Journal of the Operational Research Society (1997) 48, 295–305

Ants can colour graphs

D Costa and A Hertz
Ecole Polytechnique Fédérale de Lausanne, France

In the last few years researchers have shown how insect colonies can be seen as a natural model of collective problem solving. The analogy between the behaviour of ants looking for food and the well known travelling salesman problem has recently given rise to promising solution methods. We present in this paper an evolutionary search procedure for tackling assignment type problems. The algorithm repeatedly constructs feasible solutions of the problem under study by taking account of two complementary notions, namely the trace factor and the desirability factor. The use of such search principles will be illustrated for graph colouring problems. The results obtained have proven satisfactory when compared with those existing in the literature.

Keywords: ant algorithm; assignment type problem; collective performance; evolutionary method; graph colouring

Introduction

Introduced by Colorni, Dorigo and Maniezzo[1,2] the ant algorithm was first proposed for tackling the well known Travelling Salesman Problem (TSP). Two very similar algorithms were then proposed by the same authors for tackling the Quadratic Assignment Problem[3] and the Job Shop Problem[4]. In this paper we formalize these concepts in a more general framework and present an evolutionary algorithm which imitates the ant behaviour for the solution of a large class of combinatorial optimization problems.

The problems we are interested in are those which can be formulated as *Assignment Type Problems* (ATPs). Given n items and m resources, the problem is to determine an assignment of the items to the resources optimizing a given objective function and satisfying a set of additional side constraints. The associated mathematical model is the following[5].

$$\begin{aligned} \text{Min} \quad & f(\mathbf{x}) & & \\ \text{s.t.} \quad & \sum_{j\in J_i} x_{ij} = 1 & 1 \leqslant i \leqslant n \quad & (1) \\ & G_k(\mathbf{x}) \leqslant 0 & 1 \leqslant k \leqslant K \quad & (2) \\ & x_{ij} = 0 \text{ or } 1 & 1 \leqslant i \leqslant n, j \in J_i \quad & (3) \end{aligned}$$

where K is the number of additional side constraints

$$\mathbf{x} = (x_{11}, x_{12}, \ldots, x_{1m}, x_{21}, \ldots, x_{nm})$$

with

$$x_{ij} = \begin{cases} 1 & \text{if item } i \text{ is assigned to resource } j \\ 0 & \text{otherwise} \end{cases}$$

$$J_i \subseteq \{1, \ldots, m\} \text{ is the set of admissible resources for item } i \ (1 \leqslant i \leqslant n)$$

Restriction to function minimization is without loss of generality since $\max f(\mathbf{x}) = -\min(-f(\mathbf{x}))$. Constraints (1) and (3) force each item i to be assigned to exactly one resource j. A solution $\mathbf{x}$ which satisfies constraints (1)–(3) is said to be *feasible*. In the remainder, X will denote the set of all feasible solutions. The objective function $f(\mathbf{x})$ as well as constraints of type (2) are specific to the problem under study. Several well known problems are instances of ATPs. For example, let us cite the resource assignment problem, the quadratic assignment problem, the travelling salesman problem, the graph colouring problem and the set covering problem. The difficulty of these problems is related to the nature of the objective function and the additional constraints. With the exception of the resource assignment problem, all of the ATPs mentioned above were shown to be NP-Hard[6].

The graph colouring problem can be formulated in the following way. A *q-colouring* of a graph $G = (V, E)$ with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set E is a mapping $c\colon V \to \{1, 2, \ldots, q\}$ such that $c(v_i) \neq c(v_j)$ whenever E contains an edge $[v_i, v_j]$ linking the vertices v_i and v_j. The minimal number of colours q for which a q-colouring exists is called the *chromatic number* of G and is denoted $\chi(G)$. An optimal colouring is one which uses exactly $\chi(G)$ colours. A mathematical formulation in terms of ATP is given below. The items are the vertices of V and the resources are the colours. Since it is always possible to colour any graph $G = (V, E)$ in $n = |V|$ colours, we set $m = n$.

$$x_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ receives colour } j \\ 0 & \text{otherwise} \end{cases}$$

$$J_i = \{1, \ldots, n\} \qquad 1 \leqslant i \leqslant n$$

$$f(\mathbf{x}) = \sum_{k=1}^{n} k \cdot \delta\left(\sum_{l=1}^{n} x_{lk}\right) \quad \text{where } \delta(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$G_j(\mathbf{x}) = \sum_{[v_i, v_k]\in E} x_{ij} \cdot x_{kj} \leqslant 0 \qquad 1 \leqslant j \leqslant n$$

Correspondence: Dr D Costa, Département de Mathématiques, Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, France.

The objective function $f(\mathbf{x})$ adds up the numbers associated with the colours used in the colouring $\mathbf{x}$. In this way an optimal colouring uses necessarily all consecutive colours between 1 and $\chi(G)$. Constraints $G_j(\mathbf{x})$ avoid edges with both endpoints having the same colour.

The remainder of this paper is organized as follows. In the next section we introduce a new formalism for the ant algorithm which is well suited for tackling ATPs. Adaptations of this algorithm to the graph colouring problem are then presented and computational experiments are reported. The results achieved with the ant algorithm are compared with those existing in the literature. We conclude with general remarks on this work and directions for future research.

An ant algorithm for assignment type problems

When designing a heuristic for a combinatorial optimization problem, or more specifically for an ATP, one can either build a method which is specific to the problem under study or adapt some general existing schemes. In the latter case constructive, sequential and evolutionary techniques are search strategies which have successfully been used for the solution of difficult problems[7].

Constructive methods build feasible solutions by starting from an 'empty' solution and by inserting repeatedly a new component into the current partial solution. Sequential methods start from an initial feasible solution and move step by step from one solution to a neighbour solution by performing some elementary modifications. One of the most promising innovations in the field of general heuristics is certainly the development of evolutionary methods. These methods use the collective properties of a group of distinguishable solutions, called *population*, for searching intelligently in the solution space. The central idea is to manipulate a set of solutions located in different promising areas of the solution space. The population goes through an evolution process which is made up of two distinct phases. In the *self-adaptation phase*, solutions change their internal structure without any interaction with the other members of the population. In the *cooperation phase*, solutions of the current population exchange information with the aim of producing new solutions which inherit good attributes. Hopefully such an adaptive behaviour will make good solutions of the problem under study to emerge.[8] Examples of evolutionary methods are Genetic Algorithms[9,10], Scatter Search[11] and Ant Systems[1,2].

The collective performance of social insects, such as ants, bees, wasps or termites have intrigued entomologists for several years. Their main concern is about the mechanisms which allow the individuals of a same colony to coordinate their activities and to favor the survival of the species. Apparently everything works out because of an invisible factor which regulates the activities of each individual. Studies have shown that this global adaptive behaviour arises from a multitude of very simple local interactions. The nature of these interactions, the treatment of the information as well as the differences between the solitary behaviour and the social behaviour have remained mysterious for a long time. The realization of a specific task by a colony of insects has shown that the coordination of the work does not depend on the insects but rather on the advancing state of the task. Coordination emerges from an autocatalytic chain retroaction between stimuli and responses. An insect does not control directly its work; the whole process evolves as if each insect were guided by its work. While working an insect modifies the form of the stimulation which triggers its behaviour. This induces the emergence of a new stimulation which will trigger new reactions in the colony.

