University of Ljubljana
Institute of Mathematics, Physics and Mechanics
Department of Mathematics
Jadranska 19, 1111 Ljubljana, Slovenia

Preprint series, Vol. 42 (2004), 952

ANTS AND GRAPH COLORING

John Shawe-Taylor Janez Žerovnik

ISSN 1318-4865

December 29, 2004

Ljubljana, December 29, 2004

Ants and graph coloring

John Shawe-Taylor and Janez Žerovnik

Abstract— **Ants algorithm is an evolutionary search method for solving combinatorial optimization problems. In this note we propose a version of the algorithm for coloring a graph with a fixed number of colors. Based on approximate colorings obtained by single ants, edges of maximal evidence are added so that the new ants generation does only attempt to construct colorings with additional restrictions. Probabilities for addition of misleading edges are shown to be very low. Experimentaly we obtain speedup over not cooperating ants and a long single ant run.**

I. Introduction

Many combinatorial optimization problems are NP-hard. It is generally believed that NP-hard problems cannot be solved to optimality within times which are polynomially bounded functions of input size. Therefore, there is much interest in heuristic algorithms which can find near-optimal solutions within reasonable running times. Among heuristics there are several general approaches also called metaheuristics. Some popular metaheuristics are inspired by analogies to statistical mechanics as, for example, simulated annealing, and some others are inspired by biological systems as, for example, genetic algorithms or, more general, evolutionary algorithms.

One of the most studied NP-hard problems is the graph coloring problem [13]. Graph coloring has numerous applications in scheduling and other practical problems including register allocation, the design and operation of flexible manufacturing systems frequency assignment etc. It is well-known that both the problem of deciding whether a given graph can be colored by k colors for $k > 2$ and the problem of computing the chromatic number are NP–hard.

A version of the ants algorithm for estimating the chromatic number of a graph was recently proposed in [6]. 'Ants can color graphs' was the message of the paper, but unfortunately it was shown later [21] that the implementation of the ants paradigm in [6] did not even outperform simple memoryless restarts algorithm. Here we give an alternative ants algorithm for coloring a graph with a given fixed number of colors k. Our algorithm differs from algorithm of [6] both in the procedure performed by a single ant and in the way how the information experienced is memorized. Speedup over restarts and over long runs on a set of benchmark instances is shown experimentally.

There are many other approaches to graph coloring. A lot of both experimental and theoretical work has been done on various graph coloring methods, for example see [9], [12], [13], [14], [17], and many more. However, the aim of this paper is not to outperform the best graph coloring algorithms. The goal of our experiment is to show that the proposed way of storing the information may be useful for speeding up the computation. It remains open to see what are the conditions at which this effect can be expected.

The present authors however strongly believe that no metaheuristic can produce competitive results on any problem if the procedures it uses are not competitive. For the ants algorithm, this includes the single ants procedure and in particular the method for combining and memorizing the information obtained by past single ants runs.

The rest of the paper is organized as follows. First, we briefly recall the graph coloring problem. Then we overwiev the ants metaheuristics and its two main ingredients, the local search type procedure which is performed by each ant and the method for memorizing the information experienced by the ants. Finally, we give results of the computational experiments and conclusions.

II. Graph coloring and the main idea

If G is a graph, we write $V(G)$ for its vertex set and $E(G)$ for its edge set. $E(G)$ is a set of unordered pairs $xy = \{x, y\}$ of distinct vertices of G.

A *proper k-coloring* of vertices of a graph G is a function f from $V(G)$ onto a set (of colors) X, $|X| = k$, such that $uv \in E(G)$ implies $f(u) \neq f(v)$.

The decision problem of k–coloring is: given graph G and an integer k on input, answer the question "is G k–colorable?".

The problem is NP–hard for general graphs [10], but also for many special classes of graphs including planar graphs, regular graphs etc. It is also known that there is no polynomial approximation algorithm unless P=NP [15].

The ideas behind the randomized graph coloring algorithm reported in [17] have been extended in a series of papers [22], [23], [18], [2], [3], which have both analyzed and improved the heuristics. In [2] the energy landscape for Mean Field Annealing is altered by adapting weights on the graph edges. The paper [20] investigates an extension of this idea by introducing additional edges into the graph. This may seem a counter-intuitive strategy as it might appear to make the graph coloring problem more difficult. The idea is based on the fact that when restricted to the class of graphs with lowest vertex degree greater than αn for arbitrary $\alpha > 0$, the decision problem of 3-colouring is polynomial [7]. It is also known that the edge adding is error free with high probability provided some rather rough conditions are satisfied [20].

