An Experimental Study of the Simple Ant Colony Optimization Algorithm

MARCO DORIGO[†] and THOMAS STÜTZLE[‡]

[†]IRIDIA
Université Libre de Bruxelles
Avenue Franklin Roosevelt 50, CP 194/6, 1050 Brussels
BELGIUM

[‡]Department of Computer Science – Intellectics Group
Darmstadt University of Technology
Alexanderstr. 10, D-64283 Darmstadt
GERMANY

Abstract: - Ant Colony Optimization (ACO) is a recently proposed metaheuristic inspired by the foraging behavior of ant colonies. Although it has been experimentally shown to be highly effective on a number of static and dynamic discrete optimization problems, only limited knowledge is available to explain why the metaheuristic is so successful. In this paper we propose a simple framework that allows the investigation of some basic properties of ACO and we report about some experiments and what we learned from them.

Key-Words: - Ant Colony Optimization, Simple-ACO algorithm, shortest path problem

1 Introduction

Ant algorithms [3, 4, 1, 2] are multi-agent systems in which the behavior of each single agent, called *artificial ant* or *ant* for short in the following, is inspired by the behavior of real ants. The Ant Colony Optimization (ACO) metaheuristic is one of the most successful examples of ant algorithm [5], and has been applied to many types of problems, ranging from the classical traveling salesman problem to routing in telecommunications networks.

ACO algorithms have been inspired by an experience run by Goss et al. [10] using a colony of real ants. They ran a number of experiments with a laboratory colony of Argentine ants (*Iridomyrmex humilis*) using a double bridge connecting a nest of ants and a food source to study the ants' pheromone trail laying and following behavior in controlled experimental conditions. Particularly interesting is the case in which one branch of the double bridge is longer than the other: They found that, although in the initial phase random oscillations could occur, in most experiments all the ants ended up using the shorter branch. The connection between this experience and the design of the ACO metaheuristics has been described in a number of papers [8, 9, 6, 7].

In this article we run experiments with an elementary ACO algorithm, called Simple-ACO (S-ACO), in which artificial ants' behavior is very similar to that of their natural counterpart (i.e., many of the features that are often added to ACO artificial ants to obtain high performing algorithms for hard combinatorial optimization problems are not implemented in S-ACO). The algorithm is experimentally tested using the example problem of finding shortest paths in graphs. Although the shortest path problems can be solved with deterministic algorithms in polynomial time, it is an interesting problem for studying the behavior of ACO algorithms, because (i) it is the problem solved by real ant colonies, (ii) the shortest path problem is very simple and the algorithm behavior is not obscured by technicalities of the

problem under consideration, and (iii) we expect features which are important to solve this easy problem to be even more important when attacking much more difficult combinatorial optimization problems.

In the following we first briefly describe S-ACO, then present some simulation results, and eventually discuss what we learned from these results.

2 Simple-ACO

The problem that we consider in the following is finding a shortest path on a graph $G = (N, A)$. S-ACO exploits a set of variables $\mathcal{T} = \tau_{ij}(t)$ called *artificial pheromone trails* that are associated to the arcs (i, j) of the graph G. Pheromone trails are read and written by the ants. The amount (intensity) of each pheromone trail is proportional to the utility, as estimated by the ants, of using the corresponding arc to build good solutions.

In S-ACO each ant builds, starting from the source node, a candidate solution to the shortest path problem by applying a step-by-step decision policy. At each node local pheromone information, which is stored at the node itself and/or on its outgoing arcs, is read (sensed) by the ant and used in a stochastic way to decide to which node to move next: when located at a node i an ant k uses the pheromone trails τ_{ij} to compute the probability p_{ij}^k of choosing j as the next node:

$$p_{ij}^k = \begin{cases} \dfrac{\tau_{ij}^\alpha}{\sum_{j \in \mathcal{N}_i^k} \tau_{ij}^\alpha} & \text{if } j \in \mathcal{N}_i^k \\ 0 & \text{if } j \notin \mathcal{N}_i^k \end{cases} \quad (1)$$

