Available online at www.sciencedirect.com
ScienceDirect
Discrete Applied Mathematics 156 (2008) 313–324
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

An improved ant colony optimisation heuristic for graph colouring

Kathryn A. Dowsland[a], Jonathan M. Thompson[b,*]

[a] *Gower Optimal Algorithms Ltd. and Nottingham University, UK*
[b] *School of Mathematics, Cardiff University, Cardiff, CF24 4AG, UK*

Received 29 July 2004; received in revised form 27 January 2006; accepted 20 March 2007
Available online 22 May 2007

Abstract

The focus of this paper is an ant colony optimisation heuristic for the graph colouring problem. We start by showing how a series of improvements enhance the performance of an existing ant colony approach to the problem and then go on to demonstrate that a further strengthening of the construction phase, combined with a tabu search improvement phase, raise the performance to the point where it is able to compete with some of the best-known approaches on a series of benchmark problems.

Keywords: Meta-heuristic; Graph colouring; Ant colony optimisation

1. Introduction

Ant colony optimisation (ACO) is a heuristic technique that takes its inspiration from the way in which a colony of ants co-operates with one another when searching for food. Although it was originally conceived in the context of solving routing problems such as the TSP, in recent years there has been increasing interest in experimenting with the technique on other problems. This paper focuses on an ant algorithm for the graph colouring problem and shows how a series of modifications and enhancements can result in a significant improvement in performance when compared to an existing ant colony approach to the problem.

Given a set of vertices V and a set of edges E, the graph colouring problem is that of minimising the number of colours required to allocate a colour to each vertex such that no adjacent vertices are in the same colour. The optimal solution is known as the chromatic number. In [4] Costa and Hertz introduced ANTCOL, an ACO heuristic for the graph colouring problem. Although their results did not match those obtained by the best graph colouring heuristics available at the time, they were sufficiently good to encourage further research. In [9] we investigated the performance of ANTCOL when applied to graph colouring problems arising in the context of examination scheduling and suggested a number of modifications to improve its performance on this class of problems. The result was a graph colouring heuristic that was able to find the best-known colourings quickly and consistently over a wide range of examination timetabling graphs. However, in the examination scheduling problem the number of colours corresponds to the number of timeslots available. We exploited this fact by basing our modifications on *a priori* knowledge of the chromatic number or target value. We also showed that the behaviour of ANTCOL and its modifications on the timetabling graphs did not

* Corresponding author.
E-mail address: ThompsonJM1@cardiff.ac.uk (J.M. Thompson).

0166-218X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2007.03.025

always match that observed by Costa and Hertz on random graphs. Thus it is not clear that our modifications will result in enhanced performance on arbitrary graphs or in instances where the target value is not known. We therefore start the current investigation by showing that the improved version of ANTCOL can be generalised successfully to give enhanced performance on arbitrary graphs. Indeed we show that by strengthening the diversification strategy suggested in [9] we obtain further improvement in solution quality.

However, performance still falls well short of that of other recent meta-heuristic approaches to the problem. We therefore go on to investigate three potential sources of further improvement. Firstly, we strengthen the underlying construction heuristic, using a strategy suggested by Laguna and Martí [16] for a GRASP approach to graph colouring. Secondly, in line with many other ACO implementations we consider the inclusion of a hill-climber or local optimiser. We investigate three such operators ranging from a relatively fast rearrangement of the colours, similar to that suggested by Culberson [6], to a more expensive tabu search approach. Thirdly, given the improvements already made by employing an evaluation function that is able to differentiate between solutions using the same number of colours we investigate a function that allows a further level of differentiation.

In the next section we survey some of the existing literature on graph colouring algorithms, before going on to outline ant optimisation and its application to the graph colouring problem. This is followed by the results of our initial experiments measuring the effect of the enhancements described in [9] on randomly generated graphs. We then go on to suggest a series of further enhancements and to empirically identify the best of these using a small set of benchmark problems. This is then compared with the results of other published methods on a wider range of problems. The paper ends with some comments and suggestions for further research and enhancements.

2. Graph colouring

Graph colouring is a widely researched classical optimisation problem. The purpose of this section is not to provide a comprehensive survey of solution approaches, but to give a brief overview of the type of approaches that have proved successful and to introduce those strategies that are pertinent to the remainder of this paper.

Construction heuristics were proposed by, among others, Brélaz [2] and Leighton [17] whose methods (RLF and DSATUR, respectively) were the basis of the ANTCOL heuristic. The iterated greedy method of Culberson [6] is based on a series of constructions, each one guaranteed to produce a solution that is at least as good as its predecessor. This is achieved by permuting the colour classes and then taking the vertices in colour class order to be coloured in the next construction phase. Subsequent work used meta-heuristics to improve solution quality. Hertz and de Werra [14] consider infeasible colourings and use tabu search to reduce the number of adjacent vertices in the same colour class to zero. Chams et al. [3] compare pure simulated annealing with a hybrid method where a variant of RLF is used to colour some vertices and simulated annealing colours the rest.

Recently several researchers have been successful with multi-start local search approaches. Examples include the GRASP approach of Laguna and Martí [16] and the iterated local search of Paquete and Stützle [18]. In [16] the greedy phase builds a solution, colour by colour, by selecting vertices from a candidate list. However, rather than simply adding a new colour class to the solution once no further vertices can be added to the current class, several attempts at building the current class are made, and the one resulting in the lowest number of edges in the graph induced by the remaining vertices is selected. The improvement phase combines the two smallest colour classes into one and the descent reduces the number of clashes. In [18], when the search reaches a local optimum a new solution is obtained by applying a (fairly disruptive) perturbation operator to the current local optimum. Four different perturbations are suggested, each one of which includes some randomisation. In line with recent trends in local search there has also been some interest in the use of large neighbourhoods. One such example is Avanthay et al. [1] who consider neighbourhoods which change the colour of sets of conflicting vertices and those which change the colours of sets of vertices in certain colour classes, as well as more disruptive neighbourhoods. A second example is Glass and Prügel-Bennett [13] who propose an exponential permutation neighbourhood in which a subgraph of vertices is selected and the colours within the subgraph are permuted in such a way as to minimise the number of clashes. As the problem of finding the optimal point in the neighbourhood can be modelled as a linear assignment problem the neighbourhood can be searched in polynomial time. However, both authors report only limited success with the larger more disruptive neighbourhoods, suggesting that further research is necessary if such approaches are to be effective.

