Learning and Nonlinear Models - Journal of the Brazilian Computational Intelligence Society, Vol. 11, Iss. 2, pp. 74–91, 2013.

O IMPACTO DA PARAMETRIZAÇÃO NO ALGORITMO HEURÍSTICO BASEADO EM COLÔNIA DE FORMIGAS ARTIFICIAIS *COLORANT*$_3$*-RT*

Carla Négri Lintzmayer
Instituto de Computação – Universidade de Campinas (UNICAMP)
carla0negri@gmail.com
Mauro Henrique Mulati
Departamento de Computação – Universidade do Centro Oeste (UNICENTRO)
mhmulati@gmail.com
Anderson Faustino da Silva
Departamento de Informática – Universidade Estadual de Maringá (UEM)
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 foi desenvolvida uma classe de algoritmos denominada *ColorAnt-RT*, cujos algoritmos são baseados no comportamento de formigas durante a busca por alimento em um ambiente. Tais algoritmos têm demonstrado serem boas opções para encontrar boas soluções para o problema de coloração de grafo. Contudo, uma questão ainda não investigada é a influência dos parâmetros na qualidade dos resultados encontrados, mais precisamente a influência do tamanho da colônia de formigas e da quantidade de ciclos da busca local utilizada. Desta forma, este artigo apresenta uma investigação detalhada de tal influência no melhor algoritmo da classe *ColorAnt-RT*, a saber: *ColorAnt*$_3$*-RT*. Os resultados demonstraram que a obtenção de bons resultados está relacionado a pelo menos duas questões: uma calibragem adequada de *ColorAnt*$_3$*-RT*, como também o uso de uma estratégia construtiva em conjunto com uma estratégia melhorativa.

Palavras-chave – Meta-heurística, Problema de Coloração de Grafo, Otimização por Colônia de Formigas e *ColorAnt*$_3$*-RT*.

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 was proposed a class of algorithms that was named *ColorAnt-RT*, which algorithms are based on ant's behavior during the search for food in an environment. These algorithms have demonstrated that they are good options to find good solutions for the graph coloring problem. However, a question doesn't analyzed is the parameters influence on the quality of the results, more precisely the influence of ant colony size as well as the amount of cycles used by the local search. Thus, this paper presents a detailed investigation of this influence in the best algorithm of the *ColorAnt-RT* class, namely: *ColorAnt*$_3$*-RT*. The results demonstrated that to obtain good results it necessary to take care of at least two questions: an correct calibration of *ColorAnt*$_3$*-RT*, and also the use of a constructive strategy together with a strategy that is able to improve the results.

Keywords – Metaheuristic, Graph Coloring Problem, Ant Colony Optimization and *ColorAnt*$_3$*-RT*.

1 INTRODUÇÃO

O campo de pesquisa da inteligência coletiva é inspirado no comportamento social de enxames, os quais são compostos por indivíduos que cooperam e se organizam sem a necessidade de um controle central. Neste contexto uma metaheurística bastante explorada é 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 [1]. O *Ant System* (AS) [2] foi o primeiro algoritmo ACo a surgir, sendo aplicado originalmente ao Problema do Caixeiro Viajante. Além deste outros algoritmos ACO foram criados, como o *Max-Min Ant System* [3] e o *Ant Colony System* [1], e obtiveram bom desempenho para alguns tipos de problemas. Diversos trabalhos propõem o uso de algoritmos ACO para a resolução de problemas, tais como: organização de sites [4], processamento de sequências de DNA [5], escalonamento [6], rotas de robôs [7] e coloração de grafo [8] [9] [10] [11], entre outros. Este último aparece em diversos problemas onde é necessário particionar um conjunto de elementos em vários grupos com determinadas características compatíveis entre os membros [12].

Especificamente, 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 [13]. Assim, a menos que $\mathcal{P} = \mathcal{NP}$, não existem algoritmos exatos em tempo polinomial que possam resolver grandes instâncias [14], sendo necessárias técnicas alternativas para obter soluções satisfatórias.

O caráter $\mathcal{NP}$-difícil do PCG tem levado à realização de trabalhos que exploram metaheurísticas e algoritmos heurísticos [15–17]. E ainda, a importância do PCG pode ser verificada por suas aplicações, como escalonamento de tarefas e alocação de registradores. Portanto, é justificável investigar algoritmos que resolvam este problema de forma satisfatória.

A investigação de uma solução para o PCG ocasionou o desenvolvimento da classe de algoritmos *ColorAnt-RT*. Em uma investigação inicial foi proposto o algoritmo *ColorAnt*$_1$*-RT* no qual além de cada formiga da colônia ser utilizada para atualizar a trilha de feromônio, a melhor formiga da colônia no ciclo (s') e a melhor formiga até o momento (s^*) também são utilizadas para este fim [9]. No segundo algoritmo proposto, *ColorAnt*$_2$*-RT*, apenas s' e s^* são utilizadas para atualizar a trilha de feromônio [10]. E por fim no terceiro algoritmo proposto, *ColorAnt*$_3$*-RT*, s' e s^* não atualizam a trilha de feromônio simultaneamente. Neste último, inicialmente s' atualiza mais frequentemente do que s^* e ocorre uma gradual mudança na frequência que é feita baseada no número máximo de ciclos do algoritmo, em cada intervalo de ciclos a quantidade de ciclos na qual s^* irá atualizar a trilha de feromônio (ao invés de s') é incrementada em uma unidade [11]. De fato *ColorAnt*$_3$*-RT* é o melhor algoritmo da classe *ColorAnt-RT*.

A aplicação de *ColorAnt*$_3$*-RT* na resolução do problema de alocação de registradores [18–20] demonstrou que a utilização de uma estratégia baseada em metaheurística para a resolução do PCG tende a gerar resultados melhores do que aqueles obtidos por algoritmos tradicionais de alocação de registradores baseados em coloração de grafo [21]. Portanto, isto demonstra o grande potencial dos algoritmos *ColorAnt-RT*.

Uma questão ainda não abordada nos trabalhos realizados com os algoritmos *ColorAnt-RT* é a investigação da qualidade dos resultados mediante parâmetros obtidos com estratégias distintas. Além disto, embora a literatura aborde que o uso de colônia de formigas sem o auxílio de busca local não obtenha resultados satisfatórios, não existe uma investigação que comprove isto no contexto de *ColorAnt-RT*. Assim o objetivo deste artigo é investigar o impacto da escolha dos parâmetros na qualidade dos resultados obtidos pelo melhor algoritmo da classe *ColorAnt-RT*. Com isto será possível determinar o limite máximo de desempenho de tal algoritmo, além de corroborar com os estudos apresentados pela literatura. Portanto, as contribuições deste artigo são:

1. Uma análise detalhada do impacto dos parâmetros no algoritmo *ColorAnt*$_3$*-RT*;
2. Um estudo comparativo entre o desempenho de colônia de formigas e busca local;
3. Perspectivas quanto a novas estratégias que possam ser implementadas para melhorar o desempenho de *ColorAnt*$_3$*-RT*.

A investigação realizada com *ColorAnt*$_3$*-RT* considerou 14 instâncias de grafos utilizados para avaliar algoritmos ACO. E os resultados demonstraram que a estratégia utilizada na parametrização de *ColorAnt*$_3$*-RT* pode ocasionar uma perda de desempenho quanto ao potencial que tal algoritmo pode alcançar, tanto na redução do tempo de execução quanto principalmente na qualidade das aproximações por ele encontradas. Outra questão importante também demonstrada nesta investigação é o fato de uma estratégia melhorativa proporcionar um ganho de desempenho a uma estratégia simplesmente construtiva.

O restante deste artigo está organizado como segue. A Seção 2 descreve o PCG. A Seção 3 apresenta a metaheurística ACO. A Seção 4 apresenta alguns trabalhos relacionados. A Seção 5 descreve detalhadamente o algoritmo *ColorAnt*$_3$*-RT*. A Seção 6 apresenta os resultados obtidos por *ColorAnt*$_3$*-RT* mediante diferentes parametrizações, juntamente com uma discussão. E por fim, a Seção 7 apresenta as considerações finais.

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 deste mesmo problema é o particionamento de V em k conjuntos independentes ou *classes legais* de cores $s = \{C_1, C_2, ..., C_k\}$. Neste contexto, uma coloração é *própria* [12] quando ela não possui nenhuma *aresta conflitante*, ou seja, não existem vértices adjacentes com a mesma cor e 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. Portanto, o alvo é 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 [22]: 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 [12]. Para outros casos não especiais, o PCG necessita de técnicas que o resolvam de maneira satisfatória, já que não existem algoritmos exatos em tempo polinomial que possam resolver grandes instâncias, a menos que $\mathcal{P} = \mathcal{NP}$ [14].

Uma aplicação real do k-PCG é vista na alocação de registradores [19]. 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 com apenas 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*" [1]. 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 [2]. 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 [23].

ACO é uma metaheurística de otimização combinatória que se baseia no comportamento das formigas artificiais. Uma metaheurística "é um conjunto de conceitos algorítmicos que podem ser usados para definir métodos heurísticos aplicáveis a um largo conjunto de diferentes problemas" [1]. São exemplos de metaheurísticas: busca tabu [24] e *simulated annealing* [25].

A execução de um algoritmo ACO é composta por diversos ciclos, onde em cada ciclo 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" é utilizada uma probabilidade calculada com base em dois fatores: a trilha de feromônio e a informação heurística (desejabilidade ou atratividade) relacionadas ao item [23]. Neste contexto, a informação heurística é dependente de cada problema.

Diferentes tipos de métodos baseados em colônias de formigas artificiais foram desenvolvidos para o PCG. Eles são classificados em três classes [26]: (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" [26]. 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-RT* encontram-se na Classe 1.

