

UNIVERSITY OF LJUBLJANA
INSTITUTE OF MATHEMATICS, PHYSICS AND MECHANICS
DEPARTMENT OF MATHEMATICS
JADRANSKA 19, 1000 LJUBLJANA, SLOVENIA

Preprint series, Vol. 36 (1998), 632

HOW GOOD CAN ANTS COLOR
GRAPHS?

Aleksander Vesel Janez Žerovnik

ISSN 1318-4865

October 16, 1998

Ljubljana, October 16, 1998

How good can ants color graphs?

Aleksander Vesel ¹

Department of Mathematics, PEF
University of Maribor
Koroška c. 160, SI-2000 Maribor, Slovenia

Janez Žerovnik ²

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.

Abstract

We compare the ants algorithm for graph coloring recently proposed by Costa and Hertz with the repeated Recursive Largest First algorithm and with a Petford-Welsh type algorithm. In our experiments, the later is much better than the first two.

Keywords: graph coloring, ants algorithm, Petford-Welsh, RLF

1 Introduction

Combinatorial optimization problems arise in situations where discrete choices must be made and solving them amounts to finding an optimal solution among finite or countably infinite number of alternatives. Optimality relates to some cost criterion, which provides quantitative measure of the quality of each solution. This area of discrete mathematics is of great practical use and has attracted much attention over the years.

Many combinatorial optimization problems are NP-hard (see, for example [7], a classical introduction to theory of computational complexity). It is generally believed that NP-hard problems cannot be solved to optimality within times which are polynomially bounded functions of input size.

¹Email: vesel@uni-mb.si

²Email: janez.zerovnik@imfm.uni-lj.si.

Therefore there is much interest in approximation algorithms which can find near-optimal solutions within reasonable running times.

One of the most studied NP-hard problems are the graph coloring problems. Graph coloring problems have numerous applications in scheduling [8] and other practical problems including register allocation, the design and operation of flexible manufacturing systems [15], frequency assignment (see [2] and references there) etc. It is well-known that the problem of determining the chromatic number is NP-hard on arbitrary graphs. Costa and Hertz [5] recently proposed an evolutionary algorithm which imitates the ants behaviour for solving the graph coloring problem. The evolutionary methods use the collective properties of a group of distinguishable solutions, called population.

There are many other approaches to color a graph [9]. In this paper, we have chosen two in order to compare with the ants algorithm.

The ants algorithm proposed in [5] uses the algorithm called Recursive Largest First (RLF) as a subroutine. Therefore it is natural to compare the ants algorithm with a simple repetition of RLF within the same time bounds. The algorithm RLF belongs to the class of constructive methods. They build feasible solutions by starting from an empty solution and by inserting repeatedly a new component into the current partial solution.

The algorithm antivoter was proposed by Petford and Welsh [11]. It is a randomized algorithm based on an analogy to a model of infinite particle systems from statistical mechanics. It can be shown that the algorithm is indeed a metropolis type algorithm and is also equivalent to a generalized Boltzmann machine neural network [12]. In this paper we use an extension of the Petford-Welsh algorithm [18, 2].

Ants can color graphs was a message of the paper [5]. However, experimental results in the cited paper only compare various variants of the ants algorithm. A comparison with RLF is not usable, because the ants algorithms were let run for an order of magnitude larger times. On a basis of such an experiment one can not conclude that the ants algorithms are any better than simple (memoryless) repetition of, say RLF. Unfortunately, such false conclusions can be also found elsewhere, even in [10]. Here we first compare two versions of ANTCOL with repeated RLF. The experiments show that the version of ants algorithm which uses RLF is on average better than simple repetition of RLF, while the repeated RLF algorithm beats the version of ANTCOL using DSATUR subroutine. Then we compare the ants

algorithm with the Petford-Welsh algorithm. The later is in our experiments substantially better then the ANTCOL algorithm.

Although there is a library of benchmark problems for graph coloring available on the world wide web [9], in this experiment we decided to use the instances of random graphs of the same type as in [5] in order to be consistent with our main reference.

The idea of our experiment is to make a "fair" comparison of ANTCOL algorithm with RLF and Petford-Welsh algorithms. We do a rough analysis of algorithm running times in terms of changes of colors of single vertices in order to give comparable running times to all the algorithms. Since ANTCOL algorithms call RLF as a subroutine and it is easy to count the number of calls of RLF we compare a run of ANTCOL with this number of repetitions of RLF. Furthermore, an iteration of the Petford-Welsh algorithm is a change of a color of one vertex and one run of RLF is hence comparable with n (the number of vertices of the graph) iterations of the Petford-Welsh algorithm. The results are summarized in two tables.

