Ants cannot color graphs

ZUFFEREY, Nicolas

Abstract

Ant colony optimization was born in 1991. From that date, ant algorithms have been applied to numerous problems. One of its most famous adaptations was for graph coloring in 1997, resulting in an inspiring publication entitled “Ants can color graphs” and cited more than 700 times [3]. In this paper, with a certain sense of humor reflected in its title, it is showed that actually, 20 years later, ants cannot color graphs. This fact is explained in the light of two weaknesses of the usual ant methodology when making decisions, namely a cumbersome computation and the joint consideration of conflicting ingredients. A reconciliation of these contradictory statements relies in the enlargement of the ant optimization paradigm.

Reference


Available at:
http://archive-ouverte.unige.ch/unige:104577

Disclaimer: layout of this document may differ from the published version.
Ants cannot color graphs

Nicolas Zufferey
GSEM - University of Geneva, 1211 Geneva 4, Switzerland, n.zufferey@unige.ch

Abstract
Ant colony optimization was born in 1991. From that date, ant algorithms have been applied to numerous problems. One of its most famous adaptations was for graph coloring in 1997, resulting in an inspiring publication entitled "Ants can color graphs" and cited more than 700 times [3]. In this paper, with a certain sense of humor reflected in its title, it is showed that actually, 20 years later, ants cannot color graphs. This fact is explained in the light of two weaknesses of the usual ant methodology when making decisions, namely a cumbersome computation and the joint consideration of conflicting ingredients. A reconciliation of these contradictory statements relies in the enlargement of the ant optimization paradigm.

Keywords: ant algorithms, combinatorial optimization, graph coloring.

1 Introduction
A population of ants is employed in the state-of-the-art ant algorithms, in which an ant is simply a constructive heuristic that generates a single solution to the considered problem, iteratively, starting from an empty solution. In each step of the construction process, the involved ant makes its next decision according to two elements, namely the visibility (also called the greedy force or the heuristic information) and the trail. The former element represents the short-term profit of the decision (i.e., its immediate advantage), whereas the latter element is an information deduced from the history of the search. In other words, when making a decision, an ant has to find a tradeoff between (1) its own interest (visibility information) and (2) a kind of advice provided by other ants (trail information).

Section 2 presents the graph coloring problem, whereas Section 3 summarizes the main elements of a standard ant algorithm. The goal of this paper is to highlight that a success story like "Ants can color graphs" [3] (thesis in Section 4) can be turned into "Ants cannot color graphs" (antithesis in Section 5). Indeed, it will be showed that a basic greedy constructive algorithm can obtain much better results. Relying on the identification of the main drawbacks of the classic ant algorithms, this work proposes a reconciliation of these conflicting thesis in the enlargement of the ant optimization paradigm (synthesis in Section 6).

2 Graph coloring and its applications in engineering fields
Let $X = \{x_1, x_2, \ldots, x_n\}$ and $E$ denote the vertex set and the edge set of a graph $G = (X,E)$, respectively. In the graph coloring problem (GCP), each vertex has to be assigned exactly one color, i.e., an integer in $\{1, 2, 3, \ldots, n\}$, such that two adjacent vertices (i.e., connected with an edge) never get the same color. The goal consists in minimizing the number of different colors that are employed to fully color the graph. A conflict occurs on an edge if its two involved vertices receive the same color. In the $k$-coloring problem (denoted as $k$-GCP), an additional restriction appears as each vertex must be assigned a color in $\{1, \ldots, k\}$. To tackle this problem, the following optimization approach is often used: allow conflicts but minimize their number of occurrences. The $k$-GCP is solved if a solution with zero conflict if found. Starting with a large value of $k$ (upper bounded by $n$), the best metaheuristics for the GCP usually solve a series of $k$-GCPs with decreasing values of $k$ (both problems are NP-hard). More precisely, each time a $k$-coloring is obtained, a $(k-1)$-coloring is then searched. The process stops if zero conflict cannot be reached. The reader is referred to [6] for a survey on graph coloring, and some references on its numerous applications in various engineering domains (e.g., timetabling, scheduling, train platforming, register allocation, communication networks, frequency assignment).