4 TRABALHOS RELACIONADOS

No primeiro trabalho sobre coloração de grafos com colônia de formigas, o *ANTCOL* [27], cada formiga tenta colorir o grafo com o menor valor de k possível, utilizando os métodos construtivos *RLF* (*Recursive Large First*) [28] e *Dsatur* [29]. 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*$_3$*-RT* 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*$_3$*-RT* 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 [30]. 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*$_3$*-RT*.

Há um algoritmo para o k-PCG no qual cada formiga é um procedimento iterativo que tenta minimizar o número de conflitos [22]. 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. Este algoritmo 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*$_3$*-RT*.

Outro algoritmo aplicado ao k-PCG trabalha com cada formiga colorindo um único vértice, de forma que a colônia inteira encontra apenas uma solução [31]. Uma cor entre as k cores possíveis 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* [16] e ao algoritmo genético híbrido *GH* (Galinier e Hao) [16], considerado a melhor heurística de coloração, o algoritmo superou apenas o *Dsatur* e o *ANTCOL* com *Dsatur* para grandes instâncias. Também da Classe 2, não se assemelha ao *ColorAnt*$_3$*-RT*.

Um algoritmo mais recente, *ALS-COL* (*Ant Local Search*) [15], 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 de forma similar ao *ANTCOL*). Este foi comparado com o *PartialCol* [32], *Tabucol*, *GH*, *MOR* (algoritmo de Morgenstern) [33] e *MMT* (algoritmo de Malaguti, Monaci e Toth) [34], encontrando o número cromático ou o melhor valor conhecido de várias instâncias e se aproximando bastante das outras. Portém, este ficou pior em algumas instâncias quando comparado ao *MMT* e ao *GH*, mas com pouca diferença. Sendo da Classe 3, este algoritmo é bem diferente do *ColorAnt*$_3$*-RT*.

5 O ALGORITMO *COLORANT*$_3$*-RT*

O algoritmo proposto, *ColorAnt*$_3$*-RT*, utiliza como método construtivo para cada formiga o algoritmo *ANTCOL* modificado, que tenta colorir um grafo G com k cores fixas [27], chamado aqui de `Ant_Fixed_k`, cujo pseudocódigo é descrito no Algoritmo 1.

Algoritmo 1 *Ant_Fixed_k*

ANT_FIXED_K($G = (V, E)$, k) // V: vértices; E: arestas

```
1  NC = V;                 // conjunto de vértices não coloridos
2  s(i) = 0  ∀i ∈ V;       // s mapeia um vértice a uma cor
3  while NC ≠ {} do
4      escolhe um vértice v com o maior grau de saturação em NC;
5      escolhe uma cor c ∈ 1..k com probabilidade p de acordo com a Equação 1;
6      s(v) = c;
7      NC = NC \{v};
8  return s;   // retorna a solução construída
```

A cada passo, `Ant_Fixed_k` deve escolher um vértice v ainda não colorido e uma cor c para colorí-lo na solução s. Neste processo 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. `Ant_Fixed_k` escolhe a cor $c \in \{1..k\}$ 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.

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

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

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 (sendo $1 - \rho$ 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 apresentada 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 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.

De fato *ColorAnt*$_3$*-RT* é o melhor algoritmo da classe *ColorAnt-RT*, sendo este obtido pela modificação na maneira de depositar feromônio, como descrito anteriormente. Como seus predecessores *ColorAnt*$_3$*-RT* utiliza para melhoria das soluções a busca tabu reativa *React-Tabucol* [32], contudo cada *ColorAnt-RT* tem a sua peculiaridade quanto a estratégia de aplicar a busca local. Em *ColorAnt*$_1$*-RT* e *ColorAnt*$_2$*-RT* a busca local é aplicada apenas na melhor formiga da colônia e ao final de um ciclo. Enquanto em *ColorAnt*$_3$*-RT* a busca local é aplicada em todas as formigas da colônia em cada ciclo. O *ColorAnt*$_3$*-RT* é descrito no Algoritmo 2.

Algoritmo 2 *ColorAnt$_3$-RT*

COLORANT$_3$-RT($G = (V, E), k$) // V: vértices; E: arestas

1 $P_{uv} = 1 \ \forall (u, v) \notin E$;
2 $P_{uv} = 0 \ \forall (u, v) \in E$;
3 $f^* = \infty$; // melhor valor da função objetivo até o momento
4 **while** $cycle < max\_cycles$ **and** $time < max\_time$ **and** $f^* \neq 0$ **do**
5 $f' = \infty$; // melhor valor da função em um ciclo
6 **for** $a = 1$ **to** $nants$ **do**
7 $s =$ ANT_FIXED_K(G, k);
8 $s =$ REACT_TABUCOL(G, k, s);
9 **if** $f(s) == 0$ **or** $f(s) < f'$ **then**
10 $s' = s$;
11 $f' = f(s')$;
12 **if** $f' < f^*$ **then**
13 $s^* = s'$;
14 $f^* = f(s^*)$;
15 $P_{uv} = \rho P_{uv} \ \ \forall u, v \in V$; (Equação 4)
16 **if** $cycle \mod \sqrt{max\_cycles} == 0$ **then**
17 $phero\_counter = cycle \div \sqrt{max\_cycles}$;
18 **if** $phero\_counter > 0$ **then**
19 $P_{uv} = P_{uv} + \frac{1}{f(s^*)} \quad \forall u, v \in C_c(s^*) \mid (u, v) \notin E, c = 1..k$; (Equação 5)
20 **else**
21 $P_{uv} = P_{uv} + \frac{1}{f(s')} \quad \forall u, v \in C_c(s') \mid (u, v) \notin E, c = 1..k$; (Equação 5)
22 $phero\_counter = phero\_counter - 1$;
23 $cycle = cycle + 1$;

5.1 A BUSCA LOCAL *REACT-TABUCOL*

A busca local utilizada por *ColorAnt$_3$-RT* é um algoritmo do tipo busca tabu (*Tabucol*) que foi criado inicialmente por [35] e vem sendo utilizado em vários trabalhos por ser o algoritmo mais simples, rápido e eficiente entre os melhores procedimentos de busca local [32]. A busca local *Tabucol* é como 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 (provavelmente com arestas conflitantes) e uma solução inicial $s_0 \in S$, f deve ser minimizada sobre S. A cada iteração, a melhor solução vizinha é escolhida para substituir a solução atual. Uma solução vizinha é obtida movendo um vértice v de sua classe de cor atual ($C_{s(v)}$) para uma classe nova C_c, esse *movimento* é representado por (v, c). O vértice v deve ser conflitante com pelo menos um vértice que pertença à sua classe. Quando o movimento (v, c) ocorre, o par $(v, s(v))$ é classificado como tabu pelas próximas tl iterações, garantindo que v não volte a pertencer à classe de cores $C_{s(v)}$ neste período.

A busca gera então uma sequência $s_1, s_2, \ldots$ de soluções em S, na qual s_{i+1} é vizinha de s_i e deve ser gerada por um movimento não-tabu e possuir o menor número de conflitos entre as possíveis soluções vizinhas de s_i, a não ser que ela leve a um valor de função objetivo melhor do que o melhor encontrado até o momento durante a busca (critério de aspiração).

O parâmetro tl é chamado de *tabu tenure*, ou, algumas vezes, de *comprimento da lista tabu*. Normalmente é utilizado um esquema *dinâmico* de *tabu tenure*, cujo valor depende da solução atual e do movimento que foi executado. O *tabu tenure dinâmico* é dado por $tl = \alpha f(s) + \text{RANDOM}(A)$, onde A e α são parâmetros passados ao algoritmo. O ajuste do *tabu tenure* depende de como a função objetivo evolui durante a busca. Três parâmetros auxiliam nessa atualização: φ (frequência), η (incremento) e δ (limiar). A cada φ iterações é determinado Δ, que é a diferença entre os valores máximo e mínimo que a função objetivo atingiu nas últimas φ iterações. E desta forma, o novo valor de tl será:

$$tl = \begin{cases} tl + \eta & \text{se } \Delta \leq \delta \\ tl - 1 & \text{caso contrário} \end{cases}$$

React-Tabucol nomeia o algoritmo com uso do *tabu tenure reativo*, cujo valor é determinado com base no histórico de busca. Portanto em *React-Tabucol* tl é atualizado de acordo com o esquema tabu reativo, indicando que quando uma solução é visitada duas vezes, o tamanho de tl é modificado. O algoritmo *Tabucol* utilizado pela classe *ColorAnt-RT* é apresentado no Algoritmo 3.

É importante notar que *React-Tabucol* trabalha em um espaço de soluções que contém k-colorações (próprias ou impróprias). Assim, qualquer que seja a solução inicial s_0 fornecida ao algoritmo, ela deve obedecer a esta "restrição".

Algoritmo 3 *React-Tabucol*, adaptado de [31].