In order to illustrate the emergence of collective structures in an insect society, let us cite the example of an ant colony in search of a nearby feeding source. Initially, ants leave the nest and move randomly. When an ant discovers a feeding source, it informs its congeners by laying a temporary trail on the ground on its way back to the nest. The trail is nothing else than a chemical substance, called *pheromone*, which guides the other ants towards the same feeding source. On their way back, the latter also lay pheromone on the ground and thus reinforce the marking of the path which leads from the nest to the discovered feeding source. The reinforcement of the marking by pheromone optimizes the collection of food. All trails laid on the ground evaporate progressively as time goes by. Because of the larger elapsed time between two passages of an ant on the paths leading to remote feeding sources, trails will get undetectable faster on these paths. In the long term, ants will thus all prefer the closest feeding source. This example shows 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. In this case, as pointed out by Théraulaz *et al*[12], 'the environment plays the role of a spatio-temporal memory keeping track of the swarm past actions while selecting its own dynamic regime'.

For many years, engineers have been interested in the behaviour of social insects in order to define new models of collective problem solving. The Ant System, developed recently by Colorni *et al*[1,2], is an evolutionary algorithm which takes its roots in the collective behaviour of an ant colony. In the cooperation phase of an ant algorithm each solution of the population is examined with the aim of updating a global memory keeping track of important structures of X which have been successfully exploited in the past. The self-adaptation phase uses a problem-specific constructive method to create a new population of solutions on the basis of the global memory.

This search principle can easily be adapted to tackle general ATPs. The concepts introduced below generalize the ideas presented by Colorni *et al* in the TSP framework.

Let us consider n items i each of them having to be assigned to exactly one resource $j \in J_i \subseteq \{1, \ldots, m\}$. Such an assignment has to meet a set of constraints and its cost f needs to be as low as possible. Two decisions have to be taken at each step of a constructive method. The first is about the choice of the next item to assign while the second defines the resource to which the chosen item is assigned. In an ordinary constructive method, these decisions are made in a myopic way by completing the current partial solution at best. In other words, they deal at each step with an item and a resource the *desirability* of which is maximum. The ant algorithm uses a notion which is complementary to the desirability of an item and a resource. Feasible solutions are built probabilistically on the basis of the knowledge drawn from the previous solution constructions. The notion of *trail* will be used in the remainder to designate an element of information which is known because of previous experiments. At each stage k of the construction process $(1 \leqslant k \leqslant n)$, an ant chooses an unassigned item i with probability $p_{\mathrm{it}}(k, i)$ and a resource j, feasible for item i, with probability $p_{re}(k, i, j)$. Given a partial solution $s[k-1]$ in which items $o(1), \ldots, o(k-1)$ have already been assigned to resources and given a not yet assigned item i and a feasible resources $j \in J_i$ for i, probabilities $p_{it}(k, i)$ and $p_{re}(k, i, j)$ are defined as follows:

$$p_{it}(k, i) = \frac{\tau_1(s[k-1], i)^{\alpha_1} \cdot \eta_1(s[k-1], i)^{\beta_1}}{\sum\limits_{o \notin \{o(1), \ldots, o(k-1)\}} \tau_1(s[k-1], o)^{\alpha_1} \cdot \eta_1(s[k-1], o)^{\beta_1}}$$

$$p_{re}(k, i, j) = \frac{\tau_2(s[k-1], i, j)^{\alpha_2} \cdot \eta_2(s[k-1], i, j)^{\beta_2}}{\sum\limits_{r \in J_i} \tau_2(s[k-1], i, r)^{\alpha_2} \cdot \eta_2(s[k-1], i, r)^{\beta_2}}$$

where $\tau_1(s[k-1], i)$ and $\tau_2(s[k-1], i, j)$ measure respectively the trail of item i and of resource j for i; $\eta_1(s[k-1], i)$ and $\eta_2(s[k-1], i, j)$ measure respectively the desirability of item i and of resource j for i; α_l and $\beta_{\mathrm{L}}(l = 1, 2)$ are positive coefficients which control the relative importance of the trail and the desirability factors.

Table 1 presents a generic ant algorithm for tackling ATPs. The size of the colony is equal to *nants*. Each ant in the colony builds a feasible solution at each cycle. The best feasible solution reached so far is denoted by s°. Initially all trails are set equal to 1 since the ants do not have any knowledge of the problem to be solved. At the end of each cycle, trails are updated on the basis of the characteristics of the feasible solutions produced by the ants.

Usually only one of the two decisions are made probabilistically at each step of the constructive method. In some applications, the choice of the next item is either natural or based on a deterministic rule which is not influenced by the notion of trail. In the next section, we shall present eight sets of probabilities $\{p_{it}, p_{re}\}$ which define eight variants of an ant algorithm for tackling graph colouring problems.

Although the field of evolutionary methods has kept growing and evolving since the first developments in the 1970s there exist few evolutionary algorithms for solving the graph colouring problem. The only adaptations we are aware of are due to Davis[13], Fleurent and Ferland[14] and Costa, Hertz and Dubuis[15]. None of them are governed by mechanisms derived from the collective performance of social insects.

ANTCOL: an ant algorithm for graph colouring

Consider the graph colouring problem presented in the first section. Given a graph $G = (V, E)$ a *stable set* is a subset of vertices whose elements are pairwise nonadjacent. A feasible solution $s \in X$ of the colouring problem is any partition $s = (V_1, \ldots, V_q)$ of the vertex set V into q stable sets (q not fixed). The objective is to find a q-colouring of G with q as small as possible.

The role of each ant is to colour the graph in a constructive way. The choice of the constructive method

Table 1 The ant algorithm for a general ATP

```
$f^{\circ} := \infty$;            (best value reached so far)
$ncycles := 0$;         (cycle counter)
$best\_cycle := 0$;     (cycle at which the best solution has been found)
$\tau_1(s[k-1], i) := 1; \tau_2(s[k-1], i, j) := 1; \forall i = 1, \ldots, n; \forall j \in J_i; \forall s[k-1], k = 1, \ldots, n$;
While stopping criterion = false do
  $ncycles :=$ ncycles + 1;
  For $a := 1$ to $nants$ do
    For $k := 1$ to $n$ do
      choose an item $i \in \{1, \ldots, n\} \backslash \{o(1), \ldots, o(k-1)\}$ with probability $p_{it}(k, i)$;
      choose a feasible resource $j$ for item $i$ with probability $p_{re}(k, i, j)$;
      assign resource $j$ to item $i$; $x_{ij} := 1; x_{il} := 0; \forall 1 \neq j; o(k) := i$;
    compute the cost $f(s_a)$ of solution $s_a = (x_{11}, \ldots, x_{nm})$;
  $P := \{s_1 . s_2, \ldots, s_{nants}\}$;
  $\tilde{s} = \arg\min\{f(s) | s \in P\}$;
  If $f(\tilde{s}) < f^{\circ}$ then $s^{\circ} := \tilde{s}; f^{\circ} := f(s^{\circ})$; $best\_cycle := ncycles$;
  update trails $\tau_1$ and $\tau_2$;
```

is not restricted and will be discussed later. The experience accumulated during the previous constructions is memorized in a $n \times n$ matrix M which is updated periodically. Given two non adjacent vertices v_r and v_s, the value M_{rs} in M is proportional to the quality of the colourings obtained by giving the same colour to v_r and v_s. Hence, M_{rs} corresponds to a trail existing between vertices v_r and v_s. Initially, all values in M are set equal to 1. The trail stored in M evaporates progressively as time goes, the coefficient of evaporation being denoted by $(1-\rho)$. Let $s_1, \ldots, s_{\text{nants}}$ be the colourings obtained at the end of a cycle and let $S_{rs} \subseteq \{s_1, \ldots, s_{\text{nants}}\}$ denote the subset of solutions in which v_r and v_s have the same colour. Finally, let q_a be the number of colours used in s_a $(1 \leqslant a \leqslant \text{nants})$. The values M_{rs} in M are updated as follows

$$M_{rs} := \rho \cdot M_{rs} + \sum_{s_a \in S_{rs}} \frac{1}{q_a}$$

Such an evolutionary algorithm has been called ANTCOL. A sketch of ANTCOL is presented in Table 2.