3 Ant algorithms
Formally, a classic ant algorithm can be described as follows. Starting from scratch, in each step of the construction process, an ant adds an element to the current partial solution. As mentioned above, each decision $d$ is made according to a tradeoff between the visibility $V(d)$ ant the trail $T(d)$ of $d$. The probability $P_i(d)$ that ant $i$ selects $d$ is given in the Decision Equation (1), where $\alpha$ and $\beta$ are parameters, and $A_i$ is the set of allowed decisions that ant $i$ can make.

\begin{equation}
X = \{x_1, x_2, \ldots, x_n\} \quad \text{and} \quad E \quad \text{denote the vertex set and the edge set of a graph} \ G = (X,E), \text{ respectively. In the graph coloring problem (GCP), each vertex has to be assigned exactly one color, i.e., an integer in} \ \{1, 2, 3, \ldots, n\}, \text{ such that two adjacent vertices (i.e., connected with an edge) never get the same color. The goal consists in minimizing the number of different colors that are employed to fully color the graph. A conflict occurs on an edge if its two involved vertices receive the same color. In the} k-\text{coloring problem (denoted as} k-\text{GCP), an additional restriction appears as each vertex must be assigned a color in} \ \{1, \ldots, k\}. \text{ To tackle this problem, the following optimization approach is often used: allow conflicts but minimize their number of occurrences. The} k-\text{GCP is solved if a solution with zero conflict if found. Starting with a large value of} k \ (\text{upper bounded by} n), \text{ the best metaheuristics for the GCP usually solve a series of} k-\text{GCPs with decreasing values of} k \ (\text{both problems are NP-hard}). \text{ More precisely, each time a} k-\text{coloring is obtained, a} (k-1)-\text{coloring is then searched. The process stops if zero conflict cannot be reached. The reader is referred to} [6] \text{ for a survey on graph coloring, and some references on its numerous applications in various engineering domains (e.g., timetabling, scheduling, train platforming, register allocation, communication networks, frequency assignment).}

3 Ant algorithms
Formally, a classic ant algorithm can be described as follows. Starting from scratch, in each step of the construction process, an ant adds an element to the current partial solution. As mentioned above, each decision $d$ is made according to a tradeoff between the visibility $V(d)$ ant the trail $T(d)$ of $d$. The probability $P_i(d)$ that ant $i$ selects $d$ is given in the Decision Equation (1), where $\alpha$ and $\beta$ are parameters, and $A_i$ is the set of allowed decisions that ant $i$ can make.

\begin{equation}
X = \{x_1, x_2, \ldots, x_n\} \quad \text{and} \quad E \quad \text{denote the vertex set and the edge set of a graph} \ G = (X,E), \text{ respectively. In the graph coloring problem (GCP), each vertex has to be assigned exactly one color, i.e., an integer in} \ \{1, 2, 3, \ldots, n\}, \text{ such that two adjacent vertices (i.e., connected with an edge) never get the same color. The goal consists in minimizing the number of different colors that are employed to fully color the graph. A conflict occurs on an edge if its two involved vertices receive the same color. In the} k-\text{coloring problem (denoted as} k-\text{GCP), an additional restriction appears as each vertex must be assigned a color in} \ \{1, \ldots, k\}. \text{ To tackle this problem, the following optimization approach is often used: allow conflicts but minimize their number of occurrences. The} k-\text{GCP is solved if a solution with zero conflict if found. Starting with a large value of} k \ (\text{upper bounded by} n), \text{ the best metaheuristics for the GCP usually solve a series of} k-\text{GCPs with decreasing values of} k \ (\text{both problems are NP-hard}). \text{ More precisely, each time a} k-\text{coloring is obtained, a} (k-1)-\text{coloring is then searched. The process stops if zero conflict cannot be reached. The reader is referred to} [6] \text{ for a survey on graph coloring, and some references on its numerous applications in various engineering domains (e.g., timetabling, scheduling, train platforming, register allocation, communication networks, frequency assignment).}

