMT*	*GH*	*MOR*
dsjc500.1	?/12	13	13	**12**	**12**	**12**	**12**	**12**
dsjc500.5	?/48	52	53	**48**	49	**48**	**48**	49
dsjc500.9	?/126	130	132	127	127	127	**126**	**126**
dsjc1000.1	?/20	22	23	**20**	**20**	**20**	**20**	21
dsjc1000.5	?/83	96	98	86	89	**83**	**83**	88
dsjr500.1c	?/85	**85**	**85**	**85**	**85**	**85**	-	**85**
dsjr500.5	?/122	123	123	125	126	**122**	-	123
flat300_28_0	28/28	32	33	29	31	31	31	31
flat1000_76_0	76/82	93	97	85	88	**82**	83	89
le450_15c	15/15	16	17	**15**	16	**15**	**15**	**15**
le450_15d	15/15	16	16	**15**	**15**	**15**	**15**	**15**
le450_25c	25/25	27	27	26	26	**25**	26	**25**
le450_25d	25/25	27	27	26	26	**25**	26	**25**

Embora a qualidade das aproximações encontradas por *ColorAnt*$_2$-RT seja inferior à encontra por outras heurísticas, é importante observar que apenas para quatro instâncias (*dsjc500.5*, *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 *ColorAnt*$_2$-RT. Embora *ColorAnt*$_2$-RT obtenha aproximações piores do que aquelas obtidas por *MMT*, *ColorAnt*$_2$-RT possui um tempo de execução consideravelmente melhor. 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 *ColorAnt*$_2$-RT com os do ANTCOL. Pelo mesmo motivo, nenhum algoritmo da Classe 1, que é a classe na qual o *ColorAnt*$_2$-RT 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.

7 CONCLUSÕES E TRABALHOS FUTUROS

Este artigo apresentou um algoritmo baseado em colônia de formigas artificiais aplicado ao problema da coloração de grafo, denominado *ColorAnt*$_2$-RT. Embora os resultados obtidos por *ColorAnt*$_2$-RT 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 *ColorAnt*$_2$-RT é o fato de ter minimizado a quantidade de conflitos, e em muitos casos ter reduzido o tempo de execução quando comparado com sua versão anterior: *ColorAnt*$_1$-RT. Tais 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 outras heurísticas para avaliá-las (e compará-las) sobre as mesmas condições; estudar e implementar estratégias que reduzam o tempo de execução do algoritmo; e adaptar a implementação do *ColorAnt*$_2$-RT de forma que ele possa ser utilizado por um alocador de registradores. E, um trabalho mais ambicioso, compreende tornar *ColorAnt*$_2$-RT auto-adaptável às características da instância de entrada, de forma que os parâmetros sejam *calibrados* automaticamente.

REFERÊNCIAS

[1] R. M. Karp. "Reducibility Among Combinatorial Problems". In *Complexity of Computer Computations*, edited by R. Miller and J. Thatcher, pp. 85–103. Plenum Press, New York, NY, EUA, 1972.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, third edition, 2009.

[3] M. Plumettaz, D. Schindl and N. Zufferey. "Ant Local Search and its efficient adaptation to graph colouring". *Journal of the Operational Research Society*, vol. 61, no. 5, pp. 819–826, 2010.

[4] P. Galinier and J.-K. Hao. "Hybrid Evolutionary Algorithms for Graph Coloring". *Journal of Combinatorial Optimization*, vol. 3, no. 4, pp. 379–397, 1999.

[5] D. Johnson and M. Trick. *Cliques, coloring, and satisfiability: second DIMACS implementation challenge*. DIMACS series in discrete mathematics and theoretical computer science. American Mathematical Society, Providence, RI, EUA, 1996.

[6] M. Dorigo and T. Stützle. *Ant Colony Optimization*. Bradford Books. MIT Press, Cambridge, Massachusetts, 2004.

[7] M. Dorigo, V. Maniezzo and A. Colorni. "Ant System: An Autocatalytic Optimizing Process". Technical Report TR91-016, Politecnico di Milano, Politecnico di Milano, Itália, 1991.

[8] J. A. Bondy and U. S. R. Murty. *Graph Theory*, volume 244 of *Graduate Texts in Mathematics*. Springer, New York, NY, EUA, 2008.

[9] C. N. Lintzmayer, M. H. Mulati and A. F. da Silva. "RT-ColorAnt: Um Algoritmo Heurístico Baseado em Colônia de Formigas Artificiais com Busca Local para Colorir Grafos". In *XLIII SBPO 2011*, Ubatuba, SP, BRA, 2011.

[10] D. Costa and A. Hertz. "Ants Can Colour Graphs". *The Journal of the Operational Research Society*, vol. 48, no. 3, pp. 295–305, 1997.

[11] I. Blöchliger and N. Zufferey. "A graph coloring heuristic using partial solutions and a reactive tabu scheme". *Computers & Operations Research*, vol. 35, no. 3, pp. 960–975, 2008.

[12] J. Shawe-Taylor and J. Zerovnik. "Ants and graph coloring". In *International Conference on Artificial Neural Nets and Genetic Algorithms, ICANNGA'01*, pp. 276–279, Berlin, Heidelberg, 2001. Springer.

[13] A. W. Appel. *Modern Compiler Implementation in C*. Cambridge University Press, New York, NY, EUA, 1998.

[14] M. Dorigo, V. Maniezzo and A. Colorni. "The Ant System: Optimization by a colony of cooperating agents". *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, vol. 26, no. 1, pp. 29–41, 1996.

[15] M. Dorigo and S. Krzysztof. "An Introduction to Ant Colony Optimization". *IRIDIA Technical Report Series*, 2006.

[16] F. Glover. "Tabu Search - Part I". *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[17] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. "Optimization by Simulated Annealing". *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[18] A. Hertz and N. Zufferey. "Vertex Coloring Using Ant Colonies". In *Artificial Ants: From Collective Intelligence to Real-life Optimization and Beyond*, edited by N. Monmarché, F. Guinand and P. Siarry, France, 2010. Wiley.

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

[20] 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. 489–506, 1979.

[21] F. Comellas and J. Ozón. "An Ant Algorithm for the Graph Colouring Problem". In *First International Workshop on Ant Colony Optimization*, pp. 151–158, Berlin, Heidelberg, 1998. Springer.

[22] A. Hertz and N. Zufferey. "A New Ant Algorithm for Graph Coloring". In *Workshop on Nature Inspired Cooperative Strategies for Optimization NICSO*, pp. 51–60, Granada, Espanha, 2006. David Alejandro Pelta and Natalio Krasnogor.

[23] C. Morgenstern. "Distributed Coloration Neighborhood Search". In *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, edited by D. S. Johnson and M. Trick, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 335–357. American Mathematical Society, Providence, RI, EUA, 1996.

[24] E. Malaguti, M. Monaci and P. Toth. "A Metaheuristic Approach for the Vertex Coloring Problem". *INFORMS Journal on Computing*, vol. 20, no. 2, pp. 302–316, 2008.

[25] M. L. Scott. *Programming Language Pragmatics*. Morgan Kaufmann Publishers, third edition, 2009.