Many constructive methods for graph colouring have been proposed in the literature. Most of them colour each vertex in turn with the smallest possible colour. The quality of the resulting colouring strongly depends on the order in which vertices are scanned. It is important to point out that there always exists an order of the vertices which yields an optimal colouring. This can be seen by taking any optimal colourings $s = (V_1, \ldots, V_{\chi(G)})$ and by ordering the vertices in such a way that vertices in V_k come before those in V_1 whenever $k < 1$.

The ordering of vertices can be either *static* or *dynamic*. An ordering of the vertices is called static if it can be completely determined before any colour has been assigned. An ordering is called dynamic if the choice of the next vertex to be coloured is based on the previous colour assignments. Table 3 sketches the fundamentals of five among the most common constructive methods for graph colouring. Ties are broken randomly everytime a given criterion is not enough to order two vertices. By definition, the *degree of saturation* dsat(v) of a non coloured vertex is the number of different colours already assigned to vertices adjacent to v. Given a subset $W \subseteq V$ of vertices, the neighbourhood $N_W(v)$ of vertex v includes all vertices $w \in W$ that are adjacent to v. The *degree* $\deg_W(v)$ is the number of vertices in $N_W(v)$. The set of all vertices which have not been coloured yet is denoted by A. Other constructive methods for graph colourings have been described by Dunstan[16], Johri and Matula[17] and Morgenstern[18].

It appears in Table 6 that DSATUR and RLF perform significantly better than RANDOM, LF and SL. The constructive methods used in ANTCOL are therefore all derived from DSATUR and RLF. Eight variants of ANTCOL are proposed. Procedures ANT_RLF and ANT_DSATUR described in Tables 4 and 5 are the basis of these eight variants.

Given a partial solution $s[k-1] = (V_1, \ldots, V_q)$, an uncoloured vertex v and a feasible colour c for v, the trail factor $\tau_2(s[k-1], v, c)$ is computed as follows

$$\tau_2(s[k-1], v, c) := \begin{cases} 1 & \text{if } V_c \text{ is empty} \\ \dfrac{\sum_{x \in V_c} M_{xv}}{|V_c|} & \text{otherwise} \end{cases}$$

Such a trail corresponds to the mean quality of the solutions obtained in the past with v having the same colour as the vertices in V_c.

In procedure ANT_RLF(Σ, Ω) the two strategies in Σ are about the choice of the first vertex of a stable set while the three strategies in Ω define the desirability of a vertex which can still be inserted in the stable set under construction. In procedure ANT_DSATUR(Ψ), the desirability $\eta_1(s[k-1], v)$ of a uncoloured vertex corresponds to its degree of saturation. Two different strategies Ψ are considered for selecting the next vertex to be coloured as well as the colour to assign to it. In the second strategy $(\Psi = 2)$ vertices v do not necessarily receive the smallest feasible colour $c_{\min}(v)$. Colour c is selected by taking account of the trails $\tau_2(s[k-1], v, c)$ $(c_{\min}(v) \leqslant c \leqslant q)$ whenever colour $c_{\min}(v)$ is smaller than the number of colours q already used.

Table 2 The algorithm ANTCOL

```
$M_{rs} := 1\ \forall [v_r, v_s] \notin E$;        (initialization of the trail between pairs of non adjacent vertices)
$f^\circ := \infty$;                               (number of colours in the best colouring reached so far)
For $ncycles := 1$ to $ncycles_{\max}$ do
    $\Delta M_{rs} := 0;\ \forall [v_r, v_s] \notin E$;
    For $a := 1$ to $nants$ do
        colour the graph by means of a constructive method;
        let $s = (V_1, V_2, \ldots, V_q)$ be the colouring obtained
        If $q < f^\circ$ then $s^\circ := (V_1, \ldots, V_q); f^\circ := q$;
        $\Delta M_{rs} := \Delta M_{rs} + \frac{1}{q}\ \forall [v_r, v_s] \notin E, \{v_r, v_s\} \subseteq V_j, j = 1, \ldots, q$;
    $M_{rs} := \rho \cdot M_{rs} + \Delta M_{rs}\ \forall [v_r, v_s] \notin E$;
```

Table 3 Constructive methods for graph colouring

Static orders:
RANDOM Vertices are ordered randomly.
LF (Largest First) Vertices are ordered so that their degrees $\deg_V(v)$ form a non increasing sequence.
SL (Smallest Last) The order $v_1, v_2, \ldots, v_n$ is such that each vertex v_i has smallest degree in the subgraph induced by vertices $v_1, \ldots, v_i$.
Dynamic orders:
DSATUR[19] Vertices are ordered by choosing at each step the vertex $v \in A$ with a maximum degree of saturation dsat(v). If v is not unique preference is given to the vertex v for which $\deg_A(v)$ is minimum.
RLF (Recursive Largest First)[20] Classes of colours are built sequentially. Once a vertex $v \in A$ with maximum degree $\deg_A(v)$ has been selected, the current stable set is augmented by inserting, as long as possible, the vertex $v \in AB$ with maximum degree $\deg_B(v)$. The set B contains every uncoloured vertex which can no longer be included in the stable set under construction.

Computational experiments

Random graphs

Randomly generated graphs are used to test the efficiency and examine the average behaviour of ANTCOL. A random graph $G_{n,p}$ is a graph with n vertices such that there exists an edge with a probability p between any pair of vertices, independently of the existence or non-existence of any other edge. This family of graphs has been deeply studied with respect to colouring, especially for $p = 0.5$. There exists some asymptotic results but they are of little practical use in evaluating algorithm performance[21]. A probabilistic estimate $\chi(G_{n,p})$ of the chromatic number can be obtained easily as shown by Johri and Matula[17].

All solution methods described in the previous sections were implemented in Pascal and run on a Silicon Graphics Workstation (9 Mflops). Computing times are reported in CPU seconds of this machine. For each value of $p \in \{0.4, 0.5, 0.6\}$ a sample of 20 $G_{100,p}$ graphs, 10 $G_{300,p}$ graphs, 5 $G_{500,p}$ graphs and 2 $G_{1000,p}$ graphs is taken into account in our experiments. Each class of graphs is generated using a combination of Multiplicative Linear Congruential Generators that is described in detail by L'Ecuyer[22]. For $p = 0.5$ we consider the same random graphs as Fleurent and Ferland[14] and Costa, Hertz and Dubuis[15]. Note that the sample contains one $G_{500,0.5}$ and one $G_{1000,0.5}$ graph which have been used by Johnson *et al*[23] for testing various algorithms. The results reported in the subsequent Tables are average results obtained after one run of the algorithm considered on each graph in a class $G_{n,p}$.