A short version of this paper was published in the proceedings of the ICANNGA'01 conference.

Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK. Email: jst@dcs.rhbnc.ac.uk

Faculty of Mechanical Engineering, University of Maribor, Smetanova 17, SI-2000 Maribor, Slovenia and Department of Theoretical Computer Science, IMFM, Jadranska 19, SI-1111 Ljubljana, Slovenia. Email: janez.zerovnik@imfm.uni-lj.si

Another way of viewing the strategy is in terms of the way that humans often attempt to solve problems by further constraining them based on probabilistic reasoning from known constraints, hence reducing the size of the search space, while hopefully not excluding all feasible solutions. In the ants context, it seems likely that two vertices must be colored differently if many ants have proposed such near optimal colorings.

On the other hand, there is some theoretical evidence that repeated trials of various random heuristics (including simulated annealing and mean field annealing) on average perform better than one long run [8]. However, at each restart we loose all information obtained by previous unsuccessful trials, which may have obtained reasonably good near optimal colorings. Therefore, a possible way of accelerating the restarts algorithm is to somehow accumulate the information obtained by the near optimal colorings obtained so far. The idea of using the information of previous local searches is not new. In a different way it is explored by heuristics such as tabu search [11], reactive tabu search [1], and genetic algorithms [9].

III. Ants algorithm for the graph k-coloring problem

The algorithm is based on observations of an ant colony in search of a nearby feeding source. Entomological studies have shown that an ant colony converges towards an optimal solution whereas each single ant is unable to solve the problem by itself within a reasonable amount of time. Colorni et al [4] first developed Ant System for tackling so–called Assignment Type Problems (ATPs). The idea can be briefly explained as follows: The solution is produced by a colony of ants. Each ant builds a feasible solution at each iteration, i.e. it performs one run of a relatively fast algorithm which produces a near optimal solution. Initially, the ants do not have any knowledge of the problem to be solved. At the end of each iteration the experience of every ant is memorized. This information is then used in the next iteration.

algorithm ANTS($nants, ncycles, T, M, T_w$)
$G' := G$
for i:=1 to *ncycles* **do begin**
    for j:=1 to *nants* **do (in parallel)**
        PW(G', T, M)
    update(matrix $E(G')$, T_w)
    if k-coloring of G found **then** exit
end

IV. Single ants procedure, PW

The algorithm we use is a generalization of the algorithm of Petford and Welsh [17]. This algorithm and some of its generalizations have proved to be reasonably good heuristics for coloring various types of graphs including random k-colorable graphs, DIMACS challenge graphs [13], frequency assignment "realistic" graphs, and others [17], [23], [18].

The algorithm used here differs slightly from previous implementation because it uses the transformed edges set $E(G')$ for computing the new colors and the original edge set $E(G)$ for computing the cost function.

algorithm PW(G', T, M)
color vertices randomly using colors $1, 2, , \ldots, k$;
while coloring not proper and iterations $< M$ **do**
    select a bad vertex v (randomly)
    assign a new color to v
end while

In the first line of the while loop, a bad vertex is selected uniformly at random among the bad vertices, i.e. vertices which are endpoints of some edge which violates a constraint. In the version used in this paper, the bad vertices are defined with respect to edges of G (and not G').

A new color is assigned at random. The new color is taken from the set $\{1 \ldots k\}$ by sampling according to the distribution defined as follows: The probability of color i to be chosen as a new color of vertex v is proportional to

$$exp(-S_i/T)$$

where S_i is the number of edges of G' with one endpoint at v and the other colored by color i.

T is a parameter of the algorithm, which may be called temperature because of the analogy to temperature of the simulated annealing algorithm.

If no k-coloring is found before a time limit of M iterations is reached, the algorithm stops. (An iteration is a call of the function which computes a new color.) If the algorithm stops without finding a proper coloring, the best assignment can be regarded as an approximate solution, approximating the cost function which counts the number of bad edges, i.e. edges with both endpoints colored by the same color.

Remark: there are two decisions which are of crucial importance for the behaviour of the algorithm: to define the number of iterations M and to define a good value for parameter T. We have no complete answer to these two questions. The choices we made work well on instances tested, but in general fine tuning is probably needed as is the usual case with other heuristics of similar type such as simulated annealing, tabu search, genetic algorithms, etc. The idea of adapting the temperature is addressed in [19]. A heuristic for tuning an analogous parameter is used with success in [16].