REACT-TABUCOL($G = (V, E)$, k, $s_0 = \{C_1, ..., C_k\}$)

```
1  s = s_0;
2  s* = s;
3  lista_tabu = {};
4  inicializar tl de acordo com o esquema reativo;
5  while não atingiu critérios de parada do
6      escolher um movimento (v, c) ∉ lista_tabu, com o menor valor de δ(v, c);
                                          // onde δ(v, c) = f(s ∪ (v, c)) − f(s)
7      s = (s ∪ (v, c))\(v, s(v));
8      determinar tl de acordo com o esquema reativo;
9      lista_tabu = lista_tabu ∪ {(v, s(v))};        // por tl iterações
10     if f(s) < f(s*) then
11         s* = s;
12 return s*;
```

6 O IMPACTO DA PARAMETRIZAÇÃO

Com o objetivo de avaliar o impacto dos parâmetros passados ao algoritmo *ColorAnt*$_3$*-RT* e realizar um estudo comparativo entre o desempenho de colônia de formigas e busca local para este algoritmo foram realizados alguns experimentos, os quais foram executados em um computador Intel Xeon E5504 de 2.00 GHz, 24GB de memória RAM e sistema operacional Ubuntu 10.04.3 LTS com Kernel 2.6.32-37-server. Além disto, *ColorAnt*$_3$*-RT* foi implementado na linguagem `C` e compiladas com GCC 4.4.3 utilizando o nível de otimização `O3`. Tais experimentos foram realizados em 14 grafos do Desafio DIMACS[1] [17], a saber:

- `dsjc250.1`, `dsjc250.5`, `dsjc500.1`, `dsjc500.5`, `dsjc1000.1`: grafos aleatórios padrão `dsjc`$n.d$ que possuem n vértices e d probabilidade de quaisquer dois vértices formarem uma aresta;
- `dsjr500.1`, `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_26_0` e `flat300_28_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.

A Tabela 1 apresenta as características destes 14 grafos. Nesta tabela, a primeira coluna apresenta o nome do grafo. A segunda o par χ/k^*, onde '?' indica que χ não é conhecido e k^* indica o valor da melhor aproximação da coloração encontrada até o momento. A terceira coluna apresenta a quantidade de vértices do grafo. A quarta apresenta a quantidade de arestas do grafo. E por fim, a quinta coluna apresenta a densidade do grafo.

Tabela 1: Características das instâncias utilizadas nos experimentos.

Grafo	χ/k^*	*Vértices*	*Arestas*	*Densidade*
dsjc250.1	?/8	250	3218	0,103
dsjc250.5	?/28	250	15668	0,503
dsjc500.1	?/12	500	12458	0,099
dsjc500.5	?/48	500	62624	0,502
dsjc1000.1	?/20	1000	49629	0,099
dsjr500.1	?/12	500	3555	0,028
dsjr500.1c	?/85	500	121275	0,972
dsjr500.5	?/122	500	58862	0,472
flat300_26_0	26/26	300	21633	0,483
flat300_28_0	28/28	300	21695	0,484
le450_15c	15/15	450	16680	0,165
le450_15d	15/15	450	16750	0,166
le450_25c	25/25	450	17343	0,172
le450_25d	25/25	450	17425	0,172

Os experimentos realizados foram divididos em 2 grupos, a saber:

[1]Disponível em `http://mat.gsia.cmu.edu/COLOR/instances.html`, acessado em junho de 2012

1. Aqueles cujos parâmetros foram especificados por uma calibragem simples (Experimento 1); e

2. Aqueles cujos parâmetros foram especificados por uma calibragem dirigida e mais agressiva (Experimento 2).

O Experimento 1 tem por objetivo verificar se uma calibragem simples é capaz de obter resultados próximos as melhores aproximações apresentadas na literatura. O Experimento 2 tem por objetivo avaliar o impacto da tamanho da colônia de formigas, como também o impacto da quantidade de ciclos utilizado pela busca local. Por sua vez cada experimento é composto por duas partes:

1. A primeira é composta por um conjunto de experimentos com o objetivo de encontrar os melhores valores dos parâmetros de *ColorAnt*$_3$*-RT*, em outra palavras, com o objetivo de calibrar *ColorAnt*$_3$*-RT*; e

2. A segunda é composta por um conjunto de experimentos com o objetivo de avaliar o desempenho de *ColorAnt*$_3$*-RT*, no tocante a melhor aproximação por ele encontrada, como também ao seu tempo de execução.

Embora em cada experimento *ColorAnt*$_3$*-RT* possua parâmetros distintos, em ambos *ColorAnt*$_3$*-RT* finaliza sua execução assim que uma solução própria for encontrada ou quando o tempo máximo de execução de 1 hora for atingido (esta é a execução padrão). Além disto, cada instância foi executada 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.

6.1 EXPERIMENTO 1: UMA CALIBRAGEM SIMPLES

Um ponto importante na execução de algoritmos heurísticos é a calibragem de seus parâmetros. Desta forma, vários experimentos foram realizados com o objetivo de encontrar possíveis valores para α, β, ρ, *o tamanho da colônia de formigas* e *a quantidade de ciclos de busca local*. Uma estratégia simples é calibrar cada parâmetro de forma independente. Embora esta estratégia possa parecer incorreta pelo fato de existir relacionamento entre os parâmetros, ela foi utilizada com o objetivo de verificar a qualidade dos resultados a partir de uma calibragem que não leva em consideração tal relacionamento. Inicialmente foram calibrados α e β, utilizando o Algoritmo 4.

Algoritmo 4 Calibre_α_β

CALIBREα_β($G = (V, E)$, k^*) // V: vértices; E: arestas

```
1  for α = 1 to 20 do
2      for β = 1 to 20 do
3          ρ = 0.5;
4          formigas = 50;
5          ciclos = 50;
6          bl_ciclos = 0;
7          for tentativas = 1 to 3 do
8              COLORANT3-RT(G, k*, α, β, ρ, formigas, ciclos, bl_ciclos);
9      Armazene a quantidade média de conflitos para esta configuração
10 Retorne a configuração com a menor quantidade de conflitos
```

Como indicado no Algoritmo 4 foi utilizada uma colônia pequena sem o uso de busca local, além disto ρ foi calibrado impiricamente. Uma questão que deve ser observada não apenas na calibragem de α e β, mas também na calibragem dos outros parâmetros é o fato desta definir uma quantidade máxima de ciclos para o algoritmo. Desta forma, é possível que *ColorAnt*$_3$*-RT* finalize antes de atingir o tempo máximo de 1 hora.

Após a calibragem de α e β o próximo passo foi calibrar ρ. Basicamente foi utilizada a mesma estratégia anterior, como pode ser observado no Algoritmo 5, sendo α e β valores impíricos.

Algoritmo 5 Calibre_ρ

CALIBREρ($G = (V, E)$, k^*) // V: vértices; E: arestas

```
1  for ρ = 0.0 to 1.0 do
2      α = 3;
3      β = 5;
4      formigas = 50;
5      ciclos = 50;
6      bl_ciclos = 0;
7      for tentativas = 1 to 3 do
8          COLORANT3-RT(G, k*, α, β, ρ, formigas, ciclos, bl_ciclos);
9      Armazene a quantidade média de conflitos para esta configuração
10 Retorne a configuração com a menor quantidade de conflitos
```

Por fim, foram calibrados *o tamanho da colônia de formigas* e *a quantidade de ciclos de busca local*. A estratégia utilizada é a mesma apresentada no Algoritmo 5, com a diferença que o valor de ρ foi fixado em 0.5, o tamanho da colônia variou entre 20 e 500 e a quantidade de busca local variou entre 50 e 2000. Note que a calibragem foi independente, portanto o tamanho da colônia e a quantidade de ciclos de busca local foram estimados separadamente. Os resultados obtidos pelas calibragens são apresentados na Tabela 2.

Tabela 2: Resultados obtidos pela calibragem de α, β, ρ, *tamanho da colônia de formigas* e *quantidade de ciclos de busca local*, para o Experimento 1.

Instância		Parâmetros				
Grafo	χ/k^*	$formigas$	α	β	ρ	bl_ciclos
dsjc250.1	?/8	20	2	1	0.1	1550
dsjc250.5	?/28	20	2	11	0.9	1600
dsjc500.1	?/12	360	3	6	0.1	1950
dsjc500.5	?/48	200	3	10	0.2	1900
dsjc1000.1	?/20	20	3	10	0.0	1950
dsjr500.1	?/12	20	1	1	0.0	50
dsjr500.1c	?/85	280	4	8	0.6	1350
dsjr500.5	?/122	180	1	20	1.0	600
flat300_26_0	26/26	60	4	5	0.9	1550
flat300_28_0	28/28	100	2	7	0.4	1550
le450_15c	15/15	260	19	5	0.5	1950
le450_15d	15/15	220	12	6	0.0	1700
le450_25c	25/25	40	2	20	0.3	1250
le450_25d	25/25	40	2	8	0.7	1750

Os resultados desta calibragem indicam que os melhores resultados foram obtidos levando em consideração a informação heurística, isto para a maioria das instâncias. Apenas para `dsjc250.1`, `le450_15c` e `le450_15d` os melhores resultados foram obtidos considerando a trilha de feromônio. Além disto, as instâncias `dsjc1000.1`, `dsjr500.1` e `le450_15d` foram calibradas com o valor $\rho = 0.0$, como mostra a Tabela 2. Assim, a cada iteração as formigas constroem suas soluções com base no feromônio decorrente apenas do depósito realizado na iteração imediatamente anterior, que pode ter sido feito utilizando s' ou s^*. Desse modo, o valor 0 é atribuído a todas as posições da matriz antes do depósito de feromônio em cada iteração, fazendo com que o feromônio referente a inicialização e histórico de atualizações influenciem as decisões das formigas na iteração atual apenas por meio do que já influenciaram na construção de s^* e s'. Neste caso, há um grande favorecimento à exploração do espaço de busca. Tal exploração se mostrou benéfica, como poderá ser visto nos resultados apresentados na próxima seção, ocasionando 0% de distância para as melhores soluções, exceto para `dsjc1000.1`. Portanto, mesmo a calibragem do parâmetro ρ sendo diferente do que normalmente ocorre para algoritmos ACO, bons resultados podem ser alcançados.

6.1.1 DESEMPENHO DE *COLORANT$_3$-RT*

A Tabela 3 apresenta os resultados obtidos por *ColorAnt$_3$-RT*, com os parâmetros mencionados na seção anterior. Nesta tabela, a primeira coluna apresenta o nome do grafo e a segunda o par χ/k^*. A terceira coluna apresenta o melhor valor de k que foram encontrados. E por fim, a quarta coluna apresenta o par `S`/`T`: número de execuções bem sucedidas (`S`) e o total de execuções (`T`).