Constructive methods for graph colouring

This subsection is devoted to the analysis of the constructive methods presented in the previous section. Table 6 reports the quality of the colourings achieved with RANDOM, LF, SL, DSATUR and RLF. The running times are given in seconds and put in brackets.

We observed that the difference between the largest and the smallest number of colours used in each class $G_{n,p}$ is

Table 4 Procedure ANT_RLF (Σ, Ω) with $\Sigma \in \{1, 2\}$ and $\Omega \in \{1, 2, 3\}$

```
$q := 0$;        (number of colours used)
$W := V$;        (uncoloured vertices which can be included in the stable set under construction)
$k := 0$;        (number of coloured vertices)
While $k < |V|$ do
    $k := k + 1$; $q := q + 1$;
    $B := \emptyset$;   (uncoloured vertices which can no longer belong to the stable set $V_q$)
    select a first vertex $v$ according to one of the two strategies $\Sigma$:
    $\Sigma: \begin{cases} (1)\ v = \arg\max\{\deg_W(v') | v' \in W\}; \\ (2)\ v\text{:chosen randomly in } W; \end{cases}$

    $V_q := \{v\}$;
    While $W \backslash (N_W(v) \cup \{v\}) \neq \emptyset$ do
        $k := k + 1$;
        $B := B \cup N_W(v)$; $W := W \backslash (N_W(v) \cup \{v\})$;
        Let $s[k-1] = (V_1, \ldots, V_q)$ be the current partial solution. Choose $v \in W$ with probability $p_{it}(k, v)$ where
    $\tau_1(s[k-1], v) = \tau_2(s[k-1], v, q)$ and $\eta_1(s[k-1], v)$ is defined according to one of the three strategies $\Omega$:
        $\Omega: \begin{cases} (1)\ \eta_1(s[k-1], v) = \deg_B(v); \\ (2)\ \eta_1(s[k-1], v) = |W| - \deg_W(v); \\ (3)\ \eta_1(s[k-1], v) = \deg_{W \cup B}(v); \end{cases}$
        $V_q := V_q \cup \{v\}$;
    $W := B \cup N_W(v)$;
```

Table 5 Procedure ANT_DSATUR(Ψ) with $\Psi \in \{1, 2\}$

```
$c_{\min}(v_i) := 1$; $\mathrm{dsat}(v_i) := 0 \; \forall v_i \in V$;
$V_j := \varnothing \; \forall j = 1, \ldots, n$;
$A := V$;   (uncoloured vertices)
$v := \arg\max\{\deg_A(v') | v' \in A\}$
$V_1 := V_1 \cup \{v\}$;
$q := 1$;   (number of colours used)
For $i := 2$ to $n$ do
  update $c_{\min}(v')$ and $\mathrm{dsat}(v')$; $\forall v' \in N_A(v)$;
  $A := A \backslash \{v\}$;
  Let $s[k-1] = (V_1, \ldots, V_q)$ be the current partial solution. Augment $s[k-1]$ following one of the two strategies $\Psi$:
  $\Psi$: (1) choose $v \in A$ with probability $p_{it}(k, v)$ where $\eta_1(s[k-1]\; v) = \mathrm{dsat}(v)$ and $\tau_1(s[k-1], v) = \tau_2(s[k-1], v, c_{\min}(v))$;
              $c := c_{\min}(v)$;
          (2) choose $v \in A$ with probability $p_{it}(k, v)$ where $\eta_1(s[k-1], v) = \mathrm{dsat}(v)$ and $\tau_1(s[k-1], v) = 1$; choose a feasible colour
              $c \in \{c_{\min}(v), \ldots, \max(c_{\min}(v), q)\}$ with probability $p_{re}(k, v, c)$ where $\eta_2(s[k-1], v, c) = 1$;
  $V_c := V_c \cup \{v\}$;
  If $c = q + 1$ then $q := q + 1$;
```

extremely low. This great stability allows us to compare easily the performance of the five constructive methods considered. The ranking of the latter in quality increasing order is as follows: RANDOM, SL, LF, DSATUR and RLF. When compared with the probabilistic estimate $\tilde{\chi}(G)$ of Johri and Matula[17] it clearly appears that the quality of the colourings obtained decreases progressively a n and p increase.

The complexity of the above methods is $O(n^2)$ except for RLF which has complexity $O(n^3)$. This difference is obvious in Table 6 when considering the running times of RLF which are much higher than those of the other procedures on large graphs. It is worth pointing out that DSATUR requires twice as much time as RANDOM to colour a graph on 100 vertices even though both algorithms have the same complexity bound. This increase is compensated by a reduction of approximately 9% in the number of colours used. With the exception of SL, it can be observed that the quality of the colourings increases with the running time of the constructive method considered. Generally speaking the choice of the method should be made according to the time at disposal for building a solution.

Constructive methods for graph colouring are often implemented to produce quickly an upper bound of the chromatic number. In particular such a bound is useful for initializing another heuristic procedure whose function is to find a q-colouring of G with q fixed[24,25].

A constructive method for graph colouring can be used recursively in the following fashion. Every time a colouring $s = (V_1, V_2, \ldots, V_q)$ is produced, the stable set of maximum size V_{j^*} is removed and the method is run again on the subgraph induced by vertices $V_1 \cup \cdots \cup V_{j^*-1} \cup V_{j^*+1} \cup \cdots \cup V_q$. Johri and Matula[17] showed that the use of such a recursive method on graphs $G_{1000,0.5}$ improves by 4–8% the quality of the colourings while requiring a running time which is only two to four times greater. ANTCOL provides a new way of improving the performance of a given constructive method.

Adjustment of the parameters

Various tests have been performed in order to select appropriate values of the parameters governing the search in the procedures ANT_RLF and ANT_DSATUR. The first experiments were carried out with the 20 graphs in the class $G_{100,0.5}$.

Let us first examine the coefficients α_l and β_l which are associated with the trail and the desirability factors. For this