In this paper we do not aim to precisely tune the parameter T, therefore we either use $T = 0.7$ which corresponds to the value implicitly used in [17], [23], or use a presumably good temperature based on previous experiments (reported elsewhere [18], [19]). More tuning could only improve the performance of the algorithm.

V. Memorizing the experience, update

In the algorithm described below we will make use of the following quantities which are defined for the various colorings generated. Each coloring c is assigned a weight $w_G(c)$ given by

$$w_G(c) = \exp\{-b(c)/T_w\},$$

where $b(c)$ is the number of edges in G that connect vertices colored with the same color by c, and T_w is a parameter which defines the particular weighting function. Note that $w(c) \leq 1$, while $w(c) = 1$ for proper colorings.

The complement of a graph $G = (V, E)$ is the graph

$$\bar{G} = (V, (V \times V) \setminus (E \cup \{(i,i)|i \in V\})).$$

For a set of colorings C, we define the evidence for an nonedge $(x, y) \in \bar{G}$ to be

$$E_{G,C}((x,y)) = \sum_{c \in C, c(x) \neq c(y)} w_G(c).$$

We also define the evidence of a set of colorings as

$$E_G(C) = \sum_{c \in C} w_G(c).$$

At a call of procedure update(E, T_w), the maximal evidence over nonedges is computed and all edges with maximal evidence are added.

VI. Estimates of probabilities for edge adding

In this section we perform some analysis of the number of approximate colourings that might be needed to give good bounds on the probability of including an edge which increases the chromatic number of the graph.

Throughout this section we will make the natural assumption that approximate colourings are drawn independently according to some fixed distribution. A colouring c will be called incompatible with adding the edge (u, v), if $c(u) = c(v)$. For two vertices u, v which are not connected in the graph G, let a_{uv} denote the probability that an approximate colouring is incompatible with the edge (u, v).

By $G \cup \{(u, v)\}$ we denote the graph obtained from G by adding the edge (u, v). If $\chi(G \cup \{u, v\}) > \chi(G)$ we say that (u, v) is a bad (non) edge, otherwise (u, v) is a good edge. We denote the set of bad edges by $B(G)$. Note that the increase in chromatic number can be at most one.

Lemma 1: Let $0 < \alpha \leq a_{uv}$. The probability that a rate of $a_{uv} - \alpha$ or better is observed in a sample of m random colourings for any pair of vertices u, v not connected in G is bounded by

$$\frac{n(n-1)}{2} \exp(-\alpha^2 m/(2\alpha_{uv})) \leq \frac{n(n-1)}{2} \exp(-\alpha^2 m/2),$$

where $n = |V(G)|$ is the number of vertices in the graph G.

Proof: The probability of a rate β or better being observed for a particular pair u, v is given by

$$\sum_{i=0}^{\beta m} \binom{m}{i} (a_{uv})^i (1 - a_{uv})^{m-i} \quad \leq \quad \exp(-(a_{uv} - \beta)^2 m/2a_{uv})$$

using Chernoff bounds to obtain the inequality. Hence, if $\beta = a_{uv} - \alpha$ we obtain the bound

$$\exp(-\alpha^2 m/2a_{uv}) \leq \exp(-\alpha^2 m/2).$$

The union bound of probabilities over all possible non edges gives the result. ∎

Corollary 2: Supppose that for a graph G with $\chi(G) = k$ the probability that an approximate colouring is incompatible with a bad edge is A times the probability for a random colouring, then with probability greater than $1 - \delta$ adding any edge for which the observed relative frequency is less than $1/(2k)$ will not increase the chromatic number, provided

$$m \geq \frac{8Ak}{(2A-1)^2} \left\{ \ln \binom{n}{2} + \ln \frac{1}{2\delta} \right\}.$$

If the observed relative frequency is 0 then with probability greater than $1 - \delta$ adding any edge will not increase the chromatic number provided

$$m \geq \left\{ \ln \binom{n}{2} + \ln \frac{1}{\delta} \right\} / \ln \left(\frac{k}{k-A} \right).$$