Genetic algorithm approaches to graph colouring have tended to focus on hybridisations. Costa et al. [5] combine a genetic algorithm with descent, and Fleurent and Ferland [10] hybridise a genetic algorithm with tabu search. Both

use a union crossover where each vertex in the child is allocated its colour from one of the parents, according to which causes the smallest increase in nearby conflicting edges. Fleurent and Ferland improved their solutions by identifying K independent sets first and then using their procedure on the remaining vertices. Costa et al. employ diversification functions altering the number of colours, re-ordering the vertices according to the current solution and solving by using a greedy algorithm and removing partial colourings and re-colouring the residual vertices. Galinier and Hao [11] also hybridise a genetic algorithm with tabu search, but use a crossover operator based on selecting entire colour classes from each parent in turn. This continues until the required number of colour classes has been produced when the remaining vertices are coloured randomly. The tabu search element then attempts to repair the child solution by removing as many clashes as possible. This algorithm was shown to produce the best-known solutions for a series of benchmark problems. Glass and Prügel-Bennett [12] demonstrated that comparable results could be produced using vertex descent instead of tabu search, showing that the power of the algorithm is in the crossover operator. However, their version considerably increases the computational time.

3. Ant optimisation and its application to the graph colouring problem

Ant colony optimisation (ACO) was first suggested by Dorigo et al. [7,8] as a solution technique for the travelling salesman problem. The approach is inspired by the way in which ants use a pheromone trail to communicate and co-operate when identifying feeding sources close to their nest. As they search for food each ant leaves a trail, and subsequent ants tend to follow trails laid by other ants with a bias towards stronger trails. Ants travelling to food sources nearer the nest will return more quickly, thus enabling them to make more journeys in a given period of time, which in turn leads to stronger trails. This will encourage more ants to follow the trail to the nearest source, thereby strengthening it further. This effect is reinforced by the fact that trails evaporate over time. In essence Dorigo's analogy between the ants' behaviour and a solution approach to an optimisation problem involves mapping the ants' choice of which trail to follow onto the decisions made in a probabilistic greedy construction heuristic. The probability of selecting a particular option changes over time according to a 'trail' factor, which is determined by the quality of previous solutions in which the same decision was made. Although many species of ants are essentially blind, meaning that a true analogy would start with a completely random construction, it is usual to add some sort of greedy element to the construction approach. This is usually referred to as the visibility function. In order to implement a basic ACO algorithm for a given problem all that is necessary is to define a greedy construction heuristic that lends itself to randomisation and an appropriate definition of the trail. ACO heuristics usually work with a fixed population of *nants* ants and run for a number of cycles. In the first cycle each ant constructs a solution using the randomised greedy algorithm, with the probability of selecting any option at each stage in the construction being proportional to the visibility of that option. Each solution then contributes to the trail, with better solutions contributing stronger trails than poorer ones. The trails are then used to adjust the probabilities in the next construction cycle. At the end of each cycle the current trail values are reduced by a factor known as the evaporation rate, before being augmented by the trail values relating to the current cycle. We now formalise these ideas with respect to ANTCOL.

In the case of the graph colouring problem Costa and Hertz [4] considered construction approaches based on the two well-known graph colouring algorithms RLF (recursive largest first, Leighton [17]) and DSATUR (Brélaz [2]). Both their results and those in [9] suggest that implementations of ANTCOL based on RLF outperform those based on DSATUR. We therefore restrict our treatment in this paper to the RLF variants. In RLF colour classes are built up one at a time as maximal independent sets. Thus at each stage of the construction process the only decision is which of the feasible vertices should be added to the current set. Three different rules are suggested, giving rise to three different definitions of visibility $\eta_{ik,}$ for colouring vertex i in the current colour, k. These are defined as follows.

Let W be the set of uncoloured vertices that can be included in the current colour class, B the set of uncoloured vertices that cannot be included in the current colour class, and $deg_X(i)$ degree of vertex i in the subgraph induced by the subset of vertices, X.

Then the three options are: (1) $\eta_{ik} = deg_B(i)$, (2) $\eta_{ik} = |W| - deg_W(i)$, and (3) $\eta_{ik} = deg_{B\cup W}(i)$. As all three options depend on the vertices already coloured in the current colour, two different rules are suggested for the first vertex in each colour. The rule of random selection consistently resulted in better performance in both [4,9], and is therefore adopted for the remainder of this paper.

As the quality of the colouring depends on the sets of vertices that are placed in the same colour class, as opposed to the actual colour allocated to each individual vertex, Costa and Hertz define a trail matrix in terms of vertex pairs.