Os resultados apresentados na Tabela 3 demonstram que para esta parametrização, *ColorAnt$_3$-RT* possui os melhores resultados para grafos *geométricos*. Para tais grafos, *ColorAnt$_3$-RT* encontrou as melhores aproximações já encontradas até o momento. Embora não tenha garantido uma quantidade de sucessos significativa. Para os grafos *le450* os resultados se distanciam 2%, das melhores aproximações conhecidas. Enquanto que, para os grafos *aleatórios* e *flat ColorAnt$_3$-RT* obteve resultados que mais se distanciam das melhores aproximações conhecidas, em respectivamente 4,92% e 14,84%. Contudo, vale ressaltar o fato de *ColorAnt$_3$-RT* ser capaz de encontrar as melhores aproximações para a maioria das instâncias avaliadas, mesmo para uma configuração que não utilize uma calibragem agressiva, como aquela que será descrita na Seção 6.2.

Analisando *ColorAnt$_3$-RT* de um prisma diferente, a partir da quantidade de vértices do grafo, é possível perceber que aumentar a quantidade de vértices ocasionou uma perda de desempenho na qualidade dos resultados. Na média, os resultados seguem a seguinte distribuição de acordo com a quantidade de vértices dos grafos: 250 (0% de distância das melhores aproximações), 450 (2%), 500 (2,92%) e 1000 (10%). Isto para uma avalição que exclua os grafos *flat*, para os quais *ColorAnt$_3$-RT* obteve o pior desempenho, sendo 14,84% a distância média destes para as melhores aproximações.

No geral a execução de *ColorAnt$_3$-RT* terminou por encontrar a melhor solução ou por estagnação, em outras palavras por não convergir para uma aproximação melhor. Portanto, para uma situação na qual *ColorAnt$_3$-RT* não obteve a quantidade máxima de sucessos (10), ou ainda não conseguiu melhorar a qualidade das aproximações encontradas houve uma estagnação da execução. Apenas para duas execuções da instância `dsjc1000.1` *ColorAnt$_3$-RT* finalizou pelo fato de atingir o tempo máximo de execução, na tentativa de melhorar a qualidade da aproximação.

Tabela 3: Resultados obtidos por *ColorAnt*$_3$*-RT* no Experimento 1.

Instância		*ColorAnt*$_3$*-RT*	
Grafo	χ/k^*	k	S/T
dsjc250.1	?/8	**8**	10/10
dsjc250.5	?/28	**28**	2/10
		29	10/10
dsjc500.1	?/12	13	10/10
dsjc500.5	?/48	51	3/10
	?/48	52	9/10
	?/48	53	10/10
dsjc1000.1	?/20	22	9/10
		23	10/10
dsjr500.1	?/12	**12**	10/10
dsjr500.1c	?/85	**85**	9/10
		86	10/10
dsjr500.5	?/122	**122**	1/10
		123	9/10
		124	10/10
flat300_26_0	26/26	30	1/10
		31	1/10
		32	10/10
flat300_28_0	28/28	32	6/10
		33	10/10
le450_15c	15/15	**15**	8/10
		16	10/10
le450_15d	15/15	**15**	10/10
le450_25c	25/25	**26**	2/10
		27	10/10
le450_25d	25/25	**26**	2/10
		27	10/10

Embora, estes resultados demonstrem que um aumento gradativo na quantidade de vértices tende ocasionar uma perda de desempenho, uma questão que deve ser levada em consideração é a característica de cada instância. Ao observar tais características como apresentadas na Tabela 1 é possível perceber que embora a quantidade de vértices seja a mesma para algumas instâncias, *ColorAnt*$_3$*-RT* obteve diferente desempenho para instâncias com a mesma quantidade de vértices. Isto pode indicar que provavelmente não seja uma boa alternativa analisar as instâncias apenas pela quantidade de vértices, mas sim pelo conjunto de suas características. Uma questão que estes experimentos deixaram claro, pelo menos para o conjunto de instâncias avaliado, é o fato de não existir uma correlação direta entre densidade do grafo e qualidade dos resultados. Uma análise das informações apresentadas nas Tabelas 1 e 3 demonstra que *ColorAnt*$_3$*-RT* foi capaz de encontrar as melhores soluções tanto para grafos densos, quanto para aquelas com baixa densidade. Contudo, este obteve os piores resultados para os grafos com densidade média.

6.2 EXPERIMENTO 2: UMA CALIBRAGEM MAIS AGRESSIVA

Neste novo grupo de experimentos a estratégia de calibragem de *ColorAnt*$_3$*-RT* para cada instância foi alterada, com o objetivo de verificar se valores diferentes na parametrização ocasiona um ganho de desempenho. A estratégia utilizada neste novo experimento para calibrar *ColorAnt*$_3$*-RT* foi identificar qual fator ocasiona um melhor resultado: a trilha de feromônio (α) ou a informação heurística (β). Nesta nova estratégia, a medida que o peso da trilha de feromônio aumenta ocorre um aumento no fator de persistência de feromônio (ρ) e consequentemente uma redução no peso da informação heurística. Além disto, esta calibragem utilizou uma colônia pequena sem o uso de ciclos de busca local e a finalização padrão de *ColorAnt*$_3$*-RT* (finalizar quando encontrar uma solução própria ou quando o tempo de execução atingir 1 hora). Isto para encontrar a melhor configuração baseada apenas em colônia de formigas, em outras palavras, em uma solução construtiva. Portanto, nesta calibragem apenas os valores de α, β e ρ foram estimados. A calibragem realizada neste segundo experimento é como especificada no Algoritmo 6 e os melhores valores encontrados para α, β e ρ são apresentados na Tabela 4.

Como pode ser observado nos resultados apresentados na Tabela 4, apenas para a instância `flat_300_28_0` utilizar a trilha de feromônio como prioridade proporcionou a menor quantidade de conflitos. Para as outras instâncias a melhor estratégia foi utilizar a informação heurística. Portanto, os valores apresentados na Tabela 4 são os utilizados no Experimento 2.

Algoritmo 6 Calibre_α_β_ρ

CALIBREα_β_ρ($G = (V, E)$, k^*) // V: vértices; E: arestas

```
1  for x = 1 to 9 do
2      α = x;
3      β = 10 − x;
4      ρ = x/10;
5      formigas = 100;
6      bl_ciclos = 0;
7      for tentativas = 1 to 10 do
8          COLORANT3-RT(G, k*, α, β, ρ, formigas, bl_ciclos);
9      Armazene a quantidade média de conflitos para esta configuração
10 Retorne a configuração com a menor quantidade de conflitos
```

Tabela 4: Resultados obtidos pela calibragem de α, β e ρ, para o Experimento 2.

Instância		Parâmetros		
Grafo	(χ/k^*)	α	β	ρ
dsjc250.1	?/8	1	9	0.1
dsjc250.5	?/28	1	9	0.1
dsjc500.1	?/12	1	9	0.1
dsjc500.5	?/48	3	7	0.3
dsjc1000.1	?/20	1	9	0.1
dsjr500.1	?/12	2	8	0.2
dsjr500.1c	?/85	4	6	0.4
dsjr500.5	?/122	1	9	0.1
flat300_26_0	26/26	4	6	0.4
flat300_28_0	28/28	6	4	0.6
le450_15c	15/15	1	9	0.1
le450_15d	15/15	1	9	0.1
le450_25c	25/25	1	9	0.1
le450_25d	25/25	1	9	0.1

Os experimentos de calibragem de *ColorAnt$_3$-RT* até este momento para o Experimento 2, não levaram em consideração o tamanho da colônia de formigas nem a quantidade necessária de ciclos de busca local. Isto devido ao fato do Experimento 2 ter como principal objetivo avaliar o impacto do tamanho da colônia de formigas versus a quantidade de ciclos de busca local. Para alcançar este objetivo foram realizados diversos experimentos utilizando os parâmetros apresentados na Tabela 4. No Experimento 2 tanto o tamanho da colônia de formigas quanto a quantidade de ciclos de busca local variou entre 10 e 1.000.000, aumentado a cada experimento o valor anterior em uma ordem de grandeza. Além disto, as configurações com diferentes tamanhos de colônias não utilizaram nenhum ciclo de busca local. Por outro lado, as configurações com diferentes quantidades de ciclos de busca local utilizaram a menor colônia possível. Vale ressaltar que pelo fato de *ColorAnt$_3$-RT* ser um algoritmo ACO é possível não utilizar busca local, contudo o inverso não é verdadeiro.