Proof: The frequency of incompatible random colourings is $1/k$, since this is the fraction of random assignments that give the same colour to both ends of an edge. By assumption we have $a_{uv} \geq A/k$ for all bad edges (u, v). Therefore, we can bound the probability that a bad edge has an observed relative frequency of incompatibility less than $1/(2k)$ using the lemma with α satisfying $\alpha_{uv} - \alpha \leq 1/(2k)$. Hence,

$$\begin{aligned} \frac{\alpha_{uv} - \alpha}{\alpha_{uv}} &\leq \frac{1}{2A} \quad \Rightarrow \quad \frac{\alpha}{\alpha_{uv}} \geq 1 - \frac{1}{2A} \\ \Rightarrow \quad \frac{\alpha^2}{\alpha_{uv}^2} &\geq \left(1 - \frac{1}{2A}\right)^2 \quad \Rightarrow \quad \frac{\alpha^2}{\alpha_{uv}} \geq \frac{(2A-1)^2}{4Ak} \end{aligned}$$

we require m such that $\binom{n}{2} \exp(-m(2A-1)^2/(8Ak)) \leq \delta$. Solving for m gives the first result. If $\alpha_{uv} \geq A/k$, then the probability that the observed relative frequency is 0 for a bad edge can be bounded by

$$\binom{n}{2} \left(1 - \frac{A}{k}\right)^m.$$

Setting the right hand side equal to δ and solving gives the second result. ∎

In practice we observe that many edges are compatible with all approximate colourings. The Corollary indicates that provided the approximate colourings do increase the incompatibility of bad edges by twice that of random colourings then with probability at least $1 - \delta$ such edges can safely be added provided

$$m \geq \left\{ \ln \binom{n}{2} + \ln 1/\delta \right\} / \ln \left(\frac{k}{k-2} \right).$$

Table I tabulates some relevant values of these numbers. Note how few extra colourings are needed to give very significant increases in confidence or to apply to very much larger graphs.

We have so far considered the case where there is assumed to be a bound on the probabilities a_{uv} for all bad edges (u, v) and the observed relative frequency of incompatibility on the added edges are very low. This will be the case if there are at least some good edges for which a_{uv}

n	k	δ	$m(n,k,\delta)$
450	5	0.1	28
450	5	0.01	32
450	5	0.001	37
450	15	0.1	97
450	15	0.01	113
450	15	0.001	129
100,000	5	0.001	58

TABLE I
Numbers of colourings required to give confidence in edge adding

is also low. These restrictions may not hold in all cases, and the strategy of only including the maximal evidence edges is intended to extend the applicability of the technique. The next corollary shows that using the maximal evidence criterion does indeed make it possible to weaken the conditions on the values of a_{uv}.

Corollary 3: If there is at least one good edge (u,v) for which

$$a_{uv} < \min_{(u',v')\in B(G)} \{a_{u'v'}\} - \alpha,$$

for $\alpha > 0$, then with probability $1-\delta$ the maximal evidence edge will be a good edge provided the number m of approximate colourings satisfies

$$m \geq \frac{8}{\alpha^2}\left\{2\ln n + \ln\frac{1}{2\delta}\right\}.$$

Proof: If the difference between $a_{u'v'}$ and the corresponding observed relative frequency is less than $\alpha/2$ for all edges (u',v'), then the maximal evidence edge will be a good edge, because no bad edge will have observed relative frequency better than (u,v) as $a_{uv}+\alpha/2 < a_{u'v'}-\alpha/2$, for all bad edges $(u'v')$. By Lemma 1 the probability that the difference between observed relative frequency and $a_{u'v'}$ is less than $\alpha/2$ for all non edges is bounded by

$$n(n-1)\exp(-\alpha^2 m/8)/2.$$

By choosing m as in the corollary statement this value is made less than δ as required. ∎

Hence, the heuristic will work provided that there is at least one edge which can be separated from the bad edges by a polynomially small gap. This leads to the following further corollary.

Corollary 4: Assume that $RP \neq NP$. For any polynomial time randomised approximate colouring algorithm there exists a sequence of graphs G_n, with $|V(G_n)| = n$ for which the difference

$$\min_{(u,v)\in B(G)}\{a_{uv}\} - \min_{(u,v)\in E(G)}\{a_{uv}\}$$

decays faster than $1/P(n)$ for any polynomial $P(n)$.