3 Ant algorithms
Formally, a classic ant algorithm can be described as follows. Starting from scratch, in each step of the construction process, an ant adds an element to the current partial solution. As mentioned above, each decision $d$ is made according to a tradeoff between the visibility $V(d)$ ant the trail $T(d)$ of $d$. The probability $P_i(d)$ that ant $i$ selects $d$ is given in the Decision Equation (1), where $\alpha$ and $\beta$ are parameters, and $A_i$ is the set of allowed decisions that ant $i$ can make.
At the end of each generation (i.e., when all the ants \( i \in \{1, \ldots, N\} \) of the population have provided their solutions \( s_1, \ldots, s_N \), the following equation is generally used to update the trails: 
\[
T(d) = \rho \cdot T(d) + \Delta T(d), \hspace{1cm} \forall d \in A,
\]
where \( A \) is the set of all possible decisions, \( \rho \in [0, 1] \) is an evaporation parameter (usually around 0.9), and \( \Delta T(d) \) is a reinforcement term accounting for the trails put on decision \( d \) by the ants involved in the current generation. 
\( \Delta T(d) \) is usually computed based on the number of times the ants make decision \( d \), and on the quality of the generated solutions for which decision \( d \) was made. Formally, 
\[
\Delta T(d) = \sum_{i=1}^{N} \Delta T_i(d), \hspace{1cm} \text{where } \Delta T_i(d) \text{ is proportional to the quality of } s_i \text{ if ant } i \text{ has performed decision } d.
\]
Algorithm 1 summarizes the main steps of an ant algorithm (where the initial values of the trails are usually set in order to avoid any initial bias). Even if several variants and extensions of this generic algorithm can be found in the literature [4], the main feature is always the same: use a tradeoff between visibility and trail to make a decision.

### Algorithm 1: Generic Ant Algorithm

**Input:** instance of the considered problem, parameters \( \alpha \) and \( \beta \), initial values of the trails.

While no stopping condition is met, do:
- for \( i = 1 \) to \( N \), do: ant \( i \) builds a solution \( s_i \) step by step based on Equation (1);
- update the trails by the use of the set \( \{s_1, \ldots, s_N\} \) (or a subset based on the best solutions);

**Output:** best solution generated during the search.

### 4 Thesis: ants can color graphs

In order to understand AntCol (i.e., an ant method for graph coloring proposed in [3]), we first need to understand Dsatur [2], because the former was basically derived from the latter by adding a trail system to it. Dsatur is a greedy constructive method for graph coloring. Let \( N(x) \) be the set of vertices that are adjacent to vertex \( x \). In each step of Dsatur, an uncolored vertex \( x \) is first selected and then colored with the smallest feasible color \( j \) (i.e., without generating any conflict). The next vertex \( x \) to color has the largest saturation degree, defined as the number of different colors already used in \( N(x) \). Ties are broken with the largest number of uncolored adjacent vertices. If ties are again encountered, a random choice is performed.

In AntCol, a decision step also consists in selecting a vertex \( x \) and assigning a color \( j \) to it. The only difference relies in the way that \( x \) is selected. Indeed, a tradeoff based on Equation (1) is used for this purpose. The visibility \( V(x) \) of a vertex \( x \) is defined as in Dsatur (i.e., its saturation degree), whereas its trail \( T(x) \) is computed as \( \sum_{y \in C_j} \ell(x, y) \), where \( C_j \) is the vertex set with color \( j \) in the solution under construction, and \( \ell(x, y) \) is a trail value proportional to two ingredients, namely frequency and quality. The frequency component relies on the number of times that both vertices \( x \) and \( y \) have the same color in the solutions previously provided by the ants, whereas the quality element depends on the number of colors used in the previously generated solutions in which \( x \) and \( y \) have the same color. The parameters \( (\alpha, \beta, \rho) \) were tuned to (1, 4, 0.5). In other words, the trail dominates the visibility but the evaporation is significant (as \( \rho \) is much smaller than the usual 0.9 value).

Based on [3], Dsatur and AntCol are compared in Table 1. For AntCol, only the best score among two calibrations is given here for each graph (100 ants in the first calibration, 300 ants in the second one). For each graph, two features are first indicated: its number \( n \) of vertices (see column 1) and its density \( p \) (i.e., the average number of edge between two vertices, see column 2). Such graphs are random graphs (i.e., they do not have a specific structure). Column 3 (resp. 4) gives the average number of colors used by Dsatur (resp. AntCol) over 10 runs. On the one hand, one might observe that AntCol significantly outperforms Dsatur if the number of vertices is below 500. It highlights that the trail system added to Dsatur to derive AntCol is efficient. On the other hand, one might deduce that some effort is required to calibrate and improve AntCol for larger graphs. These two assumptions are in line with the following statements depicted in [3]: "The colorings that we have obtained are satisfactory. The latter were shown to be significantly better than those obtained by sophisticated constructive methods. AntCol is an easily implementable algorithm. Such an approach looks very promising even though the results obtained can be further improved." More generally, based on Table 1, one can say that "Ants can color graphs".
5 Antithesis: ants cannot color graphs

Based on [9], the goal of this section is to present additional results for both Dsatur and AntCol, as well as to compare their performances with state-of-the art results for well-known benchmark instances. The following ingredients are used for this purpose.