Table 6 Comparison of the main constructive methods for graph colouring

n	*p*	$\tilde{\chi}(G)$	*RANDOM*	*LE*	*SL*	*DSATUR*	*RLF*
	0.4	14	17.60 (<0.01)	16.10 (<0.01)	16.65 (0.01)	14.70 (0.01)	14.15 (0.01)
100	0.5	16	21.35 (<0.01)	20.15 (<0.01)	20.50 (0.01)	18.45 (0.01)	17.45 (0.01)
	0.6	19	26.20 (0.01)	24.50 (0.01)	24.85 (0.01)	22.60 (0.01)	21.50 (0.01)
	0.4	28	39.00 (0.02)	37.10 (0.03)	38.10 (0.05)	34.30 (0.04)	31.90 (0.23)
300	0.5	35	48.20 (0.03)	46.00 (0.04)	46.80 (0.05)	43.30 (0.05)	39.90 (0.24)
	0.6	42	61.00 (0.03)	58.20 (0.04)	58.90 (0.06)	53.70 (0.05)	50.50 (0.26)
	0.4	40	57.20 (0.07)	56.00 (0.09)	56.80 (0.16)	51.00 (0.11)	47.60 (0.94)
500	0.5	50	72.40 (0.08)	69.20 (0.10)	71.80 (0.17)	65.00 (0.13)	60.60 (1.03)
	0.6	62	91.40 (0.08)	87.60 (0.12)	89.60 (0.19)	82.80 (0.15)	76.00 (1.11)
	0.4	68	99.50 (0.27)	96.50 (0.37)	97.00 (0.68)	92.00 (0.48)	84.00 (6.73)
1000	0.5	85	126.50 (0.30)	122.00 (0.41)	124.50 (0.72)	116.00 (0.58)	107.50 (7.22)
	0.6	106	160.50 (0.34)	155.00 (0.47)	157.00 (0.81)	146.50 (0.62)	138.50 (8.10)

purpose we ran the eight variants of ANTCOL during $\text{ncycles}_{\max} = 100$ cycles with a colony of size $\text{nants} = 100$. The coefficient of trail evaporation $(1 - \rho)$ has been set experimentally equal to 0.5. Table 7 shows the results produced by ANTCOL with α_l and β_l ranging from 0–4. The number in brackets besides the name of the procedure reports the mean number of colours achieved after one run of the procedure by setting $\alpha_l = 0$ and $\beta_l = 1$ and by replacing all probabilistic choices of a vertex by a systematic choice of the vertex v that maximizes $p_{it}(k, v)$ (which is proportional to $\eta_1(s[k-1], v)$ in this case). Thus, the values [17.45] and [18.45] on bottom of Tables 7(a) and (g) correspond with the mean number of colours obtained after one run of RLF and DSATUR.

With the exception of procedure ANT_DSATUR(2), it appears that the trail factor plays an essential role on the quality of the results achieved. A constant trail ($\alpha_l = 0$) is clearly inappropriate. Procedure ANT_RLF yields the best colourings with values $\alpha_1 = 2$ and $2 \leqslant \beta_1 \leqslant 4$. Strategy $\Sigma = 1$ is preferable to strategy $\Sigma = 2$ when α_1 is smaller than or equal to 1. For larger values of α_1, the reverse holds. Indeed a random selection of the first vertex to insert in the current stable set yields a beneficial diversification effect over the long term ($\alpha_1 \leqslant 2$). The three definitions of $\eta_1(s[k-1], v)$ $(\Omega \in \{1, 2, 3\})$ do not really influence the performance of ANT_RLF. Strategy $\Psi = 2$ is better than strategy $\Psi = 1$ only if procedure ANT_DSATUR does not take into account the trail factor ($\alpha_1 = \alpha_2 = 0$). Table 7(h) shows that the choice of a colour greater than $c_{\min}(v)$ is not relevant, even if it is based on the experience gained so far. The maximum efficiency of procedure ANT_DSATUR(1) is reached with value $\alpha_1 = 1$.

Table 7 shows that ANTCOL improves considerably on the results achieved with RLF and DSATUR if parameters α_1, α_2 and β_1 are adjusted adequately. Up to now, the best average obtained with ANTCOL is equal to 15.30. As a reminder, let us point out that the evolutionary descent method presented by Costa, Hertz and Dubuis[15] could colour the same graphs $G_{100,0.5}$ with an average of 14.95 colours. With a colony of size 100, each cycle of ANTCOL

Table 7 Influence of parameters α and β on the performance of ANTCOL

$\alpha_1\backslash\beta_1$	0	1	2	3	4
0	17.75	17.40	16.95	16.90	16.80
1	16.80	16.30	16.00	15.90	15.95
2	16.00	15.80	15.95	15.75	15.90
3	16.15	16.10	16.15	16.10	15.95
4	16.60	16.30	16.30	16.10	16.05

(a) ANT_RLF(1,1) [17.45]

$\alpha_1\backslash\beta_1$	0	1	2	3	4
0	18.35	17.95	17.60	17.15	17.05
1	17.70	17.05	16.80	16.60	16.35
2	15.60	15.60	15.55	15.45	15.30
3	15.85	15.90	15.85	15.70	15.70
4	16.05	15.85	15.95	15.70	15.70

(b) ANT_RLF(2,1) [17.65]

$\alpha_1\backslash\beta_1$	0	1	2	3	4
0	17.65	17.25	17.10	16.90	16.75
1	16.80	16.60	16.30	16.30	16.20
2	16.05	15.95	15.80	15.90	15.90
3	16.25	16.15	16.05	16.00	15.90
4	16.45	16.20	16.25	16.15	16.05

(c) ANT_RLF(1,2) [18.45]

$\alpha_1\backslash\beta_1$	0	1	2	3	4
0	18.30	17.95	17.65	17.55	17.40
1	17.70	17.30	17.05	16.90	16.85
2	15.55	15.70	15.50	15.70	15.55
3	15.95	15.65	15.75	15.75	15.80
4	15.95	15.85	15.85	15.85	15.80

(d) ANT_RLF(2,2) [19.20]

$\alpha_1\backslash\beta_1$	0	1	2	3	4
0	17.70	17.60	17.40	17.15	17.15
1	16.85	16.55	16.45	16.30	16.30
2	15.95	16.00	15.95	15.80	15.95
3	16.20	16.30	16.15	16.25	16.25
4	16.35	16.60	16.40	16.45	16.55

(e) ANT_RLF(1,3) [18.90]

$\alpha_1\backslash\beta_1$	0	1	2	3	4
0	18.30	18.00	17.90	17.80	17.80
1	17.80	17.35	17.30	17.05	16.95
2	15.70	15.55	15.60	15.55	15.60
3	15.75	15.80	15.80	15.85	15.80
4	16.00	15.75	16.00	15.95	15.80

(f) ANT_RLF(2,3) [19.40]

$\alpha_1\backslash\beta_1$	0	1	2	3	4
0	19.10	19.20	19.00	19.10	18.90
1	15.55	15.60	15.60	15.70	15.50
2	15.90	15.90	15.75	15.90	15.65
3	15.90	15.90	15.90	16.00	15.90
4	16.15	16.05	15.85	16.05	15.90

(g) ANT_DSATUR(1) [18.45]

$\alpha_2\backslash\beta_1$	0	1	2	3	4
0	18.20	18.05	18.05	18.10	18.05
1	18.20	18.05	17.85	17.95	17.85
2	18.40	18.20	18.05	17.95	17.75
3	18.15	18.20	18.10	18.00	18.05
4	18.05	18.10	17.95	18.00	17.90

(h) ANT_DSATUR(2) [18.50]