Proof: If no such sequence existed for some randomised colouring algorithm $\mathcal{A}$ then we could produce a randomised polynomial algorithm for the NP-complete problem of graph colouring as follows. Let $P(n)$ be a polynomial which satisfies the condition that

$$\min_{(u,v)\in B(G)}\{a_{uv}\} - \min_{(u,v)\in E(G)}\{a_{uv}\} > 1/P(n)$$

for all graphs G with $|V(G)| = n$. Note that for a graph with no good edges we assume that the colouring algorithm always delivers a correct colouring. This is a trivial addition to the algorithm if it does not already implement it, since there is no choice of different colours up to permutation of their values. Set $\alpha = 1/(2P(n))$ and set m according to the estimate of Corollary 3 with $\delta = 1/n^2$:

$$m = 8P(n)^2\left\{4\ln n - \ln 2\right\}$$

Perform approximate colourings before adding the single maximal evidence edge. Continue until either a colouring is found or too many edges have been added to the graph (e.g. the graph is complete). If the graph is coloured pronounce the original graph k-colourable, otherwise answer not k-colourable. Since the separation between bad and good edges is at least $1/P(n)$, with probability $1-\delta$ the edges added were consistent with a k-colouring provided the original graph was k-colourable. Hence, with probability at least 1/2 the final graph remains k-colourable if the original graph was k-colourable. On the other hand if the original graph was not k-colourable the final graph will also not be. The running time of the algorithm is polynomial since a polynomial number of edges is added and at each stage a polynomial number of approximate colourings is generated. ∎

This final result suggests that an interesting avenue of research might be to identify classes of graphs for which a polynomial separation does exist for some well studied simple graph colouring algorithms.

VII. Computational experiments I

The algorithms were first run on a series of four 'Leighton' graphs which are 5-colourable and which were proposed as part of the DIMACS Challenge in graph coloring [13] as examples of graphs that are hard to color. In the first set of experiments the parameters T, M, and T_w were set to 0.7, $200n$ and 3.0, respectively. As a comparison we also ran the Petford Welsh algorithm in two regimes. In the first instance we performed repeated short runs of the algorithm with the same number of iterations as used in the ants algorithm. In the second regime one long run was performed. Note that in both cases the same fixed temperature was used. The numbers in Table II indicate the total number of vertex updates performed whether or not the color of the vertex was changed.

For the second set of experiments we took the 15-colourable Leighton graphs. Here, the temperature $T = 0.7$ did not give good results in any regime. We therefore did another experiment with $T = 0.3$. In this case, the graph le450_15c was again never coloured and the graph le450_15c was coloured only in 3 out of 10 runs of the ants algorithm

	ANTS $nants = 25$ $ncycles = 10$ $M = 200n$	STANDARD RESTARTS	ONE RUN
le450_5a: Average degree: 25.40	2.7(10)	6.2(7)	9.0(7)
le450_5b: Average degree: 25.48	3.8(10)	2.5(1)	11.1(3)
le450_5c: Average degree: 43.57	2.5(10)	6.1(10)	0.9(1)
le450_5d: Average degree: 43.36	2.7(10)	5.0(10)	5.7(2)

TABLE II

Results of the Algorithms on the 5-colourable Leighton Graphs. The average number of iterations ($\times 10^6$) over successfull runs are given. In parenthesis is the number of successfull runs (out of 10). Time limit was 22.5×10^6.

	ANTS (25, 10, ., $200n$, .)	ANTS2 (50, 5, ., $200n$, .)	ANTS3 (25, 5, ., $400n$, .)	STAND. REST.	ONE RUN
15a	0.5(2)	1.0(3)	2.0(8)	8.2(7)	1.0(10)
15b	1.2(4)	0.9(4)	1.6(10)	5.2(10)	2.0(10)

TABLE III

Results of the Algorithms on the 15-colourable Leighton Graphs. 15a and 15b denote the Leighton graphs le450_15a and le450_15b with average degrees 36.30 and 36.31, respectively. The parameters of ants algorithms are (nants, ncycles, M).

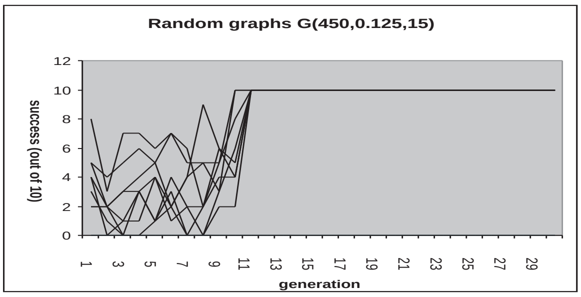

Fig. 1. Average generation successes of 10 ant colonies.