The rest of the paper is organized as follows. First we briefly recall the graph coloring problem and generation of random graphs. Then we give details on the algorithms.

Table 1. Comparison of short runs (nants=100, 5000 RLF, 5000 n PeWe). Each entry gives the average number of colors needed in 20 runs. The graphs are random graphs $G_{n,p}$ with parameters $n = 100,300$ and $p = 0.4, 0.5, 0.6$.

n	p	RLF	ANTCOL		Petford-Welsh		
			RLF	DSATUR	$T = 0.2$	$T = 0.3$	$T = 0.4$
100	0.4	13.24	12.65	12.80	12.50	12.05	12.10
	0.5	15.93	15.40	15.65	15.25	14.95	14.95
	0.6	20.85	18.65	18.85	18.35	18.00	18.35
300	0.4	30.05	29.20	36.00	27.95	26.90	28.15
	0.5	37.80	36.30	44.90	34.80	33.40	36.05
	0.6	47.00	44.40	55.00	42.55	41.45	45.55

Table 2. Comparison of long runs (nants=300, 15000 RLF, 15000n PeWe). Each entry gives the average number of colors needed in 20 runs. The graphs are random graphs $G_{n,p}$ with parameters $n = 100, 300$ and $p = 0.4, 0.5, 0.6$.

n	p	RLF	ANTCOL		Petford-Welsh		
			RLF	DSATUR	$T = 0.2$	$T = 0.3$	$T = 0.4$
100	0.4	13.24	12.70	13.95	12.30	12.00	12.00
	0.5	15.79	15.20	16.10	15.05	14.85	14.95
	0.6	20.73	18.65	18.75	18.15	17.90	18.35
300	0.4	30.00	28.90	30.40	27.90	26.70	27.95
	0.5	37.77	35.70	38.10	34.60	33.35	36.10
	0.6	46.60	43.70	47.30	42.25	41.15	45.30

2 Definitions

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 smallest number k for which a proper k -coloring exists is the *chromatic number* of G , $\chi(G)$. An *optimal coloring* is one which uses exactly $\chi(G)$ colors. A set $S \subseteq V(G)$ is *independent* if $xy \notin E(G)$ for any pair of vertices $x, y \in S$.

A *random graph* $G_{n,p}$ is generated as follows. For any pair of vertices u and v the edge uv is inserted with probability p , independently. This is one of the usual random graph models [1].

3 ANTCOL[5]: Ants algorithm for graph coloring

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 [3] first developed Ant System for tackling so called Assignment Type Problems (ATPs). Costa and Hertz [5] adapted the principles

of Ant System to solve the graph coloring problem (which is an instance of ATPs). The heart of their algorithm is a loop with some (they suggested up to 50) iterations. The algorithm builds a feasible solution at each iteration. The solution is produced by a colony of ants. The colonies of $nants = 100$ and 300 ants are used in their experiments. The role of each ant is to color the graph in a constructive way. 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 in a $n \times n$ matrix M (where $n = V(G)$). The value of an entry M_{uv} in M is proportional to the quality of the colorings obtained by giving the same color to u and v . At each step of a constructive method the next vertex is chosen with probability which depends on the values in M .

The authors used randomly generated graphs to test the efficiency of the algorithm. The experiments presented in [5] show that the tuned ANTCOL algorithm using RLF as a constructive method performs better than ANTCOL using DSATUR as a constructive method. The results on random graphs $G_{n,p}$ for $n = 100$ and 300 and for $p = 0.4, 0.5$ and 0.6 with both versions of ANTCOL in Table 1 and Table 2 are taken from [5].

4 Constructive methods: greedy, DSATUR and RLF

Constructive methods for graph coloring are often implemented to quickly produce an upper bound of the chromatic number.

Many constructive methods for graph coloring have been proposed. Most of them color each vertex in turn with the smallest possible color. The quality of the resulting coloring strongly depends on the order in which vertices are examined. The ordering of vertices can be either static or dynamic.

An example of static ordering is to preorder vertices according to their degree. This is known as the Welsh-Powell algorithm [17]. Another example of static ordering is a random (pre)ordering of vertices. If the ordering is random, we call the constructive algorithm the *greedy coloring heuristics*.