In the first cycle the trail between each pair of non-adjacent vertices is set equal to 1. At the end of each cycle the trail matrix is updated according to $t_{ij} = \rho t_{ij} + \sum_{s \in S_{ij}} 1/q(s)$, where S_{ij} is the set of solutions in which i and j were coloured in the same colour, $q(s)$ is the number of colours required in solution s and ρ controls the rate of evaporation. During the construction process the trail associated with colouring vertex i in colour k depends on the set of vertices already coloured k, denoted V_k, and is given by

$$\tau_{ik} = \frac{\sum_{j \in V_k} t_{ij}}{|V_k|}.$$

At each stage the option of colouring vertex i in the current colour k is selected with probability

$$P_{ik} = \frac{\tau_{ik}^{\alpha} \eta_{ik}^{\beta}}{\sum_{j \in W} \tau_{jk}^{\alpha} \eta_{jk}^{\beta}} \quad \text{if } i \in W, \quad P_{ik} = 0 \text{ otherwise,}$$

where α and β are parameters that control the balance between the influence of the trail and visibility, and W is the set of vertices that are still feasible for colour k. The ant algorithm can be summarised as follows:

```
Procedure ANTCOL
t_ij = 1 ∀i, j = 1, |V|, i ≠ j                    (initialise trail matrix)

for cycle = 1, ncycles
    δ_ij = 0 ∀i, j = 1, |V|, i ≠ j                (initialise trail update matrix)
    for ant = 1, nants
        X = V;                                    (initialise set of uncoloured vertices)
        k = 0;                                    (initialise number of colours used)
        while X ≠ ∅ do
            k = k + 1:
            C_k = ∅                               (initialise colour class k)
            F = X;                                (initialise set of vertices still feasible for colour k)
            Select i ∈ F with probability 1/|F|;
            Call COLOUR_VERTEX(i, k);
            While F ≠ ∅ do
                                  Select i ∈ F with probability P_ik;
                                  Call COLOUR_VERTEX(i, k);
            End while
        End while
        δ_ij = δ_ij + 1/k ∀i, j : C_i = C_j, i ≠ j;        (update trail update matrix.)
    next ant
    t_ij = ρt_ij + δ_ij ∀i, j = 1, |V|, i ≠ j;             (update trail matrix)
next cycle

Procedure COLOUR_VERTEX(i, k)
X = X\{i}
C_k = C_k ∪ {i}
F = F\(Γ_F(i) ∪ {i})              (where Γ_F(i) denotes the set of i's neighbours in F)
```

In [9] we suggest that a potential problem with ANTCOL may be that the differentiation between good and not so good solutions depends on the chromatic number, with smaller variation occurring if the chromatic number is large. We therefore introduced an evaluation function

$$f_1(s) = \frac{1}{q(s) - r},$$

where r is the chromatic number. However, this function also has its drawbacks in that it fails to differentiate between solutions that have just one or two vertices in the smallest colour class and those solutions in which the vertices are distributed evenly among all the colours, although the former is intuitively closer to a better solution than the latter.

We therefore introduced evaluation function $f_2(s)$ as follows. The construction phase continues until r colour classes have been completed and the evaluation function is given by

$$f_2(s) = \frac{1}{u(s)},$$

where $u(s)$ is the number of uncoloured vertices. Under these conditions no trail will be generated relating to the uncoloured vertices. We compensate for this with a diversification strategy by calculating a trail multiplier $tm(i)$, for each vertex i in each cycle. $tm(i)$ is initialised at 1 for each vertex in each cycle and is increased by $f_2(s)$ if i remains uncoloured in solution s. P_{ik} is then increased to $P_{ik} \times tm(i)$ for all i in the roulette wheel selection. Empirical studies on a range of graphs derived from examining scheduling data showed that each of these modifications improved over the original ANTCOL algorithm, with a combination of an evaluation function f_2 together with the diversification strategy giving the best results, with $\alpha = 2$ and $\beta = 5$ the best values for the weighting parameters [9].

This strategy cannot be applied directly to the arbitrary case where the chromatic number r is not known. In [9] we suggested that this might be overcome by defining r as a dynamic target based on the current upper bound on the solution. r is initialised as being equal to the number of vertices, and reset at the end of each cycle as $r = Q^* - 1$, where Q^* is the minimum number of colours achieved to date. As r is no longer the global target value the algorithm will not terminate when $u(s) = 0$ (i.e. a solution in r colours has been obtained). We therefore need to define a suitable trail factor for the case $u(s) = 0$ as $f_2(s)$ is undefined at this point. We would like the differential between $f_2(s)$ when $u(s) = 1$ and 0 to be greater than the factor of 2 between $f_2(s) = \frac{1}{2}$ when $u(s) = 2$ and $f_2(s) = 1$ when $u(s) = 1$ so we define $f_2(s) = 3$ when $u(s) = 0$. We will denote this version of ANTCOL with diversification and bound information ANTCOL(DB).

A second significant observation from [9] was that while Costa and Hertz found that there was little difference between using greedy strategies 1 and 2 on the randomly generated graphs, we found that RLF(2) performed extremely poorly on timetabling graphs. Thus we cannot assume that our observations concerning the improvements to ANTCOL will carry over to randomly generated graphs. Therefore the first set of experiments in this study was designed to measure the effects of the enhancements on some of the smaller randomly generated graphs in the public domain. These results are given in the next section.