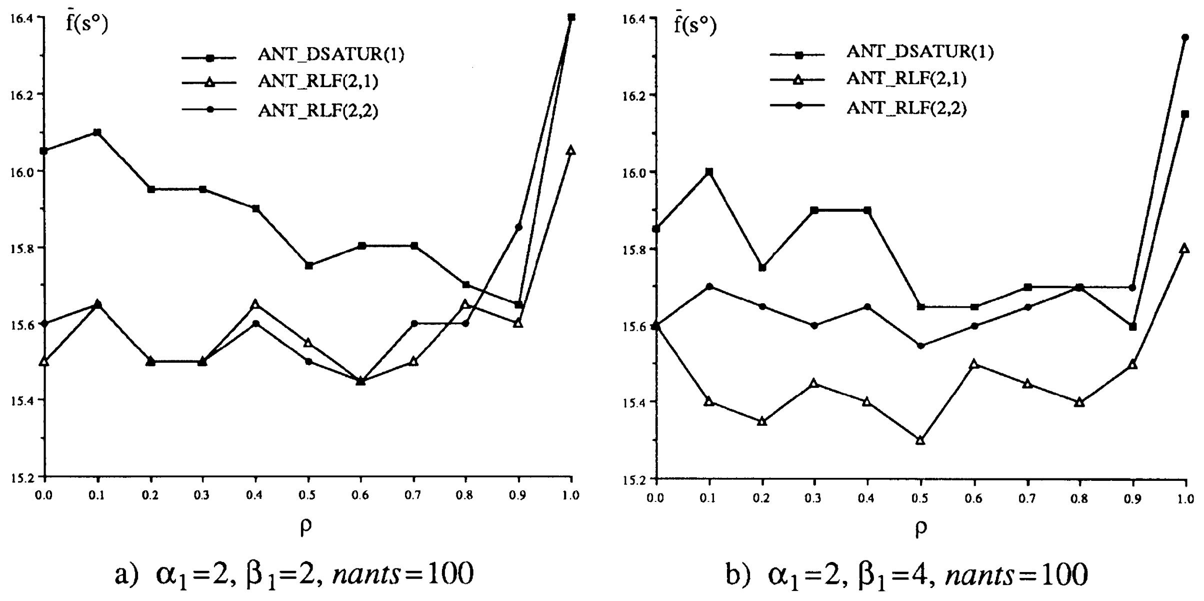


Figure 1 Influence of coefficient ρ on the performance of ANTCOL.

requires a CPU time which varies from 2 to 3.5 secs. depending on the procedure that is considered. The running time of ANTCOL will be discussed more thoroughly later.

The effects of the evaporation coefficients and the size of the colony are represented graphically in Figures 1 and 2. To this purpose two sets of parameters (α_1, β_1) were retained. ANTCOL is interrupted systematically after $\text{ncycles}_{\max} = 100$ cycles. With the exception of situation $\rho = 1$ which is clearly inappropriate (no evaporation), it follows from Figure 1 that coefficient ρ does not clearly influence the performance of ANTCOL. On the whole, value $\rho = 0.5$ yields satisfactory results.

The importance of the number of ants varies according to the retained constructive method. As a general rule, the performance of ANTCOL increases when the colony grows up to a certain size. This phenomenon is more important for procedure ANT_DSATUR. The latter, contrary to ANT_RLF, requires a minimum number of approximately

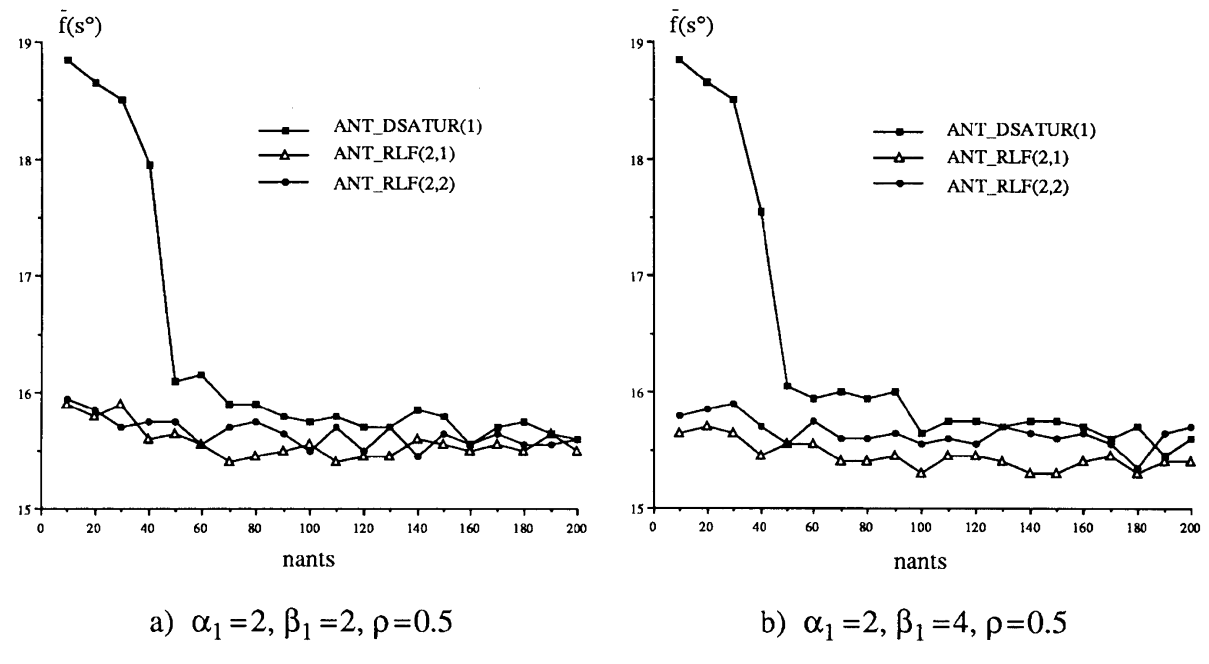


Figure 2 Influence of the size of the colony on the performance of ANTCOL.

Table 8 Evaluation of ANTCOL ($\alpha_1 \in \{1, 2\}$, $\beta_1 = 4$, $\rho = 0.5$, $ncycles_{\max} = 50$)

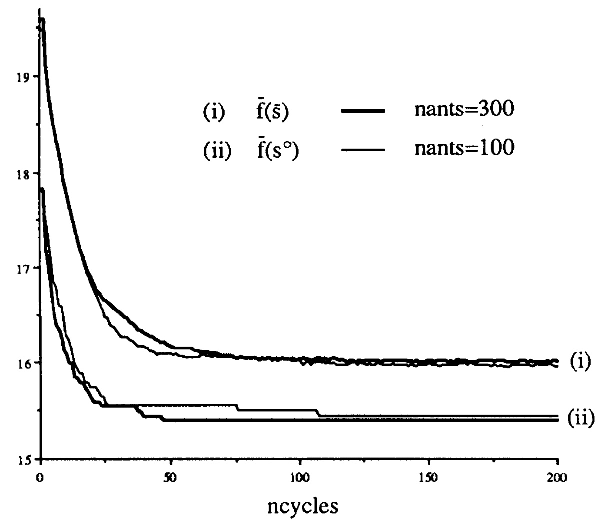
n	d	*ANT_RLF(2,1)*				*ANT_DSATUR(1)*			
		nants = 100		*nants = 300*		*nants = 100*		*nants = 300*	
	0.4	12.65	(111.1)	12.70	(329.7)	12.80	(187.0)	13.95	(609.8)
100	0.5	15.40	(88.0)	15.20	(259.0)	15.65	(178.1)	16.10	(563.6)
	0.6	18.65	(74.5)	18.65	(207.8)	18.85	(174.3)	18.75	(529.3)
	0.4	29.20	(1,187.6)	28.90	(3,471.1)	36.00	(1,730.8)	30.40	(5,245.4)
300	0.5	36.30	(952.6)	35.70	(2,813.4)	44.90	(1,600.6)	38.10	(4,924.4)
	0.6	44.40	(788.8)	43.70	(2,334.9)	55.00	(1,575.7)	47.30	(4,693.8)
	0.4	46.40	(3,930.6)	45.80	(11,693.7)	54.00	(5,145.9)	54.00	(15,428.6)
500	0.5	56.00	(3,274.4)	55.60	(9,872.8)	68.20	(4,977.1)	67.80	(14,894.4)
	0.6	68.80	(2,760.5)	68.20	(8,199.7)	84.20	(4,804.3)	83.80	(14,626.3)
	0.4	91.50	(22,943.2)	—		95.50	(24,945.4)	—	
1000	0.5	111.00	(19,914.2)	—		121.00	(22,952.1)	—	
	0.6	127.50	(16,994.9)	—		151.50	(21,753.1)	—	