5	4	4	2	2	3	0	8	5	0
4	0	2	2	2	1	0	3	2	0
5	1	3	1	0	0	0	7	3	0
6	3	4	1	0	3	0	7	3	0
5	1	5	4	1	1	0	6	4	0
2	2	7	1	4	3	0	7	2	0
4	0	6	2	2	0	0	5	4	0
5	2	2	0	2	0	0	5	9	0
5	4	5	3	6	2	0	3	6	0
10	4	8	6	4	2	0	10	5	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0
10	10	10	10	10	10	0	10	10	0

TABLE IV

The numbers of successful runs (out of 10) of 10 colonies in 30 cycles. Random graphs $G(450, 0.125, 15)$.

while the restarts and long runs were never successful. Better results were observed for the other two graphs. Because of a relatively poor performance of the ants algorithm in this case we also tested two more combinations of the parameters of ants algorithm: in the first case (ANTS2) we increased *nants* to 50 and reduced *ncycles* to 5, in the second case (ANTS3) we doubled M and reduced *ncycles* to 5. The results are given in Table III.

VIII. Computational experiments II

In the previous section we were interested in the design of an algorithm which would speed-up standard restarts in the sense that we expect (or better, hope) that a proper colouring of a given graph will be found faster. In this section we present the result of the experiments in which we are observing a population of ants and look at the rate of success of generations of ants which act according to the knowledge of previous ants generations which is stored in virtual edges. In each of the experiments 10 ant colonies are considered. For each colony, at each of the *ncycles* rounds the number of successful runs (out of *nants* runs) are recorded.

Many classes of graphs appear to be very easy for colouring with the PW algorithm [17], [23], [18], [21]. In order to see the improvement of the ants approach we need graphs for which the success rate of the PW algorithm is relatively low, or, as we argue later, is 0 with good approximate colourings obtained.

Besides the Leighton graphs from the DIMACS benchmark library we use the random k-colourable graphs $G(n, p, k)$ which are generated as follows [17]: partition the vertex set of n vertices into k sets of size $\lfloor \frac{n}{k} \rfloor$ or $\lceil \frac{n}{k} \rceil$. For each pair of vertices from different sets insert an edge with probability p, independetly.

For the first experiment, the random graphs $G(450, 0.125, 15)$ were taken. These graphs are at temperature $T = 0.6$ difficult enough for the algorithm PW so that the probability of success of a single ant is around 50%. As the Table IV and Fig. 1 show, already after around 10 iterations (*ncycles*), all 10 ant colonies have learned how to colour the graph with success rate 100%. Fig. 2 gives the numbers of virtual edges.

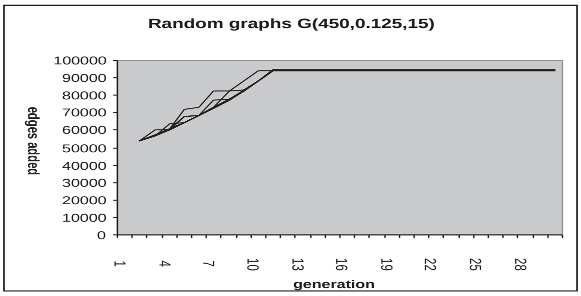

Fig. 2. Average edges added in each of 10 ant colonies.

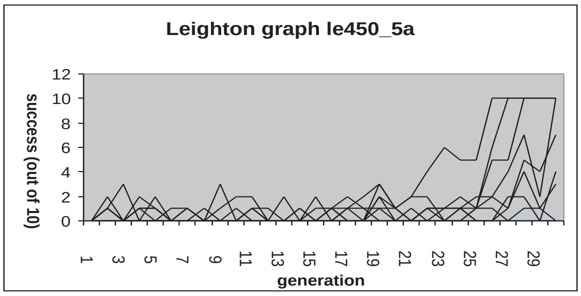

Fig. 3. Average generation successes of 10 ant colonies.

0	0	0	0	0	0	0	0	0	0
0	1	0	0	2	0	1	1	1	0
0	0	0	0	0	0	0	3	0	0
1	2	1	0	0	0	1	0	0	0
1	1	1	0	2	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	1	1	0	0
0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	3	0	0	0
2	0	0	0	0	0	0	1	0	0
2	1	0	0	0	1	1	0	0	0
0	0	0	0	0	0	1	0	0	0
0	2	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	1	0
0	2	1	0	0	0	0	0	0	0
1	0	1	0	0	0	1	1	0	0
0	0	1	0	1	1	2	0	0	0
0	0	2	0	1	0	1	0	0	0
3	2	3	0	0	2	1	0	0	1
1	0	1	0	0	1	1	0	0	0
2	0	2	0	0	0	0	1	0	0
2	0	4	0	0	1	0	0	1	1
0	0	6	0	1	1	0	0	1	0
1	1	5	0	1	1	0	0	2	0
2	1	5	0	0	0	1	1	1	0
2	6	10	0	0	0	2	1	5	0
4	10	10	0	1	2	1	0	5	0
7	10	10	0	4	2	5	1	10	0
2	10	10	0	1	0	4	1	10	0
10	10	10	0	3	4	7	0	10	0

TABLE V

The numbers of successful runs (out of 10) of 10 colonies in 30 cycles. The graph le450_5a.