4. The behaviour of ANTCOL(DB) on random graphs

The experiments described in this section were designed to measure the improvements obtained by our modifications to ANTCOL over the original Costa and Hertz algorithm on randomly generated graphs. We therefore use the same set of graphs as used in their original study. These are randomly generated graphs with 300 and 500 vertices with 50% density. They were first generated by Fleurent and Ferland [10] and can be accessed at http://mscmga.ms.ic.ac.uk/jeb/orlib/colourinfo.html. In [9], although we found the diversification strategy of multiplying the roulette wheel probabilities by $tm(i)$ worked well, no further experimentation with this strategy was carried out. Here we investigate whether or not further benefit can be gained by strengthening the influence of $tm(i)$ by raising it to a higher power i.e. using $tm(i)^\gamma$. Values of $\gamma = 1, 2, 3$ and 4 were tried. We also note that while we found that RLF(2) performed poorly on timetabling graphs, Costa and Hertz highlighted RLF(2) as the best performer of the RLF options on small graphs and therefore used this for the remainder of their experiments. Our own experiments on graphs with 100 and 300 vertices confirmed that while the performance of the two greedy options was similar, there were some indications that RLF(2) was slightly better. We therefore use this option for the remainder of this paper unless it is explicitly stated otherwise. In line with the exhaustive experiments carried out by Costa and Hertz on 100 vertex graphs we ran our experiments with $nants = 100$ for 100 cycles for the 300 vertex graphs and up to 200 cycles for the 500 vertex graphs. This differs from the results on larger graphs quoted by Costa and Hertz where the best results were obtained for 300 ants over 50 cycles. We therefore reran the Costa and Hertz version of ANTCOL using these parameters to allow a fair comparison across all methods. The results are shown in Table 1. We can see that our modified version of ANTCOL improves over Costa and Hertz's original version for all values of γ, with performance improving as γ increases from 1 to 3, but falls off for larger graphs when $\gamma = 4$.

In spite of these improvements over the original version of ANTCOL, the algorithm needs further enhancement if it is to compete with other meta-heuristic approaches to the problem. This is addressed in the next section.

Table 1
Mean number of colours for 300 and 500 vertex graphs for ANTCOL and ANTCOL(DB)

Vertices	ANTCOL	ANTCOL(DB)			
		$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 4$
300	35.7	34.5	34.5	34.4	34.2
500	54.8	53.2	53.2	52.4	52.8

5. Enhancing the algorithm

There are a variety of possibilities for improving the performance of ANTCOL. Relatively minor changes that have proved worthwhile in ACO approaches to other optimisation problems include parameter changes, the use of an elitist strategy, reducing the number of ants contributing to the trail, or using candidate lists in the selection procedure. While these may result in some improvement to a final implementation we felt that on their own such changes would not yield the necessary level of improvement. We therefore elected to seek enhanced solution quality by improving the individual solutions produced by the ants, and through the influence of the evaluation function. In the case of the individual solutions improvements were generated in two distinct ways; by modifying the construction process, and by local improvement of the output from the construction phase.

5.1. Improving the construction phase

Our results on examination scheduling graphs suggested that very good results can be obtained with the right construction algorithm. The most obvious way of achieving this is to use a different visibility function. However, as Costa and Hertz experimented with two of the most promising candidates we were unable to identify any promising new functions. Instead we adopt an approach suggested by Laguna and Martí [16] in their GRASP implementation for colouring sparse graphs. As in ANTCOL Laguna and Martí use a probabilistic greedy construction phase to build up successive colour classes of maximal independent sets, in their case based on $deg_{W \cup B}$. They strengthen this process by making several attempts at each colour class and selecting the one that results in the minimum number of edges in the graph induced by the remaining uncoloured vertices. The rationale behind this is that on average the lower this value, the lower the chromatic number. This strategy is easily incorporated into ANTCOL and will be referred to as the *multi-sets* operator.

5.2. Changing the evaluation function

We have already noted that it can be advantageous to use the number of uncoloured vertices instead of the number of additional colours as the evaluation function used to determine the strength of the trail, as this gives a greater degree of differentiation between different solutions. Here we take this one stage further. For two solutions with the same number of uncoloured vertices a differentiating factor might be the number of clashes that would result if these vertices were inserted into an existing colour class. Intuitively a solution with a larger number of clashes is further from a feasible solution than one with just one or two clashes. We therefore introduce a new evaluation function as follows. Let C_1 to Cr be the sets of vertices in the r colour classes and let U be the set of uncoloured vertices. For $u \in U$ and $k = 1$, r define $e(k, u)$ to be the number of vertices in colour class C_k that clash with u i.e. $e(k, u) = |\Gamma(u) \cap C_k|$, where Γ denotes the set of neighbouring vertices. For each $u \in U$ in turn place u in that C_k that minimises $e(k, u)$ and update $e(k, v)$ for any $v \in U$ still unassigned.

Let $w(k)$ be the number of edges between vertices in C_k and define the new evaluation $f_3(s)$ function as

$$f_3(s) = \frac{1}{\sum_{k=1}^{r} w(k)}.$$

(Note: this value may depend on the order in which the vertices in U are considered. However, as it was anticipated that the final version of the algorithm would use a local improver that would move these vertices around, it was not considered worthwhile to introduce a more complex procedure to ensure that the number of clashes was minimised.)

We also note that when using this evaluation function all the vertices are included in one of the r colour classes and the diversification operator $tm(i)$ is therefore redundant.

5.3. Adding a local improvement operator

Three local search algorithms were considered to improve the current solution. The first two are designed to try and colour one or more of the uncoloured vertices using evaluation function $f_2(s)$ while the third is a more time-consuming tabu search designed for use with objective function $f_3(s)$. We will denote this family of modifications ANTCOL(LS).

5.3.1. Shuffle operator

We denote our first improvement operator as the shuffle operator as it 'shuffles' the vertices around by reordering the colour classes and then moving vertices into a colour class earlier in the order if appropriate. It works as follows.

Suppose we have a colouring in r colours using colour classes C_1 to C_r. Let $k(i)$ denote the index of the colour class containing vertex i. Then the following procedure always results in a colouring in r or less colours.

Recolouring procedure:

Select any permutation of the colours.

Select any ordering of the vertices such that i precedes j if $k(i)$ precedes $k(j)$ in the permutation of the colours.

Colour the vertices in this order using the 'first available colour'.