100 ants to produce satisfactory results. Above a certain value *nants*$^+$, the size of the colony does not really influence the quality of the solutions obtained. With the parameters considered in Figure 2, *nants*$^+$ ranges from 100–150. Since our algorithm was implemented in a sequential environment, its running time increases linearly with the size of the colony. Hence it is important to use a number of ants close to *nants*$^+$ in order to get good results within a reasonable amount of CPU time.

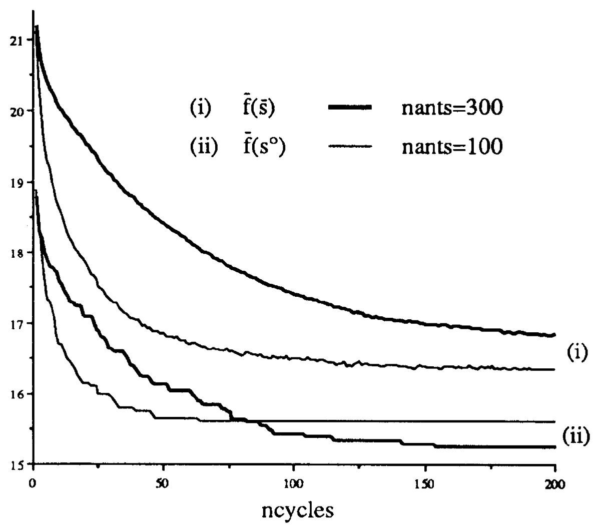
Results

Next runs of ANTCOL will be governed by procedures ANT_RLF(2,1) and ANT_DSATUR(1). The latter are based on the parameters which yielded the best results in Table 7, that is to say $\alpha_1 = 2$, $\beta_1 = 4$, $\rho = 0.5$ for ANT_RLF(2,1) and $\alpha_1 = 1$, $\beta_1 = 4$, $\rho = 0.5$ for ANT_DSATUR(1). In order to reduce the running time when dealing with large graphs, ANTCOL is interrupted automatically after $\text{ncycles}_{\max} = 50$ cycles. Colourings achieved with colonies of size $\text{nants} \in \{100, 300\}$ are presented in Table 8.

We notice that procedure ANT_RLF is much more efficient than ANT_DSATUR when the search is limited to 50 cycles. The superiority of ANT_RLF is obvious from a computational point of view also. This speed difference is surprising because RLF is much slower than DSATUR (see Table 6). The reason is that the number of vertex candidates is larger and the updating of the trails $\tau_2(s[k-1], v, c)$ requires more time at each step of ANT_DSATUR. For a given number of vertices, the running time of ANTCOL decreases slightly as the density increases. This phenom-



a) ANT_RLF(2,1) α_1=2, β_1=4, ρ=0.5
nants $\in$ {100, 300}



b) ANT_DSATUR(1) α_1=1, β_1=4, ρ=0.5
nants $\in$ {100, 300}

Figure 3 Evolution of the quality of solutions with ANTCOL.

enon can be explained in the following way. When the density increases the mean size of the stable set V_c decreases and thus the new trails $\tau_2(s[k\text{–}1], v, c)$ can be computed faster.

Figure 2 shows that procedure ANT_DSATUR needs to be governed by a large number of ants in order to be really efficient. A colony of size *nants* < n is generally not sufficient to produce satisfactory results after 50 cycles. Indeed significantly better results were achieved with a simple run of DSATUR (see Table 6).

The behaviour of procedure ANT_DSATUR is very surprising on graphs $G_{100,0.4}$ and $G_{100,0.5}$. We notice that the quality of the colourings achieved decreases significantly as the number of ants goes from 100 to 300. Experiments illustrated in Figure 3 help us to understand this phenomenon. Procedure ANT_RLF and ANT_DSATUR are used again to colour the 20 graphs in the class $\bar{G}_{100,0.5}$ with colony sizes equal to 100 and 300. $\bar{f}(\tilde{s})$ and $\bar{f}(s^\circ)$ are average values which measure respectively the mean quality of the colourings at the end of a cycle and the quality of the best colouring s° reached previously (ncycles $= 1, 2, \ldots,$ 200).

Figure 3(b) shows that ANT_DSATUR converges more slowly towards good solutions when the number of ants is high. It looks like the handling of too many elements of information (when compared to the size of the graph) does not really help in identifying faster pairs of vertices to be coloured the same in a good colouring. The 20 final colourings s° were all obtained in less than 64 cycles with 100 ants whereas 154 cycles were necessary with 300 ants. Even though ANT_DSATUR yields systematically colourings with a superior average quality when nants $= 100$, a colony of size 300 is able to find better colourings s° after 87 cycles. This phenomenon is proper to procedure ANT_DSATUR. Figure 3(a) shows that the number of ants does not affect the shape of the curves $\bar{f}(\tilde{s})$ and $\bar{f}(s^\circ)$ when dealing with procedure ANT_RLF.

With the exception of graphs on 100 and 300 vertices, the quality of the solutions reported in Table 8 is rather low when compared with the CPU time required to produce them. Substantial improvements were observed with larger values of nants and ncycles$_{\text{max}}$. The results obtained in these circumstances have not been detailed in this paper because no extensive algorithmic testing was carried out. Indeed such experiments require an excessive computational effort in comparison with the latest graph colouring algorithms reported in the literature[14,15,23,25]. Before concluding, let us point out that ANTCOL was able to colour the 20 graphs in the class $G_{100,0.5}$ with an average of 15.05 colours. This result was obtained after 837.8 s with procedure ANT_DSATUR and parameters $\alpha_1 = 1$, $\beta_1 = 2$, $\rho = 0.5$, nants $= 260$, ncycles$_{\text{max}} = 100$. The best known average number of colours for these graphs is equal to 14.95 and has been obtained by Fleurent and Ferland[14] and Costa, Hertz and Dubuis[15].

Concluding remarks

Despite the impressive abilities of today's computers, many combinatorial optimization problems still exist which requires a heuristic search approach in the general case. In this paper we show that the way insects coordinate their activities to perform a specific task can be used for tackling ATPs and more specifically graph colouring problems. The potential of the ant algorithm we have developed is investigated on various random graphs. Although they do not match the best results achieved up to now, the colourings we have obtained are satisfactory. The latter were shown to be significantly better than those obtained by sophisticated constructive methods.

