An Ant Algorithm for the Graph Colouring Problem

Francesc Comellas and Javier Ozón

Departament de Matemàtica Aplicada i Telemàtica, Universitat Politècnica de Catalunya

C/ Jordi Girona 1-3, Campus Nord C3, 08034 Barcelona, Spain

Tel: 93 401 60 09   Fax: 93 401 59 81

comellas@mat.upc.es ozzy@mat.upc.es

ANTS'98 - From Ant Colonies to Artificial Ants: First International Workshop on Ant Colony Optimization
Brussels, Belgium, October 15–16, 1998. © F.Comellas and J. Ozón.

An Ant Algorithm for the Graph Colouring Problem

F. Comellas, J. Ozón
Departament de Matemàtica Aplicada i Telemàtica
Universitat Politècnica de Catalunya
`comellas@mat.upc.es ozzy@mat.upc.es`

Abstract
This paper describes an ant algorithm that colours graphs in an efficient way. First, we describe the graph colouring problem and the ant algorithm. We also present some results and propose a simple generalisation of the algorithm that might allow its application to other assignment problems. Finally, a short study of the algorithm operators explains its performance and shows that it may be seen as a parallel variation of tabu search, with an implicit memory instead of an avoidance list.

1 An ant algorithm for colouring graphs

Many assignment and combinatorial problems may be formulated in terms of graph colouring. A proper colouring of a graph $G = (V, E)$ is a function from the nodes of the graph to a set C of colours such that any two adjacent nodes have different colours. If k is the cardinal of C, we say that G is k-coloured. The graph colouring problem is NP-complete. Hence, we need to use approximate algorithmic methods to obtain solutions close to the absolute minimum in a reasonable execution time. Most of the algorithms used to solve the colouring problem are summarised in Hurley [4], and include sequential assignment methods, simulated annealing and genetic algorithms.

The ant algorithm that we propose for the colouring problem is a multi-agent system based on the idea of parallel search. In this algorithm a given number of ants move around the nodes of the graph and changes the colour of each visited node according to a local criterion. At a given iteration each ant moves from the current node to the adjacent node with the maximum number of violations (see Fig 1), and replaces the old colour of the node with a new colour that minimises this number. For a given node i, the number of violations is computed as the number of adjacent nodes to i with the same colour than i. This action is randomly repeated for each ant: the ant moves to the

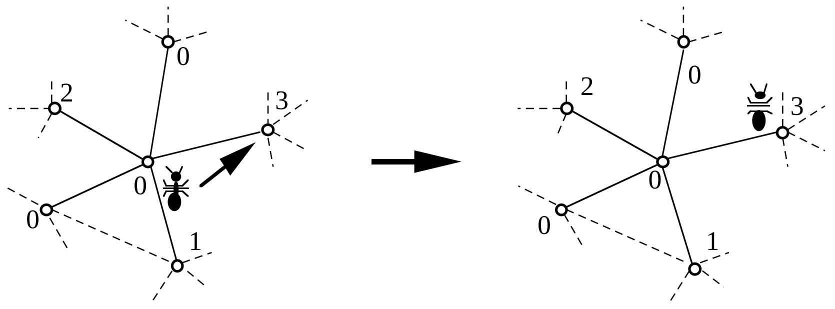

Figure 1: Movement of an ant towards the worst local node (the figures indicate the number of violations of each node).

worst adjacent node with a certain probability p_n (otherwise it moves to any other adjacent node randomly chosen), and assigns the best possible colour with a probability p_c (otherwise any colour is assigned at random). The probabilistic nature of the algorithm allows the ants to escape from local minima and obtain bounds close to the absolute minimum. This process, which is carried out simultaneously by the set of ants, is repeated until the optimal solution is found or the algorithm converges. The number of ants that move along the graph is an adjustable parameter and increases with the order of the graph.

In [2] it is shown that the ant algorithm clearly outperforms other classical methods such as simulated annealing and genetic algorithms. On the other hand, in Table 1 we present a comparison between our algorithm and another ant algorithm proposed by Costa and Hertz [3]. The ant algorithm was tested on random graphs with 100, 300, 500 and 1000 nodes (ten graphs for each order) in which there exists an edge between any pair of nodes with a probability 0.5. The better performance of our algorithm may come from the fact that the ants move over the graph instead of the solution space, and by the simple way used by any ant to "remember" former actions of the algorithm.

nodes	100	300	500	1000
C&H	15.2	35.7	55.6	111.0
`ants`	14.9	34.8	53.9	99.5

Table 1: A comparison between our algorithm, `ants`, and Costa and Hertz ant algorithm [3]. We give the average of the number of colours used in each case.

One important concern for any algorithm is the extent of its possible applications. Although it is possible to design specific algorithms for most combinatorial problems, there are some general schemes, such as tabu search, simulated annealing or the ant algorithm by Colourni et al. [3], which can be successfully applied to a wide range of combinatorial problems. The understanding of the basic mechanisms of our algorithm will help to apply it to further problems.