We next turned to graphs for which the success rate of a single ant run is closer to 0. A typical example is the graph le450_5a, with $T = 0.7$ (see Fig. 3 and Table V): 4 ants colonies (out of 10) have reached the 100% success rate, 3 ants colonies had results between 3 and 7, while the other 3 colonies had success rate less than 10%. In one ant colony, no single ant run was successful. It is interesting to note that when increasing the parameter *nants* to 25 the colonies with success rate around 0 disappeared, but on the other hand the most successful colonies did not reach the 100% rate (see Fig. 4).

So far, the results are very promising so that we were tempted to conclude that nearly always the learning is successful. This of course is not likely to be true, as the graph colouring problem is NP-hard. Indeed, one can find examples for which the PW algorithm in reasonably short time gives approximate colourings, which are useless because they are too far from optimal. Another phenomena observed is the following. On some graphs, the learning of colonies is very fast, but after some time, the success rate gets lower. Example of such behaviour is given on Figs. 5 and 6.

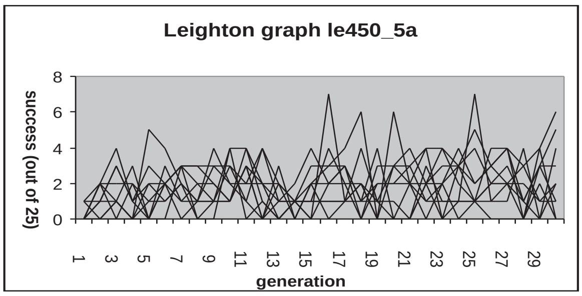

Fig. 4. Average generation successes of 25 ant colonies.

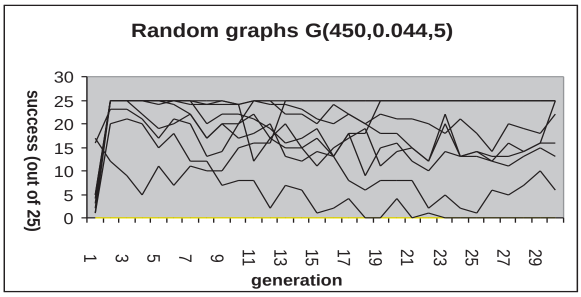

Fig. 5. Average generation successes of 25 ant colonies.

Again, similar to the results on the Leighton graph le450_5a above, the smaller colonies exhibit better behaviour. While the colonies of 10 ants learn in about less than 10 cycles how to colour the graph with success rate 100%, the larger colonies are learning even faster, but after some time, the success rate decreases in some colonies. An intuitive explanation is the following: the reason may be that not too many good solutions guide the convergence less ambigously than a large number of comparably good solutions. The result of adding the virtual edges may in this case be influenced by too many good solutions and the information stored may be ambigous.

IX. Conclusions

In this note we show that on some instances the proposed ants algorithm outperforms both standard restarts (i.e. "ants not cooperating") and long runs (i.e. "a long sin-

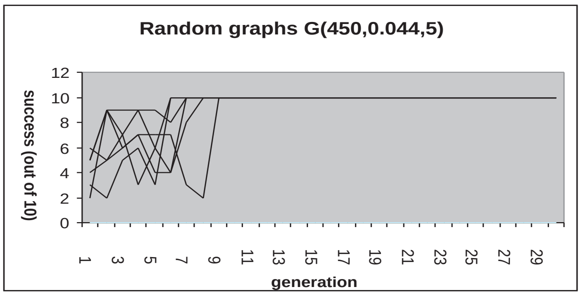

Fig. 6. Average generation successes of 10 ant colonies.

gle ant search"). Observing the behaviour of ants colonies which the graph colouring information in a form of virtual edges, we saw that the colonies of ants indeed learn how to colour a graph with better success rate. There are however examples of graphs on which the results were less satisfactory. It seems clear that in the case when the single ants procedure is unable to produce good approximate solutions with few bad edges, the information accumulated is not reliable and thus can not be expected to be useful. So far we only guess what quality of the solutions provided by single ants is needed if we want to expect successful behaviour of an ant colony. We are going to study this relationship both experimentally and theoretically. In any case, for very difficult graphs, fine tuning of the parameters of the single ants procedure will be neccessary. On the other hand, we note that the parameters *nants* and *ncycles* were guessed here, mainly based on the values used elsewhere [6]. Heuristics with some arguments for particular choices of these parameters may be worth studying which opens another avenue for both experimental and theoretical research.