The graph colouring algorithms described by Chams, Hertz and de Werra[24], Hertz and de Werra[25], Fleurent and Ferland[14] and Costa Hertz and Dubuis[15] do not attempt to minimize directly the number of colours used in a colouring. Instead they deal with a fixed number q of colours and they try to minimize the number of edges whose endpoints are coloured the same. In this case, the set of feasible solutions is all partitions of V into q sets. The objective function $f(\mathbf{x})$ counts the number of monochromatic edges in the solution $\mathbf{x}$. Everytime the algorithm succeeds in finding a q-colouring it is applied again with a number of colours q reduced by one unit.

A similar fixed-q approach governed by an ant algorithm was also investigated in this study. Vertices are coloured by choosing repeatedly the uncoloured vertex v with a maximum degree of saturation dsat(V) among the uncoloured vertices. The choice of the colour $c(v) \in \{1, 2, \ldots, q\}$ is made probabilistically on the basis of the trail factor and the desirability factor. Like in ANTCOL trails are visible between pairs of non adjacent vertices only. The intensity M_{rs} is related to the quality of the previous solutions obtained by colouring vertices v_r and v_s identically. At the end of a cycle the values M_{rs} are updated by laying a new mark of intensity $1/f(\mathbf{x})$ between v_r and v_s ($[v_r, v_s] \notin E$) everytime v_r and v_s are coloured the same in a solution $\mathbf{x}$. The desirability $\eta_2(s[k-1], v, c)$ of colour c for an uncoloured vertex v is inversely proportional to the number of vertices adjacent to v and already assigned to colour c.

The results achieved by such a fixed-q ant algorithm were not reported in this paper because they do not differ significantly from those produced by ANTCOL.

Our experience with the two adaptations of ANTCOL described in this paper leads us to the following general comments:

(1) ANTCOL is an easily implementable algorithm which can be modified, with little effort, to tackle a variety of ATPs where there are competing items and resources. The principle of laying a trail between pairs of items assigned to the same resource is well suited for any ATP in which the number of items is significantly larger than the number of resources.

(2) The choice of the size of the colony is essential when running ANTCOL on a non parallel machine. A good compromise needs to be found between the running time and the solution quality. Experience shows that the value *nants* does not have a significant influence on the quality of the results when it is large enough. Beyond a certain value *nants*$^+$, the extra information we gain by adding some ants in the colony does not really help.

(3) The experiments carried out in this paper make up a first step in the field of graph colouring techniques using the collective structures of an insect society. Such an approach looks very promising even though the results obtained can be further improved. Up to now models of collective problem solving have been little studied in the literature. In our opinion they define a very interesting research framework which is worth investigating deeper. Further studies should concentrate on convergence theorems in order to better understand the functioning of these emerging techniques.

(4) Due to its asynchronicity and intrinsic parallel nature, ANTCOL is well suited for parallel computation. The use of a parallel MIMD machine would have reduced the execution time of the algorithm by a factor equal to the size of the colony approximately. Investigations to evaluate the speed-up of the algorithm have not been carried out in this study. Evolutionary algorithms form an emerging framework in computer programming that could challenge in the near future very sophisticated algorithms. The results we have obtained with ANTCOL are encouraging for future research.

Acknowledgements—The authors would like to thank Charles Fleurent for having provided the random graph generator used in a recent study[14]. Support of the *Fonds National de la Recherche Scientifique* under grant No. FN 21-36173.92 is gratefully acknowledged.

References

1 Colorni A, Dorigo M and Maniezzo V (1991). Distributed optimization by ant colonies *Proceedings of the first European Conference on Artificial Life* Paris, ed. by FJ Varela and P Bourgine, MIT/Press/Bradford Books, Cambridge, Massachussetts: 134–142.

2 Colorni A, Dorigo M and Maniezzo V (1992). An investigation of some properties of an ant algorithm *Proceedings of the Second Conference on Parallel Problem Solving from Nature*). Brussels, ed. by R Maenner and B Manderick, North-Holland, Amsterdam: 509–520.

3 Maniezzo V, Colorni A and Dorigo M (1994). The ant system applied to the quadratic assignment problem. Technical Report 94/28, IRIDIA, Université Libre de Bruxelles. Belgium.

4 Colorni A, Dorigo M, Maniezzo V and Trubian M (1994). Ant System for job shop scheduling *Belgian J Opns Res, Stat and Comp Sci* **34**: 39–53.

5 Ferland JA, Hertz A and Lavoie A (1996). An object oriented methodology for solving assignment type problems with neighbourhood search techniques *Opns Res* **44**: 347–359.

6 Garey MR and Johnson DS (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company: New York.

7 Costa D (1995). Méthodes de résolution constructives, séquentielles et évolutives pour des problèmes d'affectation sous contraintes, Ph.D. Thesis no. 1411, Département de Mathématiques, Ecole Polytechnique Fédérale de Lausanne, Switzerland.

8 Hertz A (1995). A colourful look on evolutionary techniques. Report 95/10, Départment de Mathématiques, Ecole Polytechnique Fédérale de Lausanne, Switzerland.

9 Goldberg DE (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley: New York.

10 Liepins GE and Hilliard MR (1989). Genetic algorithms: foundations and applications *Annals of Opns Res* **21**: 31–58.

11 Glover F (1994). Genetic Algorithms and scatter search: unsuspected potentials *Stat and Comp* **4**: 131–140.

12 Theraulaz G, Goss S, Gervet J and Deneubourg JL (1991). Task differentiation in polistes wasp colonies: a model for self-organizing groups of robots. *Simulation of Adaptive Behavior: From Animals to Animats*, MIT Press/Bradford Books, Cambridge, Mass. 346–355.

13 Davis L (ed) (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York.

14 Fleurent C and Ferland JA (1996). Genetic and hybrid algorithms for graph coloring. *Annals of Opns Res*, **63**: 437–461.

15 Costa D, Hertz A and Dubuis O (1995). Embedding of a sequential algorithm within an evolutionary algorithm for coloring problems in graphs *J of Heuristics* **1**: 105–128.

16 Dunstan FDJ (1976). Sequential colourings of graphs *Congressus Numerantium* **15**: 151–158.

17 Johri A and Matula DW (1982). Probabilistic bounds and heuristic algorithms for coloring large random graphs. Technical Report 82-CSE-06, Southern Methodist University, Department of Computing Science, Dallas.

18 Morgenstern C (1990). Algorithms for general graph coloring. Ph.D. Thesis, University of New Mexico, Department of Computer Science, New Mexico.

19 Bralaz D (1979). New methods to color vertices of a graph *Communications of the ACM* **22**: 251–256.

20 Leighton F (1979). A graph coloring algorithm for large scheduling problems *J Res National Bureau of Standards* **84**: 489–505.

21 Bollobas B (ed) (1995). *Random Graphs*. Academic Press: New York.

22 L'Ecuyer P (1988). Efficient and portable combined random number generators *Communications of the ACM* **31**: 742–774.

23 Johnson DS, Aragon CR, McGeoch LA and Schevon C (1991). Optimization by simulated annealing: an experimental evaluation, part II, graph coloring and number partitioning *Op Res* **39**: 378–406.

24 Chams M, Hertz A and de Werra D (1987). Some experiments with simulated annealing for coloring graphs *Eur J Op Res* **32**: 260–266.

25 Hertz A and de Werra D (1987). Using tabu search for graph coloring *Computing* **39**: 345–351.

Received February 1996;
accepted September 1996 after one revision