The dynamic ordering is generally more effective since the choice of the next vertex to be colored is based on the previous color assignments. The most important methods using dynamic ordering are DSATUR and RLF.

The experiments presented in [5] show that better coloring can be obtained using RLF.

RLF algorithm builds the classes of colors sequentially. The method selects vertices one color class at a time. Assume V_i is the next color class to be completed. Let W be the set of uncolored vertices which can be included in the independent set V_i and let B be the set of uncolored vertices which can no longer be included in the independent set because they are adjacent to at least one vertex in V_i . The algorithm builds V_i as follows:

```

 $B := \emptyset$ ;
choose  $v \in W$  to be a vertex with a maximum degree in  $W$ ;
move  $v$  from  $W$  to  $V_i$ ;
move all vertices of  $W$  that are adjacent to  $v$  from  $W$  to  $B$ ;
While  $W \neq \emptyset$  do
    choose  $v \in W$  of maximum degree in  $B$ ;
    move  $v$  from  $W$  to  $V_i$ ;
    move all vertices of  $W$  that are adjacent to  $v$  from  $W$  to  $B$ ;

```

To compare the ants algorithm and RLF with the same assumptions a comparable repeated RLF algorithm using solely RLF is used. Our RLF algorithm is a simple repetition of 5000 (15000) runs of RLF, and only the best result obtained so far is stored in the memory. This corresponds to 50 iterations of ANTCOL with 100 (300) ants.

We found that RLF was on average worse than ANTCOL using subroutine RLF. However, it is interesting that simple repetitions of RLF were on average better than ANTCOL using DSATUR.

We also did a few experiments with repeated greedy coloring heuristics. Not surprisingly, it appeared to be much worse than repeated RLF.

5 Petford-Welsh type algorithms, PeWe

The algorithm we use is a generalization of the algorithm of Petford and Welsh [11]. 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 [9], frequency assignment "realistic" graphs, and others [11, 18, 19, 13, 2, 16].

In [2] we used the algorithm as given in [18]. Here, instead of starting with a random 2-coloring we start with a coloring obtained by the greedy coloring algorithm. The order of vertices colored by greedy algorithm is random. The main loop is the same as in [18, 2]. We now briefly recall the algorithm for completeness of the presentation.

```

get a random order of vertices;
run a greedy coloring algorithm;
while not stopping condition do
    if the coloring is proper then recolor vertices of the maximal color
    select a bad vertex  $v$  (randomly)
    assign a new color to  $v$ 
end while

```

The greedy coloring always takes the minimal color which does not violate any constraints.

Eventually the algorithm finds a proper coloring with k colors for some k . We hence know that $\chi(G) \leq k$. The role of the first line in the while loop is to reduce the number of colors in this case. All vertices of color k are recolored to obtain a (usually not proper) coloring with $k - 1$ colors. The algorithm continues and tries to find a proper coloring with fewer than k colors.

In the second line of the while loop, a bad vertex is selected uniformly random among bad vertices, i.e. vertices which are endpoints of some edge which violates a constraint.

New color is assigned at random. The new color is taken from the set

$$\{0..maxcolor\}$$

by sampling according to 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 with one endpoint at v and violating the constraint provided color of vertex v is in the set i .

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

The stopping criteria we choose is a time limit (in terms of the number of calls to the function which computes a new color) and the number of colors with which we want to color the graph. The algorithm stops if either of conditions is satisfied. If we do not want to guess the chromatic number, we can simply take 1 or 2 for the desired number of colors. This can never be achieved, so the only stopping condition is the time limit. Since it starts with a greedy coloring we always have a proper assignment (with a number of colors which may be bigger than optimal). This assignment can be regarded as an approximate solution.

Remark: there are two decisions which are of crucial importance for behaviour of the algorithm: to define the number of iterations before restart (or define a stopping criteria) 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 [14].

In order to be comparable with RLF and ANTCOL, we run Petford-Welsh algorithm with $5000n$ iterations and $15000n$ iterations. Each iteration of Petford-Welsh algorithm is a change of color of one vertex. Since a RLF changes all n vertices, one RLF corresponds to approximately n iterations of Petford-Welsh algorithm. The running times are in range from less than a minute for runs of $500n$ iterations on small graphs ($n = 100$) up to 18 minutes CPU for $1500n$ iterations on large graphs ($n = 300$) (on AlphaServer 1000 4/200 VAX/VMS).