- Consider benchmark graphs with various structures (i.e., not only random graphs) and with densities belonging to a broader range (i.e., not only in interval [0.4, 0.6], but in interval [0.1, 0.9]). Below, the random graphs will have a label starting with "DSJ", whereas the other graphs (i.e., with a label starting with "le" and "flat") have a specific structure.
- Perform experiments with the well-established coloring metaheuristic PartialCol, which is a simple, quick and efficient tabu search proposed in [1], in which one can also find more information on the graph structures of the used benchmark instances. In PartialCol, a decision \( d \) mainly consists in recoloring one vertex as best as possible (a quick and basic repair process is performed to remove the possible conflicts). If a vertex gets a new color in iteration \( t \), it is then forbidden to change its color for \( tab \) iterations, where \( tab \) is a parameter that is dynamically managed along the search.
- Use a same time limit \( TL \) (namely, 1 hour) and a common computer (namely, Intel Core2 Duo Processor E6700, with 2.66 GHZ and RAM2GB) for the compared methods. This allows an accurate comparison of the involved methods. Indeed, it is not fair to deduce that a method \( M_2 \) outperforms a method \( M_1 \) if the time employed by \( M_2 \) is much smaller. In this case, as Dsatur typically needs some seconds for each graph, it is restarted as long as \( TL \) is not reached, and the best generated solution is provided at the end.

The results are given in Table 2. The name of the graph is indicated in the first column, whereas \( n \) and \( p \) (defined as above) are given in columns 2 and 3, respectively. For each graph, the best number of colors used by Dsatur (resp. AntCol and PartialCol) is given in column 4 (resp. 5 and 6), whereas the best-known value, denoted as \( k^* \), is indicated in the last column. The \( k^* \) values were obtained from [5]. One can observe that PartialCol is much better than Dsatur. This can be expected as a tabu search metaheuristic is usually better than a greedy constructive heuristic, even if the latter is restarted several times. Surprisingly, Dsatur significantly outperforms AntCol for each graph, independently of its size/structure/density. This obviously contributes to the antithesis, namely "Ants cannot color graphs". Indeed, such results show that the ingredients added to Dsatur to derive AntCol are not efficient (in terms of solution quality), and they actually lead to a more complex and slower method. The following reasons could explain this situation. First, Equation (1) is needed to make a very basic decision \( d \) which only consists in giving a color to a vertex. This is obviously cumbersome in terms of the computational effort. In other words and in contrast with PartialCol, a lot of computation is required for making the decision of recoloring a single vertex. Second, conflicting components are used in Equation (1), namely the visibility and the trail. A tradeoff between these two elements appears as inappropriate to efficiently assign a color to a vertex.

6 Synthesis: reconciliation and enlargement of the ant paradigm

In this paper, the statement "Ants can color graphs" and its opposite are confronted. In a work dating back to 1997 [3], a first ant approach was proposed for graph coloring, and its potential has been shown under certain assumptions. Unfortunately, this potential cannot be confirmed here when accurately and fairly comparing Dsatur (i.e., a greedy constructive heuristic) with AntCol (i.e., the ant metaheuristic derived from Dsatur in [3]). Indeed, if the same computation time is allocated to both methods, the former clearly outperforms the latter for each type of graph. However, this finding of apparent failure opens the door, positively, to a better understanding of the highlighted weaknesses of the ant paradigm, namely a bad and cumbersome mix of conflicting criteria (i.e., the visibility and the trail) when making a decision (which could be as simple as coloring a single vertex!). Building on that, the questions below can be raised in order to enlarge the ant paradigm.

- Should an ant be different from a constructive heuristic, but without loosing its two inherent elements (i.e., visibility and trail)? For instance, an ant might be either a decision helper or a full metaheuristic.
- Can we use the visibility and the trail at two different times of the decision process (i.e., do not confront these elements simultaneously)? For instance, the visibility (resp. trail) should only focus on the exploration (resp. exploitation) ability of the method.
- What should be the role of the trail among intensification (i.e., focus the search process on identified
promising zones of the search space, which is the usual role of the trail system) and diversification (i.e., use the search history to orient the future steps towards unexplored zones of the solution space)?