We use this procedure in an attempt to make room for the uncoloured vertices as follows. Let U be the set of uncoloured vertices. For each colour class, C_k, let $f(k)$ be the number of vertices C_k that are adjacent to at least one vertex in U i.e.

$$f(k) = \left| \bigcup_{u \in U} \Gamma(u) \cap C_k \right|.$$

Order the colour classes in non-increasing order of $f(k)$. Let $C_{\sigma(i)}$ denote the ith colour class according to this ordering. Then

Recolouring procedure with attempted inclusion of vertices in U:

For $i = 1, r$,

Take the vertices of $C_{\sigma(i)}$ in any order and colour in the first available colour.

If $C_{\sigma(i)}$ is feasible for any $u \in U$ allocate u to $C_{\sigma(i)}$ and remove u from U.

If $U = \emptyset$ stop.

Next i.

The advantage of this operator is that it is fast and often moves vertices in groups, thus preserving large groupings from the original construction. Intuitively this will be advantageous from the point-of-view of ACO as it will preserve many of the decisions influenced by the trail. However, apart from sorting the colour classes according to $f(k)$ there is little to guide the procedure towards finding a better solution. The tree operator, described in the next section is designed to be more focussed in this respect.

5.3.2. Tree operator

The tree operator is so called because the series of vertex re-colourings it generates can be represented as a tree. The nodes represent vertex colour pairs (i, k), indicating that vertex i is to be recoloured in colour k. This node then generates branches representing the set of vertices in colour k that must now be recoloured because they are neighbours of i. The first level of the tree is defined by the set of uncoloured vertices U. For each $u \in U$ we select a colour class C_k and swap u with u's neighbours in C_k i.e. set $C_k = C_k \cup \{u\} \backslash \Gamma_k(u)$ and $U = U \cup \Gamma_k(u) \backslash \{u\}$, where $\Gamma_k(u) = \Gamma(u) \cap C_k$. The next level of branches is then defined by the new set U, etc. The tree operator searches part of this tree using a depth first search implemented in the form of a stack of uncoloured vertices, where the next vertex to be assigned to a colour class is determined by a 'last in first out' policy. C_k is selected randomly from those with the minimum number of neighbours i.e. those for which $\Gamma_k(u)$ is minimised. We avoid cycling and limit the time taken by the search by only allowing each vertex to be placed in the set U at most once. When selecting C_k for the current choice of u we therefore define as tabu any C_k for which $\Gamma_k(u)$ contains any vertex that has previously belonged to U. When all C_k are tabu for

all vertices on the stack the process terminates and the node corresponding to the minimum sized set U is returned as the updated solution.

5.3.3. Tabu search operator

The tabu search improver is a minor modification to that used by Galinier and Hao [11]. The solution space is the set of partitions of the vertices into r colour classes, the evaluation function is defined by the number of edges between two vertices in the same colour class and a neighbourhood move involves moving one vertex to another colour class. Valid moves are limited to a candidate list of vertices defined as those vertices making a positive contribution to the cost. The tabu list consists of vertex colour pairs such that a move returning vertex i to a previously allocated colour C_k is tabu for tl iterations, where the tabu tenure, tl, is a random variable. We have followed the suggestion of Galinier and Hao and used $tl = R(10) + 0.6 \times N_{cand}$, where N_{cand} is the number of candidates for moving in the current solution and $R(10)$ is a random integer between 1 and 10. Tabu tenure can be overwritten if a move meets the aspiration criterion of bettering the best solution found to date. The search is initialised by placing each uncoloured vertex into the colour class giving the least number of clashes i.e. by applying the procedure used to calculate cost function f_3. Our search differs from that of Galinier and Hao in that we accept any improving solution as soon as we find it. Our reason for this is to speed up the search. Other researchers including Resende and Ribeiro [19] who tackle a variety of problems have shown that such a strategy achieves the same quality of solution but in a reduced solution time. In our case there is the added motivation that the neighbourhood size depends on the number of candidate vertices so the computational effort required will tend to reduce as the cost function reduces. Our rationale was to reduce the cost function as quickly as possible early on in the search to take advantage of this.

6. The experiments

We carried out a series of experiments to measure the effect of each of our modifications based on a number of benchmark problems that had been used by other researchers. All times quoted are CPU seconds on a Pentium IV 2400. Our first objective was to determine which of the modifications was the most promising for further experimentation. Because of the large number of variants these experiments were carried out on just one set of relatively small graphs—namely the $G_{300,0.5}$ graphs used by Costa and Hertz [4] and Costa et al. [5]. Even with this limited test-bed the computational effort required by the experiments described in Section 4 suggests that a faster implementation is called for. An obvious way of achieving this is to reduce either the number of ants or the number of cycles. In spite of Costa and Hertz's observations that results improved when the number of cycles were reduced and the number of ants increased, this is likely to lead to slower convergence as it takes longer for the trail information to feed back into the construction process. In our case this was exacerbated by the fact that the bound information was not updated until the end of the cycle. We therefore carried out some experiments with smaller numbers of ants and compensated for this by lowering the evaporation factor given by $(1 - \rho)$. $Nants = 10$ and $\rho = 0.75$ seemed to provide a good compromise in terms of improved speed without significant loss of solution quality. However, a potential problem of running with less ants is that of premature convergence. It therefore makes sense to apply several restarts rather than allow one single long run. In what follows we therefore define a single solution attempt to be up to *nstart* starts, each run for *ncycle* cycles, where *nstart* and *ncycle* are parameters that depend on the size of the problem.