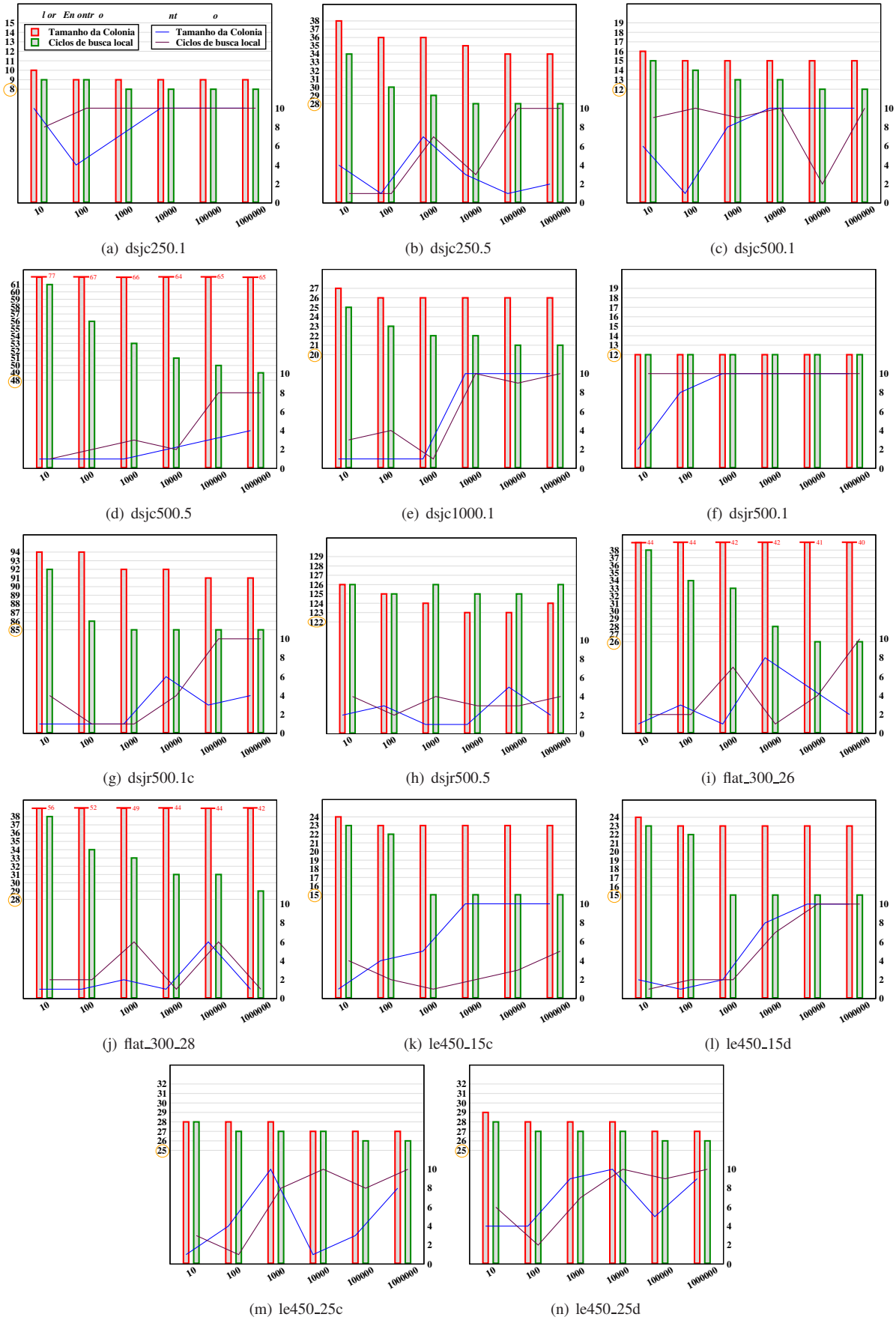
6.2.1 DESEMPENHO DE *COLORANT$_3$-RT*

Devido ao fato do Experimento 2 variar a grandeza do tamanho da colônia de formigas como também da quantidade de ciclos de busca local, nesta seção são apresentados os resultados para distintas configurações.

Os gráficos da Figura 1 apresentam as melhores aproximações encontradas para cada instância. Em cada gráfico são apresentadas duas barras para a mesma grandeza, a barra da esquerda representa o tamanho da colônia de formigas enquanto a da direita a quantidade de ciclos de busca local. Cada gráfico também apresenta a quantidade de sucessos obtida por cada configuração (gráfico de linhas), para o melhor valor de k encontrado. Além disto, sobre cada conjunto de barras é apresentado um par contendo o valor de k para o qual a configuração obteve a quantidade máxima de sucessos, em outras palavras encontrou nas 10 tentativas o valor desejado de k.

Como pode ser observado pelos resultados apresentados na Figura 1 o uso de busca local com o menor tamanho de colônia foi a configuração que obtive as melhores aproximações de k para a maioria das instâncias. Para sete instâncias (`dsjc250.1`, `dsjc250.5`, `dsjr500.1`, `dsjr500.1c`, `flat300_26_0`, `le450_15c`, `le450_15d`) foram encontradas as melhores aproximações conhecidas. E em cinco instâncias (`dsjc500.5`, `dsjc1000.1`, `flat300_28_0`, `le450_25c`, `le450_25d`) a distância entre a aproximação encontrada e as melhores conhecidas foi de apenas uma unidade. O uso de busca local não melhorou as soluções apenas para `dsjr500.1` e `dsjr500.5`, para estas instâncias o uso de apenas uma abordagem construtiva foi a melhor solução. Isto pode ser explicado para a primeira instância pelo fato desta ser uma instância simples, sendo a que

Figura 1: Qualidade das aproximações obtidas no Experimento 2.



(a) dsjc250.1 (b) dsjc250.5 (c) dsjc500.1

(d) dsjc500.5 (e) dsjc1000.1 (f) dsjr500.1

(g) dsjr500.1c (h) dsjr500.5 (i) flat_300_26

(j) flat_300_28 (k) le450_15c (l) le450_15d

(m) le450_25c (n) le450_25d

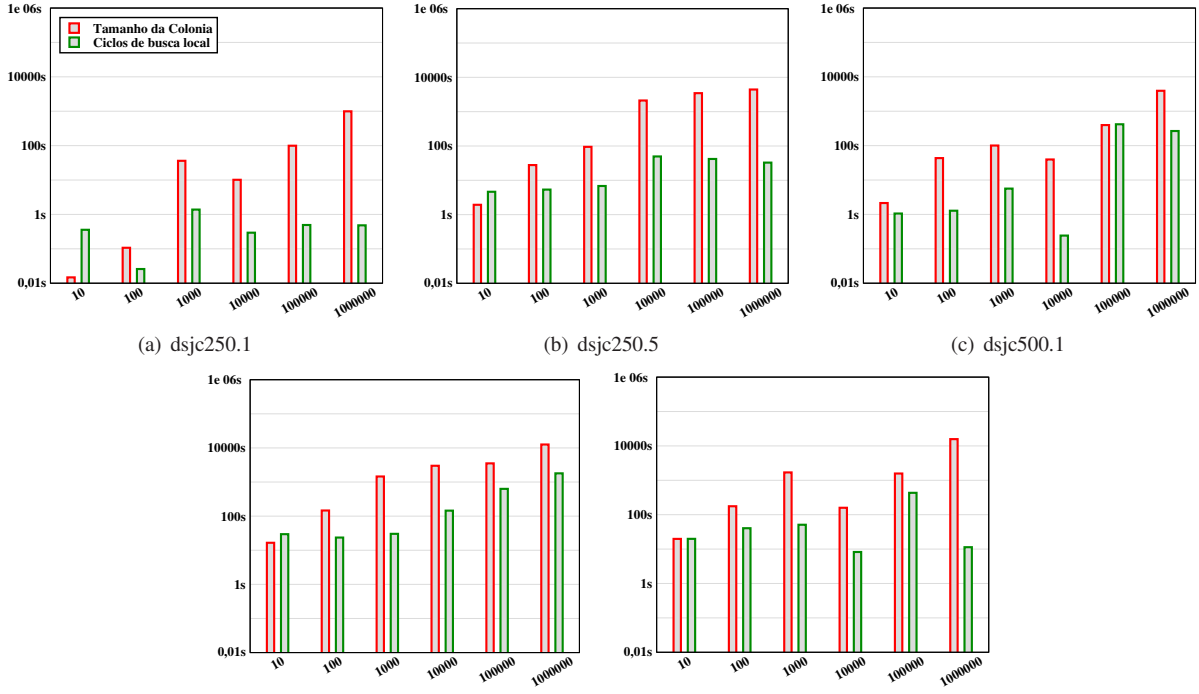
possui a menor densidade entre as instâncias avaliadas. Por outro lado, é difícil explicar esta situação para a segunda instância pois existem outras instâncias com densidades maiores do que ela, para as quais a busca local encontrou as melhores ou boas aproximações.

Estes resultados demonstram que utilizar apenas colônia de formigas não é uma boa opção. De fato mesmo utilizando colônias de tamanhos consideráveis os resultados encontrados se apresentam distantes das melhores aproximações conhecidas, isto pelo menos no contexto de *ColorAnt*$_3$*-RT*. A melhor configuração é aquela que possui uma colônia pequena, mas utiliza uma quantidade considerável de ciclos de busca local. Como pode ser observado nos resultados apresentados na Figura 1, a medida que ocorre um aumento na quantidade de ciclos de busca local a tendência é ocorrer uma melhora da qualidade das aproximações. Neste sentido, as melhores aproximações foram encontradas para uma quantidade de ciclos de busca local igual ou superior a 10.000, para a maioria dos casos.

Outra consequência do uso de busca local é o aumento da quantidade de tentativas com sucesso para a melhor aproximação. Como pode ser observado nos resultados apresentados nesta seção, isto não ocorre ao utilizar apenas colônia de formigas, independente do seu tamanho.

Os gráficos das Figuras 2 e 3 apresentam os tempos de execução[2] (em escala logarítmica) de cada instância, para as melhores aproximações conhecidas. Assim como os gráficos apresentados na Figura 1, em cada gráfico são apresentadas duas barras para a mesma grandeza, a barra da esquerda representa o tamanho da colônia de formigas enquanto a da direita a quantidade de ciclos de busca local. Além disto, sobre cada conjunto de barras é apresentado um par contendo o valor médio de ciclos necessários por *ColorAnt*$_3$*-RT* para cada configuração.

Figura 2: Tempos de execução obtidos no Experimento 2, para os grafos *aleatórios*.