We took three values of the parameter T : 0.2, 0.3 and 0.4. These values were guessed, based on our experience with the algorithm on other classes of graphs. More tuning could only improve the performance of the algorithm.

Remark: According to Costa and Hertz, the best average result on $G_{100,0.5}$ is 14.95, obtained in [6] and in [4]. The Petford-Welsh algorithm scored 14.95 with both $T = 0.3$ and $T = 0.4$ in short version. The longer version results were: 14.80 at $T = 0.3$ and 14.90 at $T = 0.4$.

References

- [1] B. Bollobas, *Random graphs*, Academic Press, London 1985.

- [2] B. Chamaret, S. Ubeda, J. Žerovnik, *A Randomized Algorithm for Graph Coloring Applied to Channel Allocation in Mobile Telephone Networks*, in Proceedings of the 6th International Conference on Operational Research KOI'96 (T.Hunjak, Lj.Martić, L.Neralić, eds.), 25-30.
- [3] A. Coloni, M. Dorigo and V. Maniezzo, *Distributed optimization by ant colonies*, *Proceedings of the first European Conference on Artificial Life*, Paris, ed. by F.J. Varela and P. Bourguine, MIT/Press/Bradford Books, Cambridge, Massachusetts, (1991), 134-142.
- [4] 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.
- [5] D. Costa and A. Hertz, *Ants can colour graphs*, *Journal of the Operational Research Society* **48**, (1997), 295-305.
- [6] C. Fleurent and J.A. Ferland, *Genetic and hybrid algorithms for graph coloring*, *Annals of Operations Research* **63**, (1996), 437–461.
- [7] M.R. Garey, D.S. Johnson, *Computers and Intractability*, Freeman, San Francisco 1979.
- [8] F. Leighton, *A graph coloring algorithm for large scheduling problems*, *J. Res. National Bureau of Standards* **84**, (1979), 489-505.
- [9] D.S. Johnson, M. Trick, (editors) *Cliques, Coloring and Satisfiability*, DIMACS Series in Discrete Mathematics and Theoretical computer Science, vol.26, AMS, Rhode Island 1996.
DIMACS coloring benchmarks, available via ftp at `rutgers.dimacs.edu`
- [10] D.S. Johnson, C.A. Aragon, L.A. McGeoch, C. Schevon, *Optimization by Simulated Annealing: an Experimental Evaluation - Part II (Graph Coloring and Number Partitioning)*, *Operations Research* **31**, (1991), 378–406.
- [11] A.D. Petford in D.J.A. Welsh, *A randomized 3-colouring algorithm*, *Discrete Mathematics* **74**, (1989), 253–261.
- [12] J. Shawe-Taylor, J. Žerovnik, *Boltzmann Machines with Finite Alphabet*, *Artificial Neural Networks 2*, vol.1., Brighton, U.K. (1992) 391-394.
- [13] J. Shawe-Taylor, J. Žerovnik, *Analysis of the Mean Field Annealing Algorithm for Graph Colouring*, *Journal of Artificial Neural Networks* **2**, (1995), 329-340.

- [14] J. Shawe-Taylor, J. Žerovnik, *Adapting temperature for some randomized local search algorithms*, Preprint series Dept. Math., University of Ljubljana, **36** (1998) no.614.
- [15] K. Stecke, *Design Planning, Scheduling and Control Problems of Flexible Manufacturing*, Annals of Operations Research **3**, (1985), 3–12.
- [16] S. Ubéda, J. Žerovnik, A randomized algorithm for graph coloring applied to channel assignment problem, in Proceedings of the 4th International Symposium on Operational Research (V.Rupnik, L.Zadnik Stirn and S.Drobne, eds.), Slovenian Society INFORMATIKA - Section for Operational Research, Ljubljana 1997, p.361-366.
- [17] D.J.A. Welsh, M.B. Powell, *An upper bound for the chromatic number of a graph and its application to timetabling problems*, The Computer Journal **10**, (1967), 85–86.
- [18] J. Žerovnik, *A randomised heuristical algorithm for estimating the chromatic number of a graph*, Information Processing Letters **33**, (1989), 213-219.
- [19] J. Žerovnik, *A Randomized Algorithm for k -colorability*, Discrete Mathematics **131**, (1994), 379–393.