where $\mathcal{N}_i^k$ is the feasible neighborhood of ant k when in node i. (At the beginning of the search process, an amount of pheromone $\tau_0 = 1$ is assigned to all the arcs of the graph G to avoid division by zero.) In S-ACO the feasible neighborhood $\mathcal{N}_i^k$ of ant k located in node i contains all the nodes directly connected to node i, except for the predecessor of node i (that is, the last node ant k visited before moving to i). This way the ants avoid returning to the same node they visited immediately before node i. Only in case $\mathcal{N}_i^k$ is empty (corresponding to a dead end in the graph), node i's predecessor is included into $\mathcal{N}_i^k$. Note that this decision policy can easily cause the ants to enter a loop.

An ant repeatedly hops from node to node using its decision policy until it eventually reaches the destination node. Due to differences among the ants' paths, the time step at which ants reach the destination node may differ from ant to ant (ants traveling on shorter paths will reach their destinations faster).

Once reached the destination node, the ants eliminate loops they might have done while searching for the destination node and then retrace step by step the same, loop-free, path backward to the source node in a deterministic way.

While an ant returns to the source, it adds pheromone to the edges it traverses: during its return travel the generic ant k deposits an amount $\Delta\tau^k$ of pheromone on each arc it has visited (in the loop-free path). In particular, if ant k at time t traverses the arc (i, j), it updates the pheromone value τ_{ij} as follows:

$$\tau_{ij}(t) \leftarrow \tau_{ij}(t) + \Delta\tau^k \quad (2)$$

By this rule an ant using the arc connecting nodes i and j increases the probability that forthcoming ants will use the same arc in the future.

An important aspect of S-ACO (and of ACO algorithms in general) is how the value $\Delta\tau^k$ is chosen. In the simplest case, this can be a same constant value for all the ants. In this case the only effect which works in favor of the detection of a shorter path is the so called *differential path length* effect observed also in real ants: ants which have detected a shorter path can deposit pheromone earlier than those traveling on a longer path; therefore, the short path becomes more desirable quicker than longer paths and this in turn determines an increase in the probability that forthcoming ants will choose it (this is an example of *autocatalysys* or *positive feedback*).

A more sophisticated way of updating the pheromone trails is to make $\Delta\tau^k$ a function of the quality of the solution generated, that is, of the path length: the shorter the path the more the pheromone deposited by the ant. Generally, we will require the amount of pheromone deposited by an ant to be a non-increasing function of the path length. In S-ACO, in particular, an ant deposits an amount of $1/L^k$, where L^k is the length of ant k's path.

To avoid a quick convergence of all the ants towards a sub-optimal path, pheromone trails "evap-

orate". It is interesting to note that although also real pheromone trails evaporate, evaporation does not seem to play an important role in shortest path experiments with real ants. The fact that, as we will see later in the experiments section, evaporation is important to get good results with artificial ants is probably due to the fact that the optimization problems tackled by artificial ants are much more complex than those real ants can solve. A mechanism that allows a form of "learning" of the problem structure seems therefore to be necessary for artificial ants. Evaporation allows the learning of new policies by allowing to forget bad choices made in the past.

Evaporation is carried out decreasing pheromone trails at exponential speed. In practice, at each iteration of the algorithm the following equation is applied to all pheromone trails:

$$\tau \leftarrow (1-\rho)\cdot\tau \tag{3}$$

where $\rho \in (0,1]$ is a paremeter.

It is interesting to note that, in experiments with real ants, pheromone trail evaporation does not play any role. Hence, in one experiment we test the performance of S-ACO when setting $\rho = 0$, that is, when there si no evaporation. We will see that the more complex the graph the more important is the role played by pheromone evaporation to obtain the desired behavior of convergence on a shortest path. Note that if the pheromone trails evaporate completely, that is, when $\rho = 1$, then the algorithm is actually reduced to a random search.

3 Experiments with S-ACO

In the following we run a few experiments on two simple graphs with the goal of studying how changing some aspects of S-ACO results in a different behavior of the algorithm. The graphs used are the *double bridge* of Figure 1 and a more complex graph shown in Figure 2.