The ant algorithm can be understood as a directed simulated annealing or a singular version of tabu search. In our algorithm, we have at each move an specific solution (the actual colouring of the graph, i.e., a string of $|V|$ integers, where $|V|$ is the number of nodes and the integer at position i corresponds to the colour assigned to the ith node of the graph at that time). This colouring is not normally proper. At each step the ant moves to a node and changes its colour according to a local optimisation criterion. In the solution space, this action corresponds to select a position in the current solution and a new integer that will be assigned to this position. These two operations are the same than in simulated annealing, although in our ant algorithm they are performed trying to minimise a local function, whereas in simulated annealing they are done at random. Besides, this search mechanism may also be seen as a modification of the tabu search algorithm with the difference that in tabu search a list (which holds a memory of former actions) is updated at each step and used to decide whether a move is forbidden or not, whereas in our algorithm this decision is based on a local study of the graph.

Memory may be seen not as an explicit database with information from past events, but as a strategy that will benefit from the recent history of the algorithm. Our ant algorithm has actually this sort of implicit memory which is not specifically saved in some file although it takes into account past events when making a new decision. As shown in Fig 1, once an ant k has moved from node i to node j and has changed its colour, it remains there until the next iteration, when it will move again towards one of the nodes of j's neighbourhood. Implicit memory means that the ant "remembers" at each step the former changes performed by the algorithm, and takes into account that these changes may have modified the cost function of the neighbourhood of j. Therefore, at the next step, the ant k will normally try to arrange the colouring of the worst adjacent node to j. Any single action depends strongly on the last move of each ant; and this dependence reinforces the results of recent modifications. This implicit memory is not continuously kept by the algorithm in some matrix or list. It only exists over the graph and on the actual position of each ant, as it happens in a real ant society with pheromones. Therefore there is a reduction in the global computational effort in relation to other multi-agent algorithms, since the algorithm does not have to evaluate all the nodes of the graph.

2 Further applications

To apply our algorithm to other assignment problems (a general class of problems where a set of resources must be assigned to a set of items with some restrictions) we have classified the different actions which are performed in our algorithm in three generic operators:

Op 1.a. From the actual node (item) move to the worst node (item) of the neighbourhood.
Op 1.b. Assign the best local colour (resource) to the new node (item).

Op 2. Continue the search in the neighbourhood (short term memory).
Op 3. Parallel operator (several ants work simultaneously).

To evaluate the importance of these operators we have done some modifications to the initial algorithm excluding some of the operators from it, and we have tested the resultant algorithms on random graphs. Table 2 shows the description of the algorithms, their features, and for each case the missing operator from the list above.

Alg	**description**	**performing**	**missing op.**
a-alg.	each ant jumps at random to a neighbour and changes to the best local colour (cbc)	does not work	1a (and so 2)
b-alg	each ant jumps at random to any node and cbc	does not work	1a (and so 2)
c-alg	each ant moves either to the worst neighbour or randomly to other neighbour and cbc	ok	none
cI-alg	each ant moves either to the worst neighbour or randomly to another neighbour and changes to any colour	does not work	1b
cII-alg	as c, but not in parallel	poor	3
d-alg	each ant moves either to the worst neighbour or randomly to any other node and cbc	ok, but worse than 3	2 is weaken
e-alg	each ant moves randomly to any node, searches the worst neighbour and cbc	quite poor	2

Table 2: A comparison between different ant algorithms with missing operators.

As shown in Table 2, the best algorithm is c-alg, i.e., the original ant algorithm [2], whereas if we eliminate one of the operators 1a, 1b or 2, the algorithm which results does not work at all. Therefore, to obtain an efficient algorithm these three operators should be used together. While operators 1a and 1b benefit from local information at each iteration, operator 2 reinforces the memory of the algorithm and gives more restrict colourings of any given graph.

The three operators, as described above, can be applied to any assignment problem without loss of generality by defining previously a neighbourhood for each item of the problem (normally it should be the subset of items which had some restriction related to it), and a local function to establish some local criterion. Although this local function may be the global function, it is advisable (to increase the algorithm speed) to define this local function which would be computed at each step according only to the neighbourhood of the current node.

3 Conclusions

In this paper a new ant algorithm has been proposed for the graph colouring problem. It has been shown that this algorithm, that may be implemented in less than 200 lines of code, performs better than other methods. Furthermore we have shown how to generalise it and possible applications to a wide class of assignment problems.

4 Bibliography

[1] Abril J., Comellas F., Cortés A., Ozón J., and Vaquer M., A Multi-agent System for Frequency Assignment on Cellular Radio Networks, submitted to IEEE on Vehicular Technology
[2] Comellas F. and Ozón J., Graph colouring algorithms for assignment problems in radio networks, Applications of Neural Networks to Telecommunications 2. Edit. J. Alspector, R. Goodman y T.X. Brown, Lawrence Erlbaum Ass., pp. 49-56, 1995. http://www-mat.upc.es/~comellas/radio/radio.html
[3] Costa D. and Hertz A., Ants can colour graphs, Journal of the Operational Research Society (1997) 48, 295-305
[4] Hurley S., Thiel S.U. and Smith D.H., Frequency Assignment Algorithms, Final Report Year2 1996/97, http://www.cs.cf.ac.uk/User/SteveHurley/Ra/year2