Our first set of experiments was designed to compare different variants based on the f_2 objective and to determine a suitable value for m, the number of constructions for each independent set in the case of the *multi-set* operator. Sixteen variants of the algorithm were run 5 times on each of the 10 data sets using $nstart = 5$ and $ncycle = 100$. The 16 variants were the result of running the *multi-sets* option with $m = 1, 2, 3$ and 4, without any local search, with the shuffle operator alone, with the tree operator alone, and with both. Note that $m = 1$ with no improvement operator is equivalent to the basic algorithm as reported in Table 1 but with different parameters for *nants* and ρ. As with any combination of ant algorithm and local search improvement operator a decision has to be made as to how often the local search improver is used. As we felt it was more important to improve good solutions than poorer ones in each case we applied the relevant improvement strategy with probability $P = f_2(s)$ i.e. with a probability that is inversely proportional to solution quality. As both local search operators involve a probabilistic aspect resulting from the resolution of ties for the best colour, there may be some benefit in applying the operator a second time. We do this only if $f_2(s) = 1$. Where both operators are applied the shuffle operator is applied first. Statistical estimates for the $G_{300,0.5}$ graphs give an estimated chromatic number of 35 but Costa et al. colour 7 of them in 33 colours. As our objective at this stage was simply to

Table 2
Performance of different variants of ANTCOL(LS) on 300 vertex graphs

m	$f_2(s)$ = (1 / no. of uncoloured vertices)					$f_3(s)$ = (1 / no. clashes)	
	No LS	Shuffle $P = f_2(s)$	Tree $P = f_2(s)$	Both $P = f_2(s)$	Tree $P = 1$	Tree $P = 1$	TS $P = 1$
a. *Number of times a solution in 34 colours is achieved*							
1	0	0	0	0	50	42	50
2	13	32	31	40	50	50	50
3	29	42	49	49	50	50	50
4	37	48	50	50	50	50	50
b. *Number of times a solution in 33 colours is achieved*							
1	0	0	0	0	14	4	42
2	0	0	0	0	19	13	48
3	0	0	1	5	19	18	49
4	0	3	6	14	16	15	50

differentiate between the different methods each attempt was stopped when 33 colours was reached so we do not know if any of the options could produce a solution in 32 colours. The results in terms of the number of times each variant found a solution in 34 and 33 colours is given in columns 2–5 in Table 2.

The results suggest that both the *multi-sets* operator and the two local search operators enhance solution quality. The tree operator is stronger than the shuffle operator and there is little benefit to be gained in using both. Observations of the behaviour of the two local search operators made during the search showed that, while the shuffle operator was effective when the target value, r, was relatively high and there was some slack in the solutions, it had little effect in the latter stages, whereas the tree operator remained effective throughout. The results also show that increasing m in the *multi-sets* option seems to improve the results. Although this obviously increases the time per cycle, the faster convergence means that overall solution times are improved. For example, all four values of m result in solutions of 33 colours 5 times out of 5 for the first $G_{300,0.5}$ graph and the mean solution times range from 119.8 s when $m = 1$ to 87.4 s when $m = 4$. Limited experiments with higher values of m did not yield any further improvements for these graphs. Given the success of the tree operator one further set of experiments were carried out—again using $m = 1, 2, 3$ and 4, but calling the tree improver for every solution i.e. with probability $P = 1$. These results are shown in column 6 of Table 2. Here values of $m = 2$ or 3 perform better than $m = 4$—possibly due to premature convergence, but overall solution quality has improved suggesting that a more aggressive use of an improvement operator is worthwhile. Our overall recommendation would therefore be to use the tree operator on every solution with a value of $m = 2$ or 3.

Having established that some form of local operator is worthwhile and that the tree operator dominates the shuffle operator the next set of experiments concern the use of the evaluation function f_3 with the tree operator and the tabu search operator. In the case of the tabu search operator it is necessary to define the number of moves allowed for each search. For these experiments we used a value of $L = 2 \times |V|$. The results are given in columns 7 and 8 of Table 2 and show that there is little difference in solution quality when using the f_2 or f_3 with the tree operator, especially for the higher values of m. However, the gains when using the tabu search operator are considerable.

The results for the $G_{300,0.5}$ graphs are somewhat better than those given by Costa et al. in terms of solution quality. The mean solution time to reach a colouring in 33 colours using $m = 4$ was 87.2 s for data set 1 and 165.84 s across all data sets. Costa et al. quote a mean of 11,920.1 s over the 7 (out of 10) attempts that achieved a solution of 33 colours. However, given that their work was carried out more than 10 years ago it is not meaningful to compare these times directly. Given the success of our modified ANTCOL on these graphs, our next set of experiments was designed to compare its performance with other published methods on a range of commonly used benchmark graphs. These include the four $G_{500,0.5}$ graphs generated by Fleurent and Ferland and one $G_{500,0.5}$ graph generated by Johnson et al. [15] all of which were also used by Costa et al., and 4 other graphs from the DIMACS challenge. Table 3 shows the results using the standard parameter settings outlined in the last section, compared with the results achieved by Galinier and Hao [11], and Costa et al. [5].

The results are reasonable, but fail to match those of the best published algorithms in almost all cases, although it should be noted that the results quoted by Galinier and Hao were based on the best of a series of different settings for the parameter L, which differed from graph to graph. We therefore carried out further analysis based on 4 graphs to

Table 3
Performance of ANTCOL(LS) using tabu search with evaluation f_3, $m = 4$, $L = 2|V|$ on benchmark graphs

Graph	GA Approach		ANTCOL(LS)			
	No. colours	No. successes	No. colours	No. successes	Mean time (s)	Mean no. of cycles
Le450_15c	15[a]	6/10	15	5/5	544	160.2
Le450_25c	26[a]	10/10	28	5/5	196	42.6
Flat300_28	31[a]	6/10	32	5/5	79	32.8
DSJC250.5	28[a]	9/10	29	5/5	40	24.8
$G_{500,0.5}$	49[b]	5/5	50	24/25	1394	129.0
			49	2/25	6222	480.0

[a]The result from the GA approach is of Galinier and Hao [11].
[b]The result from the GA approach is of Costa et al. [5].