(a) dsjc250.1 (b) dsjc250.5 (c) dsjc500.1

(d) dsjc500.5 (e) dsjc1000.1

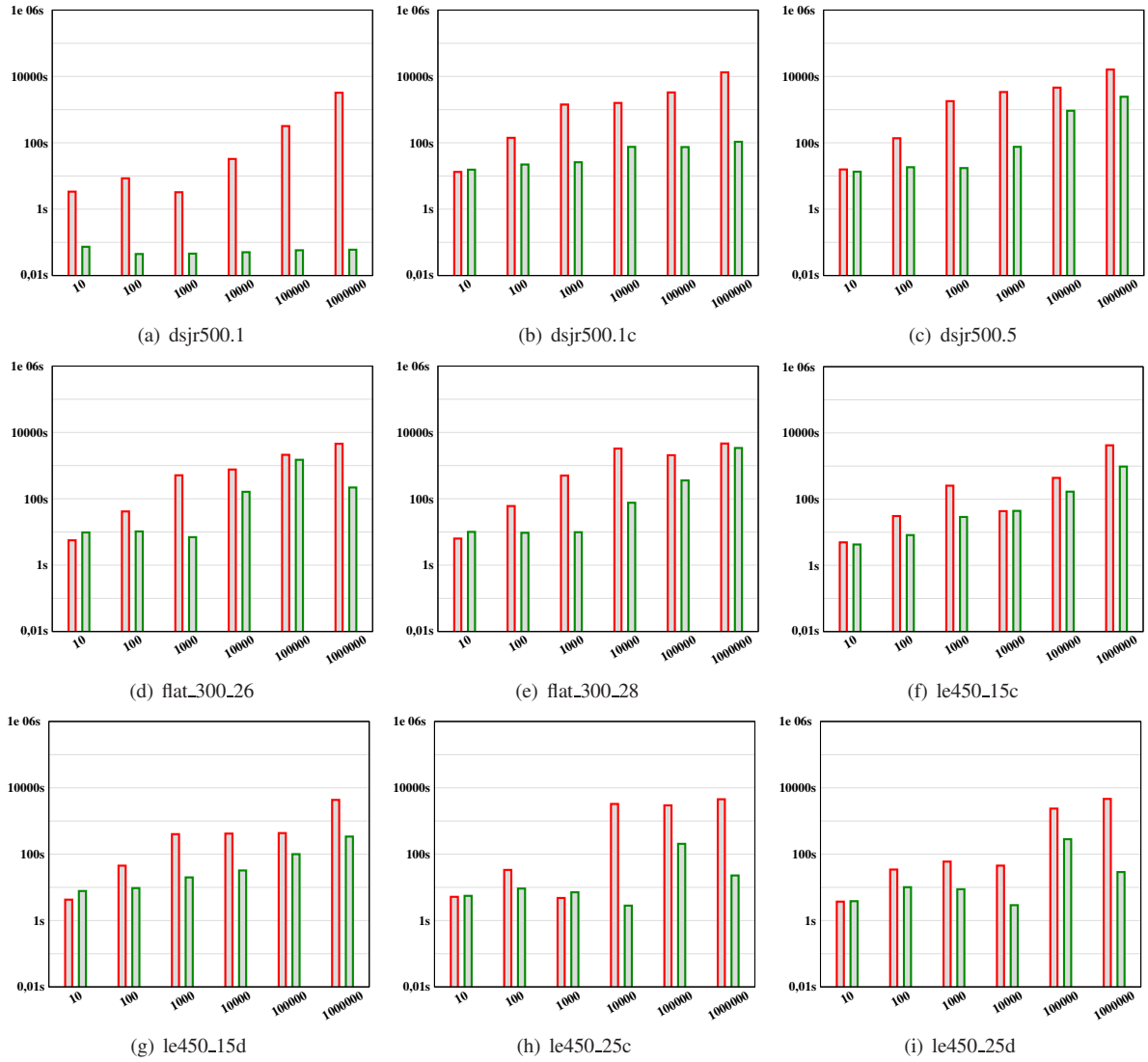
De forma geral é possível perceber que o impacto negativo ocasionado no tempo médio de execução pelo aumento do tamanho da colônia de formigas é maior do que o ocasionado pelo aumento da quantidade de ciclos de busca local. Outra questão que também deve ser observada é o fato de mesmo que ambas configurações (uso apenas de colônia de formigas versus uso de colônia com o auxílio de busca local, para uma mesma ordem de grandeza) tenham um desempenho similar para a melhor aproximação encontrada, a tendência é que o menor tempo médio de execução seja obtido pelo uso de uma colônia pequena com o auxílio de busca local. Portanto, para reduzir o tempo de execução de *ColorAnt*$_3$*-RT* o uso de uma colônia pequena com uma quantidade considerável de ciclos de busca local ainda é a melhor opção. E isto não apenas para uma redução no tempo de execução, mas também para a obtenção de boas aproximações.

Apenas para quatro instâncias o uso de busca local ocasionou um tempo médio de execução maior que 10 minutos para que boas aproximações fossem encontradas, a saber: `dsjc500.5` (29,9 minutos), `dsjr500.5` (76,7 minutos), `flat300_28_0` (56,5 minutos) e `le450_15c` (16 minutos). Isto é um excelente resultado, pelo fato dos melhores algoritmos necessitarem de um tempo médio de execução superior a 1 hora para que boas aproximações sejam encontradas, como é demonstrado na literatura.

Outro fato importante que também pode ser observado nestes resultados é que o aumento do tamanho da colônia de formigas ocasiona uma redução na quantidade de ciclos do algoritmo. Esta situação ocasiona um efeito não desejado, o uso de um único

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

Figura 3: Tempos de execução obtidos no Experimento 2, para os grafos *geométricos*, *flat* e *le450*.



(a) dsjr500.1 (b) dsjr500.1c (c) dsjr500.5

(d) flat_300_26 (e) flat_300_28 (f) le450_15c

(g) le450_15d (h) le450_25c (i) le450_25d

ciclo pelo algoritmo. Isto indica que o algoritmo não levará em consideração a trilha de feromônio, nem a informação heurística durante a busca por uma solução. Esta situação pode ser solucionada pelo aumento do tempo de execução do algoritmo, na tentativa de verificar se boas aproximações podem ser encontradas para tal tamanho de colônia. De fato, um experimento similar foi realizado com *ColorAnt*$_3$*-RT* o qual será discutido na Seção 6.5.

Assim como no Experimento 1, no geral a execução de *ColorAnt*$_3$*-RT* terminou por encontrar a melhor solução ou por estagnação. Com algumas exceções para uma configuração acima de 100.000, a qual terminou pelo fato do tempo de execução exceder o valor máximo de 1 hora.

6.3 EXPERIMENTO 1 VERSUS EXPERIMENTO 2

A Tabela 5 apresenta uma síntese dos resultados obtidos nos dois experimentos. Nesta tabela, a primeira coluna apresenta o nome do grafo. A segunda apresenta o par χ/k^*. A terceira coluna apresenta o melhor valor de k que foi encontrado. A quarta coluna apresenta o par `S/T`. A quinta coluna apresenta o tempo médio até a melhor solução encontrada[3], em segundos. A sexta coluna apresenta a quantidade média de ciclos. A sétima coluna apresenta o número médio de conflitos de cada instância. Por fim, as próximas colunas apresentam as mesmas informações porém para os resultados obtidos pelo Experimento 1. Além disto, esta tabela apresenta apenas as melhores soluções encontradas.

Os dados apresentados na Tabela 5 demonstram que a *qualidade* dos parâmetros influência diretamente a qualidade dos resultados obtidos. A parametrização realizada no Experimento 2 ocasionou um ganho de desempenho a *ColorAnt*$_3$*-RT*, não apenas na qualidade das aproximações, mas também no tempo de execução, quantidade de ciclos necessária pela algoritmo e quantidade de conflitos.

De fato a parametrização realizada no Experimento 2 proporcionou uma melhora no valor de k para cinco instâncias, a saber: `dsjc500.5`, `dsjc1000.1`, `flat_300_28`, `dsjc500.1` e `flat_300_26`. Sendo que para as duas últimas, a nova

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

Tabela 5: Resultados obtidos por *ColorAnt*$_3$*-RT*, com diferentes calibragens.

Instância		*ColorAnt*$_3$*-RT* (Experimento 2)					*ColorAnt*$_3$*-RT* (Experimento 1)				
Grafo	χ/k^*	k	S/T	Tempo (s)	Ciclos	Conflitos	k	S/T	Tempo (s)	Ciclos	Conflitos
dsjc250.1	?/8	**8**	10/10	1,381	62,1	0	**8**	10/10	6,426	17,2	0
dsjc250.5	?/28	**28**	10/10	41,947	13,9	0	**28**	2/10	629,074	151,8	2,6
dsjc500.1	?/12	**12**	10/10	266,385	16,4	0	13	10/10	44,109	15,0	0
dsjc500.5	?/48	49	8/10	1799,525	29,6	0,9	51	3/10	1690,425	194,3	2,8
dsjc1000.1	?/20	21	10/10	11,361	1	0	22	9/10	426,408	28,4	0,3
dsjr500.1	?/12	**12**	10/10	0,073	1,8	0	**12**	10/10	3,057	40,9	0
dsjr500.1c	?/85	**85**	10/10	74,381	12,8	0	**85**	9/10	29,494	20,5	0,1
dsjr500.5	?/122	123	5/10	4603,312	3	0,5	**122**	1/10	625,379	152,9	1,5
flat300_26_0	26/26	**26**	10/10	219,358	2,1	0	30	1/10	1043,777	176,9	8,3
flat300_28_0	28/28	29	1/10	3392,849	69,2	10,4	32	6/10	434,151	99,7	1,1
le450_15c	15/15	**15**	5/10	964,004	76,4	1,4	**15**	8/10	222,446	58,4	1,4
le450_15d	15/15	**15**	10/10	100,301	36,4	0	**15**	10/10	168,655	14,9	0
le450_25c	25/25	26	10/10	22,859	1	0	26	2/10	456,198	202,7	3,3
le450_25d	25/25	26	10/10	28,918	1,2	0	26	2/10	724,874	176,6	3,4

parametrização ocasionou a obtenção da melhor aproximação conhecida. Embora para outras instâncias o valor de k tenha se mantido estável, a nova parametrização teve um efeito positivo no aumento da quantidade de sucessos. Isto é um fator positivo, pelo fato de fornecer estabilidade ao algoritmo. Embora, pelas características dos algoritmos ACO não existam garantias que um determinado resultado será reproduzido em futuras execuções.