- Can we accelerate the decision process, for instance, by reworking the nature of the Decision Equation (and not simply its frequency of use, as often observed in the literature)?
- Should we use the visibility and trail features within another optimization framework (e.g., a well-established local search or another population-based method), instead of simply boosting the solutions provided by the ants with local search procedures (as it is often done in the literature)?

Some basic steps in the above directions have been already performed (e.g., [7, 8, 9]), but further investigations should be made. Even if "ants cannot color graphs", it is obvious that "ants can inspire optimization". Without any doubt, [3] has significantly contributed to the second statement.

Table 1: Comparison of Dsatur and AntCol on random graphs.

<table>
<thead>
<tr>
<th>n</th>
<th>p</th>
<th>Dsatur</th>
<th>AntCol</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.4</td>
<td>14.7</td>
<td>12.8</td>
</tr>
<tr>
<td>0.5</td>
<td>18.45</td>
<td>13.65</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>22.6</td>
<td>18.75</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>0.4</td>
<td>24.4</td>
<td>20.9</td>
</tr>
<tr>
<td>0.5</td>
<td>30.65</td>
<td>25.95</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>36.7</td>
<td>33.95</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>0.4</td>
<td>42.4</td>
<td>34.1</td>
</tr>
<tr>
<td>0.5</td>
<td>47.95</td>
<td>40.15</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>53.7</td>
<td>47.3</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Performance of Dsatur and AntCol on benchmark graphs.

<table>
<thead>
<tr>
<th>Graph</th>
<th>n</th>
<th>p</th>
<th>Dsatur</th>
<th>AntCol</th>
<th>PartialCol</th>
</tr>
</thead>
<tbody>
<tr>
<td>flat</td>
<td>300</td>
<td>0.48</td>
<td>39</td>
<td>44</td>
<td>26</td>
</tr>
<tr>
<td>flat</td>
<td>300</td>
<td>0.48</td>
<td>39</td>
<td>44</td>
<td>28</td>
</tr>
<tr>
<td>1e450,5xc</td>
<td>450</td>
<td>0.17</td>
<td>23</td>
<td>28</td>
<td>15</td>
</tr>
<tr>
<td>1e450,10d</td>
<td>450</td>
<td>0.17</td>
<td>23</td>
<td>28</td>
<td>15</td>
</tr>
<tr>
<td>1e450,25d</td>
<td>450</td>
<td>0.17</td>
<td>28</td>
<td>33</td>
<td>27</td>
</tr>
<tr>
<td>1e450,5xc</td>
<td>450</td>
<td>0.17</td>
<td>28</td>
<td>33</td>
<td>27</td>
</tr>
<tr>
<td>1e450,25d</td>
<td>450</td>
<td>0.17</td>
<td>28</td>
<td>33</td>
<td>27</td>
</tr>
<tr>
<td>DS/JC1000.1</td>
<td>500</td>
<td>0.1</td>
<td>15</td>
<td>17</td>
<td>12</td>
</tr>
<tr>
<td>DS/JC1000.5</td>
<td>500</td>
<td>0.9</td>
<td>68</td>
<td>68</td>
<td>58</td>
</tr>
<tr>
<td>DS/JC1000.9</td>
<td>500</td>
<td>0.9</td>
<td>158</td>
<td>167</td>
<td>127</td>
</tr>
<tr>
<td>DS/JR500.1c</td>
<td>500</td>
<td>0.97</td>
<td>87</td>
<td>97</td>
<td>85</td>
</tr>
<tr>
<td>DS/JR500.5</td>
<td>500</td>
<td>0.47</td>
<td>129</td>
<td>136</td>
<td>126</td>
</tr>
<tr>
<td>DS/JC1000.1</td>
<td>1000</td>
<td>0.49</td>
<td>111</td>
<td>120</td>
<td>100</td>
</tr>
<tr>
<td>DS/JC1000.5</td>
<td>1000</td>
<td>0.49</td>
<td>111</td>
<td>120</td>
<td>100</td>
</tr>
<tr>
<td>DS/JC1000.9</td>
<td>1000</td>
<td>0.9</td>
<td>233</td>
<td>313</td>
<td>226</td>
</tr>
</tbody>
</table>

References