Table 4
Comparison of ANTCOL(LS) using tabu search with evaluation f_3, and best parameter values

Graph	Results from GA approach	Results from ANTCOL(LS)	Mean time (s)	Mean no. of cycles	Best setting of parameters		
					Visibility	m	L
Le450_15c	15 (6/10)[a]	15 (5/5)	43	10.6	$deg_B(i)$	4	900
Le450_25c	26 (10/10)[a]	26 (5/5)	365	67.2	$deg_B(i)$	5	900
		25 (4/5)	15,286	682.0	$deg_B(i)$	10	10,000
Flat300_28	31 (6/10)[a]	31 (5/5)	361	43.6	$deg_W(i)$	5	5000
DSJC500.5	48 (5/10)[a]	49 (5/5)	2008	71.8	$deg_W(i)$	5	10,000
DSJC250.5	28 (9/10)[a]	28 (5/5)	440	51.2	$deg_B(i)$	5	5000
$G_{500,0.5}$	49 (5/5)[b]	49 (20/20)	1811	269.7	$deg_W(i)$	5	10,000

The numbers in columns 2 and 3 are number of colours (number of successes/number of trials).
[a]Galinier and Hao [11].
[b]Costa et al. [5].

see if further parameter tuning would improve the results. Observation of the progress of the algorithm suggested that the ant algorithm mechanism was working reasonably well in that solution quality tended to improve as the algorithm progressed and there was clear evidence of convergence. For example, the percentage of vertex pairs that appeared in the same colour class in every solution of a given cycle increased as the algorithm progressed. However, the rate of increase varied considerably from graph to graph. Although some improvement might have been obtained by changing parameters such as α and β we chose to concentrate our analysis on another important factor—the quality of solutions resulting from the construction phase, both before and after the improvement phase. There are three ways of influencing the quality of the individual solutions—the definition of visibility, the value of m, and the value of L. Changes in each of these were considered for the 4 graphs in question. Those leading to significant improvements in each of the 4 graphs are discussed below, and results from the best set of parameters for each of these graphs are given in Table 4.

The Leighton graphs: Performance on the Leighton graphs was greatly improved by changing the definition of the visibility from $deg_W(i)$ to $deg_B(i)$. This is not surprising. The structure of the Leighton graphs is such that the optimum number of colours is equal to the size of the largest clique. In [9] we conjectured that the reason for the poor performance of RLF(2) on the examination timetabling graphs was that they also tend to have a large clique of cardinality equal to or just short of the chromatic number, and that the vertices in these cliques tend to have large degree. Visibility given by RLF(1) will tend to favour such vertices, whereas RLF(2) will tend to favour vertices of small degree. In the case of Le450_25c using $m = 4$ resulted in consistent solutions in 26 colours, in a mean time of 437.6 s and a mean number of cycles of 89.0. Increasing m to 5 resulted in improved running speed, and, without increasing the number of cycles, higher values of m resulted in longer running times to reach solutions of equal quality. However, increasing m to 10, L to 10,000 and *ncycle* to 300 resulted in solutions in 25 colours being found in 4 of the 5 runs. This is a better result than that of Galinier and Hao but was achieved at the expense of a mean run-time of 15,285.5 s.

Flat300_28: Changing the visibility made little difference to Flat300_28. However, improvements did result from increasing both m and L. With $m = 4$ and L=10,000 all 5 runs converged to solution of 31 in less than 100 cycles, with

a mean run-time of 767.2 s. Lower values of L were able to produce the same quality but with longer mean solution times. The best results occurred when m was increased to 5 and L to 5000 with a mean time of 409 s, when all 5 runs converged to 31 (as compared with 6 out of 10 for Galinier and Hao).

DSJC500.5: As with the Flat300_28 graph the choice of visibility function has little effect on solution quality. Here the best results came from increasing L to 10,000 (the maximum value tried) when 5 out of 5 runs converged to 49 colours. In this case the results are inferior to those of Galinier and Hao who obtained a solution of 48 for 5 out of 10 runs. Experimenting with higher values of m failed to improve the results further.

The results with the 'tuned' version of the algorithm compare well with those of other approaches, especially on the structured Flat and Leighton graphs. On the Johnson graph the results fall one short of that obtained by Galinier and Hao, but they do improve on the pure tabu search implementation of Hertz and De Werra [14], suggesting that even with a high value of L the ant algorithm component is playing an important part. In the light of the improvements gained on the above graphs DSJC250.5 and the remaining four $G_{500,0.5}$ graphs were run with new parameters. These results are reported in the last two rows of Table 4. Improvements for the $G_{500,0.5}$ graphs are in line with those obtained for the DSJC500.5 graph. DSJC250.5 also benefited from increases in m and L. However, what is surprising about this graph, in that it differs from the observations for the other random 50% density graphs, is that using a visibility based on deg_B is better than using that based on deg_W. Indeed the former outperformed the latter for all combinations of m and L.

With the right choice of parameters ANTCOL(LS) is very successful in terms of solution quality. When allowed a reasonable amount of computation time it is able to match the quality of results produced by Galinier and Hao on all but one data set and is often more consistent, reaching the best result in every trial. The exception is DSJC500.0.5 where solution quality is marginally inferior. When longer computation times are considered ANTCOL(LS) is able to better the results reported by Galinier and Hao on one data set. In general, although our solution times are better than those quoted by Glass and Prügel-Bennett [12] they are not competitive with those of Galinier and Hao. This is largely due to the computational cost of constructing a solution from scratch for each ant in each cycle. For this reason we have not included any results for larger graphs as they proved too time-consuming.

7. Further comments and future developments

Our current version of ANTCOL is a dramatic improvement over the original Costa and Hertz version, and it is clear that the underlying definition of the trail is instrumental in guiding the search towards better solutions. Further minor improvements may be possible by more careful tuning of the parameters and other features. For example, it would make sense to include some measure of convergence to determine when a restart should be made rather than using a fixed number of cycles.