Esta comparação entre os dois experimentos também demonstra que é possível obter bons resultados em um tempo de execução reduzido, a medida que parâmetros adequados sejam estimados. Como consequência desta escolha, a quantidade de ciclos necessários pelo algoritmo é reduzida como também a quantidade média de conflitos.

Como pode ser observado nos resultados apresentados na Tabela 5, no geral o Experimento 2 obteve um tempo de execução menor do que aquele obtido pelo Experimento 1. A redução de tempo variou entre 40,53% a 97,61%, excluindo as instâncias para as quais o Experimento 2 ocasionou um aumento no tempo de execução, a saber: `dsjc500.1`, `dsjc500.5`, `flat_300_28`, `dsjr500.1c`, `le450_15c` e `dsjr500.5`. Sendo que para as três primeira, embora o tempo de execução tenha aumentado o Experimento 2 ocasionou uma melhora na qualidade da aproximações encontradas. Para as próximas duas instâncias não houve uma melhora nas aproximações, para `dsjr500.1c` houve um aumento na quantidade de sucessos, porém para `le450_15c` houve uma redução nesta quantidade. Por fim, para a instância `dsjr500.5` houve uma perda na qualidade da aproximação.

Uma questão importante a ser observada é o fato dos resultados apresentados na Tabela 5 para Experimento 2 serem obtidos pela menor colônia (10 formigas) e diferentes quantidades de ciclos de busca local. A única exceção a esta regra é a instância `dsjr500.5`, para a qual o melhor resultado foi obtido pela configuração com 100.000 formigas, sem o uso de ciclos de busca local. Outro ponto a ser observado é o fato das instâncias necessitarem de grandezas diferentes quanto a quantidade de ciclos de busca local. Neste caso, `dsjr500.1` necessitou apenas de 10 ciclos; `dsjc250.1` de 1.000 ciclos; `dsjc250.5`, `dsjr500.1c` e `le450_15d` necessitaram cada uma de 100.000 ciclos; enquanto as outras instâncias necessitaram de 1.000.000 de ciclos de busca local. Comparativamente com os resultados obtidos no Experimento 1, isto corrobora o que foi mencionado na Seção 6.3: para *ColorAnt*$_3$*-RT* uma estratégia melhorativa proporciona melhores resultados do que aqueles obtidos por uma estratégia apenas construtiva.

6.4 *COLORANT*$_3$*-RT* E OUTRAS HEURÍSTICAS

Os resultados obtidos por *ColorAnt*$_3$*-RT* para as instâncias avaliadas 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*$_3$*-RT*. Os valores encontrados por outras heurísticas são os apresentados para uma comparação com o *ALS* [15] e indicam o menor valor de k tal que pelo menos uma k-coloração própria foi encontrada. A Tabela 6 apresenta os resultados obtidos pelos *ColorAnt*$_3$*-RT* e por heurísticas. Os valores nesta tabela aparecem em negrito quando χ ou k^* foram atingidos.

Como em *ColorAnt*$_3$*-RT*, o *ALS* também utilizou tempo máximo de execução de 1 hora. O algoritmo *MMT* teve tempo limite de 100 minutos [15], mas nenhuma menção é feita com relação aos tempos dos outros algoritmos.

Como pode ser observado pelos valores apresentados na Tabela 6, uma calibragem adequada proporciona um ganho de desempenho significativo, ocasionando a *ColorAnt*$_3$*-RT* um desempenho similar a outras heurísticas. É importante observar que apenas para quatro instâncias (`dsjc500.5`, `dsjc1000.1`, `le450_25c` e `le450_25d`) as aproximações são distantes, contudo apenas de uma unidade.

A heurística *MMT*, conforme mecionado, possui um tempo máximo de execução (100min) mais elevado do que o tempo máximo de execução apresentado por *ColorAnt*$_3$*-RT* (60min), ressaltando que ambos foram executados em plataformas (sistema

Tabela 6: Valores de k para *ColorAnt*$_3$-RT, *ALS-COL*, *Tabucol*, *MMT*, *GH* e *MOR*.

Instância		*ColorAnt*$_3$*-RT*		Outras Heurísticas				
Grafo	χ/k^*	Experimento 2	Experimento 1	*ALS-COL*	*TabuCol*	*MMT*	*GH*	*MOR*
dsjc250.1	?/8	**8**	**8**	-	-	-	-	-
dsjc250.5	?/28	**28**	**28**	-	-	-	-	-
dsjc500.1	?/12	**12**	13	**12**	**12**	**12**	**12**	**12**
dsjc500.5	?/48	49	51	**48**	49	**48**	**48**	49
dsjc1000.1	?/20	21	22	**20**	**20**	**20**	**20**	21
dsjr500.1	?/12	**12**	**12**	-	-	-	-	-
dsjr500.1c	?/85	**85**	**85**	**85**	**85**	**85**	-	**85**
dsjr500.5	?/122	123	**122**	125	126	**122**	-	123
flat300_26_0	26/26	**26**	30	-	-	-	-	-
flat300_28_0	28/28	29	32	29	31	31	31	31
le450_15c	15/15	**15**	**15**	**15**	16	**15**	**15**	**15**
le450_15d	15/15	**15**	**15**	**15**	**15**	**15**	**15**	**15**
le450_25c	25/25	26	26	26	26	**25**	26	**25**
le450_25d	25/25	26	26	26	26	**25**	26	**25**

operacional e *hardware*) diferentes. Assim, embora *ColorAnt*$_3$*-RT* obtenha aproximações piores do que aquelas obtidas por *MMT* e como ambos podem ter tempo total de execução menor do que o máximo (nos casos em que uma coloração ótima – sem arestas conflitantes – é encontrada), *ColorAnt*$_3$*-RT* possui um tempo de execução melhor, principalmente nos piores casos. Isto é um atrativo para contextos onde reduzir o tempo de execução é um objetivo a ser alcançado.

6.5 DISCUSSÃO

Os experimentos realizados com *ColorAnt*$_3$*-RT* respondem alguns questões enquanto ocasiona o surgimento de outras, as quais são discutidas a seguir.

Algoritmo Construtivo Versus Algoritmo Melhorativo Pelo menos no contexto de *ColorAnt-RT* o ganho de desempenho está relacionado ao uso de um algoritmo que combine uma estratégia construtiva com uma melhorativa, como demonstrado nos resultados obtidos com tais estratégias.

Estagnação de *ColorAnt*$_3$*-RT* Como demonstrado nos resultados apresentados nas seções anteriores, *ColorAnt*$_3$*-RT* não foi capaz de encontrar as melhores aproximações para seis instâncias. Neste caso, a busca por uma solução melhor levou a uma finalização do algoritmo pelo fato deste não convergir para uma solução melhor. Novos experimentos foram realizados para tais instâncias, com o objetivo de verificar se uma configuração diferente era capaz de melhorar as aproximações encontradas. Estes experimentos levaram em consideração os parâmetros de α, β e ρ encontrados na calibragem realizada no Experimento 2. Contudo, foram alterados o tamanho da colônia e o tempo máximo de execução de *ColorAnt*$_3$*-RT* e isto em três experimentos diferentes. No primeiro foi utilizado uma colônia de 100 formigas, uma quantidade de ciclos de busca local de 1.000.000 e o tempo máximo de execução padrão (1 hora). No segundo foi utilizado a configuração do primeiro, com exceção do tamanho da colônia aumentar uma ordem de grandeza, ou seja foi utilizado uma colônia de tamanho 1000. E por fim, o terceiro utilizou uma configuração com uma colônia de tamanho 10, 1.000.000 de ciclos de busca local e um tempo máximo de execução de 2 horas. Nenhum destes três experimentos melhoraram a qualidade das aproximações para as seis instâncias avaliadas. Isto demonstra que as estratégias até então utilizadas em *ColorAnt*$_3$*-RT* chegaram ao seu limite máximo de desempenho. E que a parametrização realizada no experimento descrito na Seção 6.2 é a melhor possível. Portanto, para melhorar os resultados das seis instâncias em questão é necessário alterar as estratégias utilizadas na implementação de *ColorAnt*$_3$*-RT*.

Calibragem Durante a calibragem de um algoritmo ACO a melhor estratégia é relacionar α, β e ρ. Como demonstrado no Experimento 2, esta relação proporciona os melhores resultados. Contudo, levando em consideração o melhor tamanho de colônia com também a melhor quantidade de ciclos de busca local, para tal relacionamento. Uma parametrização adequada alcança diferentes objetivos, a saber: obtenção das melhores aproximações conhecidas, redução do tempo médio de execução, como também redução da quantidade média de conflitos. Além disto, uma parametrização adequada tende a melhorar a convergência do algoritmo.