X. Acknowledgements

This work was supported in part by the IST Programme of the European Com munity under the PASCAL Network of Excellence IST2002506778. The work by the second author has been partially supported by the Ministry of Education, Science and Sport of Slovenia. This publication only reflects the authors' views.

References

[1] R.Battiti and G.Tecchiolli, The Reactive Tabu Search, *ORSA Journal of Computing*, 6 (1994) 126–140.
[2] P. Burge and J. Shawe-Taylor, Adapting the Energy Landscape for MFA, *Journal of Artificial Neural Networks* 2 (1995) 449–454.
[3] P. Burge and J. Shawe-Taylor, Bitstream Neurons for Graph Colouring, *Journal of Artificial Neural Networks* 2 (1995) 443–448.
[4] A. Colorni, M. Dorigo and V. Maniezzo, Distributed optimization by ant colonies, in *Proceedings of the first European Conference on Artificial Life* (F.J. Varela and P. Bourgine, Eds.), (1991) pp.134-142, MIT Press/Bradford Books, Cambridge, Massachusetts.
[5] D. Costa, A. Hertz and O. Dubuis, Embedding of a sequential algorithm within an evolutionary algorithm for coloring problems in graphs, *Journal of Heuristics*, **1** (1995) 105–128.
[6] D. Costa and A. Hertz, Ants can colour graphs, *Journal of the Operational Research Society*, **48** (1997) 295-305.
[7] K. Edwards, The complexity of colouring problems on dense graphs, *Theoretical Computer Science*, 43 (1986) 337–343.
[8] A.G. Ferreira and J. Žerovnik, Bounding the probability of success of stochastic methods for global optimization, *Journal of computers & mathematics with applications*, 25 (1993) 1–8.
[9] C. Fleurent and J.A. Ferland, Genetic and hybrid algorithms for graph coloring, *Annals of Operations Research* 63 (1996) 437–461.
[10] M.R. Garey, D.S. Johnson, *Computers and Intractability*, Freeman, San Francisco 1979.
[11] F. Glover and M. Laguna, Tabu Search, Kluwer, Dordrecht, 1998.
[12] D.S. Johnson, C.R. Aragon, L.A. McGeoch and C. Schevon, Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning, *Operations Research*, 39:378–406, 1991.
[13] D.S. Johnson and M. Trick, (eds.), Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge, American Mathematical Society 1996. DIMACS colouring benchmark instances available via ftp at `rutgers.dimacs.edu`
[14] F. Leighton, A graph coloring algorithm for large scheduling problems, *J. Res. National Bureau of Standards*, **84** (1979) 489-505.
[15] C. Lund and M. Yanakakis, On the Hardness of Approximating Minimization Problems, *Journal of the Association for Computing Machinery*, 41 (1994) 960–981.
[16] C.Morgenstern, Distributed Coloration Neighborhood Search, pp. 335–357 of ref. [13].
[17] A. Petford and D. Welsh, A Randomised 3-colouring Algorithm, *Discrete Mathematics*, 74 (1989) 253–261.
[18] J. Shawe-Taylor and J. Žerovnik, Analysis of the Mean Field Annealing Algorithm for Graph Colouring, *Journal of Artificial Neural Networks* 2 (1995) 329–340.
[19] J. Shawe-Taylor and J. Žerovnik, Adapting temperature for some randomized local search algorithms, Preprint series Dept. Math., University of Ljubljana, 36 (1998) no.614.
[20] B.Rising, J. Shawe-Taylor and J. Žerovnik, Graph Colouring by Maximal Evidence Edge Adding, sumitted.
[21] A.Vesel and J. Žerovnik, How well can ants color graphs?, *Journal of computing and Information Technology - CIT* 8 (2000) 131–136.
[22] J. Žerovnik, A randomised heuristical algorithm for estimating the chromatic number of a graph, *Information Processing Letters*, 33 (1989) 213–219.
[23] J. Žerovnik, A Randomized Algorithm for k–colorability, *Discrete Mathematics*, 131 (1994) 379–393.