However, two factors suggest that the way forward may be to take a slightly different approach. Firstly, given the computational burden of the construction phase, if the approach is to be competitive on larger graphs it will be necessary to modify the process so that it is not necessary to reconstruct whole solutions every time. Secondly, the influence of the visibility function, and the relatively poor results on the unstructured DSJC500.5 graph suggests that the trail acts more as a background operator, and that it is relatively ineffective without an underlying strategy for building good solutions. This suggests that an approach that does not rebuild new solutions vertex by vertex in the construction phase may be worth considering. We are currently considering two possible approaches. The first is to use the trail to influence the choice of candidates for search algorithms that make use of large neighbourhoods, for example, to replace the random choices in the perturbation phase of the iterated local search algorithm of Paquete and Stützle [18], or to select an appropriate subset for the large neighbourhood suggested by Glass and Prügel-Bennett [13]. The second is influenced by the success of Galinier and Hao, and involves using previously generated colour classes to build solutions. The choice of which colour class to add next could be based on the existing trail definition together with a suitable visibility definition. As with Galinier and Hao's GA a local search based repair would be needed to ensure that all vertices were coloured at the end of the process and to act as a source of diversity. We intend to research both these possibilities.

8. Conclusions

This paper has introduced a series of enhancements to an ant colony optimisation approach to the graph colouring. Although the performance of the original implementation falls far short of the benchmarks set by recent meta-heuristic approaches to the problem our final implementation is competitive with the best-known methods over a wide range of

moderately sized problem instances. However, it is apparent that the method relies heavily on a sufficiently powerful solution builder. In our case this was achieved by a combination of the visibility function, *multi-set* options in the construction phase, and the tabu search improvement phase. However, the best way of balancing these features differed from problem to problem. Thus, in spite of its good performance the current implementation is not particularly robust. We also noted that repeated application of the construction phase means that the algorithm is very slow on larger problems.

This suggests that further work is needed in order to produce an ant colony optimisation based graph colouring algorithm that will work well across the entire spectrum of problems. However, we believe that the peak performance results of ANTCOL(LS) have demonstrated the potential of the method, and further work looking at issues such as automatic parameter selection and following some of the suggestions made in the previous section is certainly worth pursuing.

References

[1] C. Avanthay, A. Hertz, N. Zufferey, A variable neighborhood search for graph coloring, European J. Oper. Res. 151 (2003) 379–388.
[2] D. Brélaz, New methods to color vertices of a graph, Comm. ACM 22 (1979) 251–256.
[3] M. Chams, A. Hertz, D. de Werra, Some experiments with simulated annealing for coloring graphs, European J. Oper. Res. 32 (1987) 260–266.
[4] D. Costa, A. Hertz, Ants can colour graphs, J. Oper. Res. Soc. 48 (1997) 295–305.
[5] D. Costa, A. Hertz, O. Dubuis, Embedding of a sequential procedure within an evolutionary algorithm, J. Heuristics 1 (1995) 105–128.
[6] J.C. Culberson, Iterated greedy graph coloring and the difficulty landscape, in: D.S. Johnson, M.A. Trick (Eds.), Proceedings of the Second DIMACS Implementation Challenge, DIMACS Series, Discrete Mathematics and Theoretical Computer Science, vol. 26, American Mathematical Society, Providence, RI, 1996, pp. 245–284.
[7] M. Dorigo, V. Maniezzo, A. Colorni, Positive feedback as a search strategy, Technical Report 91-016, Dipartimento di Elettronica e Informazione, Policecnico di Milano, Italy, 1991.
[8] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of co-operating agents, IEEE Trans. Systems Man Cybernet. 26 (1996) 29–41.
[9] K.A. Dowsland, J.M. Thompson, Ant colony optimisation for the examination scheduling problem, J. Oper. Res. Soc. 56 (2005) 426–438.
[10] C. Fleurent, J.A. Ferland, Genetic and hybrid algorithms for graph coloring, Ann. Oper. Res. 63 (1996) 437–464.
[11] P. Galinier, J. Hao, Hybrid evolutionary algorithms for graph coloring, J. Combin. Optim. 3 (1999) 379–397.
[12] C.A. Glass, A. Prügel-Bennett, Genetic algorithm for graph coloring: exploration of Galanier and Hao's algorithm, J. Combin. Optim. 7 (2003) 229–236.
[13] C.A. Glass, A. Prügel-Bennett, A polynomially searchable exponential neighbourhood for graph colouring, J. Oper. Res. Soc. 56 (2005) 324–330.
[14] A. Hertz, D. de Werra, Using tabu search techniques for graph coloring, Computing 39 (1988) 345–351.
[15] D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon, Optimization by simulated annealing: an experimental evaluation, part II, graph coloring and number partitioning, Oper. Res. 39 (1991) 378–406.
[16] M. Laguna, R. Martí, A GRASP for coloring sparse graphs, Comput. Optim. Appl. 19 (2001) 165–178.
[17] F.T. Leighton, A graph coloring algorithm for large scheduling problems, J. Res. Nat. Bur. Standards 84 (1979) 489–506.
[18] L. Paquete, T. Stützle, An experimental investigation of iterated local search for coloring graphs, in: S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, G.R. Raidl (Eds.), Applications of Evolutionary Computing, Lecture Notes in Computer Science, vol. 2270, Springer, Berlin, 2002, pp. 122–131.
[19] M.G.C. Resende, C.C. Ribeiro, Greedy randomized adaptive search procedures, in: F. Glover, G. Kochenberger (Eds.), State-of-the-Art Handbook in Metaheuristics, Kluwer, Dordrecht, pp. 219–249.