Tempo de Calibragem Como mencionado anteriormente, uma questão fundamental em algoritmos ACO é a correta parametrização. Porém, esta tarefa consome um tempo considerável. A parametrização de cada instância no Experimento 2 variou entre 4 horas para a instância `dsjc250.1` e 80 horas para a instância `dsjr500.5`. Para parametrizar todas as instâncias utililizadas foram necessárias 466 horas, as quais não indicam necessariamente 20 dias. Isto pelo fato do processo de

parametrização ter sido paralelizado, já que o hardware utilizado possui 8 núcleos. Embora, fosse possível executar paralelamente 8 parametrizações simultâneas, a abordagem utilizada foi executar apenas 5. Esta decisão foi tomada para reduzir as possíveis influências do sistema nos experimentos. Portanto, foram necessários 3 dias de execuções ininterruptas para finalizar a parametrização. Levando em consideração, todo o tempo de preparação dos experimentos esta quantidade de dias pode dobrar, o que de fato aconteceu. Isto demonstra o quanto tal tarefa é dispendiosa, necessitando de um processo automatizado ou ainda uma estratégia mais elegante a qual elimine a necessidade de tal parametrização, pelo menos para alguns parâmetros. Este é um desafio ainda aberto nesta área. Além deste, existe também o desafio de propor novas estratégias para melhorar a qualidade das aproximações encontradas por *ColorAnt*$_3$*-RT*, o que provavelmente ocasionará o desenvolvimento de um novo algoritmo para a classe *ColorAnt-RT*.

Tamanho da Colônia de Formigas Este tamanho deve ser ajustado de acordo com o tempo máximo de execução de algoritmo. Caso este ajuste não ocorra de forma correta, o algoritmo poderá não utilizar as informações heurísticas a qual se propõe utilizar para encontrar uma solução.

Características da Instância Versus Tamanho da Colônia Ao observar os resultados obtidos no Experimento 2 é possível perceber que não existe uma relação direta entre as características da instância e tamanho da colônia. Isto pelo fato dos melhores resultados serem encontrados com a menor colônia. Portanto, para *ColorAnt*$_3$*-RT* o uso de uma colônia pequena independente da instância analisada é a melhor opção. Isto pelo fato, de minimizar o impacto no tempo de execução.

Características da Instância Versus Quantidade de Ciclos de Busca Local Observando as características de cada instância utilizada nos experimentos (Tabela 1) é possível identificar uma relação entre tais características e a quantidade de ciclos de busca local. Isto considerando os resultados apresentados no Experimento 2, os quais são melhores do que aqueles obtidos por uma parametrização que não leva em consideração a relação entre os parâmetros. No contexto de *ColorAnt*$_3$*-RT* a quantidade de ciclos de busca local pode ser definida por:

$$ciclos = \begin{cases} 10 & \text{se } densidade \leq 0,03 \\ 1000 & \text{se } ((0,03 < densidade \leq 0,103) \text{ E } (vertices \leq 250)) \\ 100000 & \text{se } ((0,103 < densidade \leq 0,503) \text{ E } (vertices \leq 250)) \\ 1000000 & \text{caso contrário} \end{cases}$$

Uma questão importante que deve ser ressaltada é o fato da quantidade de ciclos de busca local expressada pela relação fornecida ser apenas uma aproximação, o que indica que tal valor pode ser melhor ajustado. Isto devido ao fato dos experimentos serem realizados aumentando tal quantidade em uma ordem de grandeza, o que pode ser um aumento considerável. Contudo, observando os resultados do Experimento 2 é possível perceber que o impacto deste aumento no tempo de execução, não é tão considerável quanto ao impacto do aumento do tamanho da colônia. De fato, em várias instâncias este aumento ocasionou a obtenção da melhor aproximação, um aumento na quantidade de tentativas bem sucedidas, como também um aumento na convergência o que resultou em um tempo de execução menor quando comparado com o experimento que utilizou uma ordem de grandeza imediatamente menor. Portanto, embora a quantidade de ciclos de busca local possa ser melhor ajustada, a relação fornecida para a estimativa da quantidade de ciclos de busca local é válida e não fornece uma degradação significativa no desempenho de *ColorAnt*$_3$*-RT*, no tocante ao tempo de execução.

Custo do Tamanho da Colônia Versus Custo da Quantidade de Ciclos de Busca Local O custo do uso de formigas é superior ao custo de ciclos de busca local no contexto de tempo de execução. Embora aumentar a quantidade de ciclos de busca local ocasione um aumento no tempo de execução isto é compensado pela melhora na qualidade das aproximações encontradas. E mesmo que este aumento ocorra, ele ainda é proporcionalmente menor que o tempo ocasionado pelo aumento do tamanho da colônia.

7 CONSIDERAÇÕES FINAIS

ColorAnt$_3$*-RT* tem demonstrado ser um bom algoritmo para solucionar o PCG, sendo capaz de encontrar boas aproximações para diversas instâncias de grafos. A aplicação de *ColorAnt*$_3$*-RT* na resolução do problema de alocação de registradores demonstrou que o uso de uma estratégia baseada em metaheurísticas é capaz de proporcionar resultados melhores do que aqueles obtidos por alocadores de registradores tradicionais, os quais não utilizam metaheurísticas.

Embora *ColorAnt*$_3$*-RT* tenha demonstrado em trabalhos anteriores possuir pontencial, uma questão que estava em aberto era sua análise detalhada com o objetivo de determinar o ponto limite da curva de melhora dos resultados, como também determinar quais são os melhores parâmetros para diferentes instâncias. Este foi o objetivo deste artigo. Portanto, este artigo apresentou uma análise detalhada do algoritmo baseado em colônia de formigas artificiais aplicado ao problema da coloração de grafo, denominado *ColorAnt*$_3$*-RT*.

Como foi demonstrado nos experimentos apresentados, *ColorAnt*$_3$*-RT* foi capaz de encontrar as melhores aproximações conhecidas para diversas instâncias, enquanto que para outras a distância para as melhores aproximações foi de apenas uma unidade. Isto indica que *ColorAnt*$_3$*-RT* está próximo dos melhores algoritmos propostos na literatura.

A análise realizada e apresentada neste artigo indicou que para *ColorAnt*$_3$*-RT* os melhores resultados são alcançados pelo uso de colônia de formigas com o auxílio de busca local, com o objetivo de melhorar a qualidade das soluções encontradas. Além

disto, é necessário uma calibragem adequada do algoritmo. Por fim, a análise demonstrou que existe um ponto limite na curva de melhora dos resultados. O qual não pode ser ultrapassado aumentado o tamanho da colônia, a quantidade de ciclos de busca local ou ainda o limite máximo de tempo de execução. Portanto, existe a necessidade da implementação de novas estratégias para que resultados melhores sejam encontrados.

Por fim, outras questões que também devem ser abordadas em novos trabalhos é a necessidade de implementar outras heurísticas para avaliá-las (e compará-las) sobre as mesmas condições e projetar um novo algoritmo para a classe *ColorAnt-RT* que seja auto-adaptável às características da instância de entrada, de forma que os parâmetros sejam calibrados automaticamente.

REFERÊNCIAS

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

[2] 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.

[3] T. Stützle and H. H. Hoos. "MAX-MIN Ant system". *Future Generation Computer System*, vol. 16, no. 9, pp. 889–914, June 2000.

[4] C.-C. Lin and L.-C. Tseng. "Website Reorganization Using an Ant Colony System". *Expert Systems with Applications*, vol. 37, no. 12, pp. 7598–7605, December 2010.

[5] T. B. Kurniawan, N. K. Khalid, Z. Ibrahim, M. Khalid and M. Middendorf. "An Ant Colony System for DNA sequence design based on thermodynamics". In *Proceedings of the International Conference on Advances in Computer Science and Technology*, pp. 144–149, Anaheim, CA, USA, 2008. ACTA Press.

[6] J. Yin and W. Xiang. "Ant Colony Algorithm for Surgery Scheduling Problem". In *Proceedings of the Third International Conference on Advances in Swarm Intelligence - Volume Part I*, pp. 198–205, Berlin, Heidelberg, 2012. Springer-Verlag.

[7] N. B. Sariff and N. Buniyamin. "Ant Colony System for Robot Path Planning in Global Static Environment". In *Proceedings of the International Conference on System Science and Simulation in Engineering*, pp. 192–197, Stevens Point, Wisconsin, USA, 2010. World Scientific and Engineering Academy and Society.

[8] M. Bessedik, R. Laib, A. Boulmerka and H. Drias. "Ant Colony System for Graph Coloring Problem". In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-1*, pp. 786–791, Washington, DC, USA, 2005. IEEE Computer Society.

[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 Simpósio Brasileiro de Pesquisa Operacional*, Ubatuba, SP, Brasil, 2011.

[10] C. N. Lintzmayer, M. H. Mulati and A. F. da Silva. "Algoritmo Heurístico Baseado em Colônia de Formigas Artificiais ColorAnt2 com Busca Local Aplicado ao Problema de Coloração de Grafo". In *X Congresso Brasileiro de Inteligência Computacional*, Fortaleza, SP, BRA, 2011.

[11] C. N. Lintzmayer, M. H. Mulati and A. F. da Silva. "Toward Better Performance of ColorAnt ACO Algorithm". In *XXX International Conference of the Chilean Computer Science Society*, Curico, Chile, 2011.

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

[13] 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.

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

[15] 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.

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

[17] 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.

[18] C. N. Fischer, R. K. Cytron and R. J. LeBlanc. *Crafting a Compiler*. Addison Wesley, 2010.

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

[20] S. S. Muchnick. *Advanced Compiler Design and Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[21] C. N. Lintzmayer, M. H. Mulati and A. F. da Silva. "Register Allocation with Graph Coloring by Ant Colony Optimization". In *XXX International Conference of the Chilean Computer Science Society*, Curico, Chile, 2011.

[22] 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.

[23] M. Dorigo, M. Birattari and T. Stützle. "Ant Colony Optimization - Artificial Ants as a Computational Intelligence Technique". *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

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

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

[26] 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.

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

[28] D. Brélaz. "New Methods to Color the Vertices of a Graph". *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[29] 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.

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

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] A. Hertz and de Dominique de Werra. "Using Tabu Search Techniques for Graph Coloring". *Computing*, vol. 39, no. 4, pp. 345–351, 1987.