The behavior of S-ACO is judged with respect to convergence towards the shortest path. By convergence we mean the situation in which, as the algorithm runs for an increasing number of iterations, the ants' probability of following the arcs of the shortest path increases — in the limit to a point where the probability of selection for arcs of the shortest path becomes arbitrarily close to one while for all the other arcs it becomes close to zero.

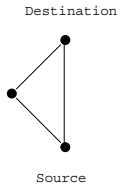

Figure 1: Double bridge experimental setup. The ants can choose to go from the source node to the destination node either via the short path on the right or via the longer path on the left.

Note that the choice of judging the algorithm using convergence as defined above instead of more standard performance indexes like the time or the number of iterations necessary to find the optimal solution is consistent with the goal of this paper, that is, understanding the relationship between design choice and algorithm's behavior. In fact, in such simple graphs as those used here, the shortest path is found very quickly because of the large number of ants compared to the relatively small search space. On the contrary, when attacking more complex problems like $\mathcal{NP}$-hard optimization problems or routing in dynamic networks, the way experimental results are judged is different. In $\mathcal{NP}$-hard optimization problems the main goal is to find quickly very high quality solutions and therefore we are interested mainly in the solution quality of the best solution(s) found by the ACO algorithm. In routing in dynamic networks the algorithm has to be able to react rapidly to changing conditions and to maintain exploration capabilities so that it can effectively evaluate alternative paths which, due to the dynamics of the problem, may become more desirable; in both cases we will need a different definition of algorithm convergence.

Double bridge setup: Number of ants versus solution quality based pheromone update

In a first experiment we applied the S-ACO algorithm to the double bridge setup depicted in Figure 1. We report results for two experiments with S-ACO:

1. Run S-ACO with different values for the number of ants and setting parameter $\alpha = 2$ (this

Table 1: The table entries give the number of times S-ACO converged to the long path in 100 independent trials for varying values of m and with $\alpha = 2$. The columns give the number m of ants in the colony. The first row (`ns`) concerns results obtained performing pheromone updates without considering path length; the second row (`sq`) concerns results obtained performing pheromone updates proportional to path length.

m	1	2	4	8	16	32	64	128	256	512
ns	50	42	26	29	24	18	3	2	1	0
sq	18	14	8	0	0	0	0	0	0	0

is the same value as used in the equations approximating real ants' behavior as identified in [10]).

2. Same experiment as above except that the ants deposit an amount of pheromone which is inversely proportional to the length of the path they have found.

We ran 100 trials and for each experiment and each trial S-ACO was stopped after 1000 steps done by each ant. Evaporation was set to $\rho = 0$. At the end of the trial we check whether the pheromone trail is higher on the short or on the long path. In Table 1 we then report the percentage of experiments in which the pheromone trail was higher on the long path. We could verify that reporting this number was enough because S-ACO showed convergence behavior for the given parameter settings. Table 1 gives the results of the two experiments. Let us focus first on the results of experiment 1. For a small number of ants (up to, say, 32), S-ACO converges relatively often to the longer path. This effect is certainly due to fluctuations in the path choice in the initial iterations of the algorithm which can lead to a strong reinforcement of the long path. Yet, with an increasing number of ants, the number of times we observed this behavior decreases drastically and for large number of ants (here 512) we never observed this behavior in any of the 100 trials. The experiments also indicate that, as it could be expected, when using only one ant S-ACO shows very poor behavior: the number of ants has to be significantly larger than one to obtain convergence to the short path.

The results obtained with solution quality based pheromone update (experiment 2) are much better. As can be clearly observed in Table 1, S-ACO converges to the long path much less frequently than when pheromone updates are independent of the solution quality. With only one ant, S-ACO converges in only 18 of 100 trials to the long path, which is significantly less than in experiment 1, and with as few as 8 ants, it did not convergence anymore to the long path.

In some additional experiments, we examined the influence of the parameter α on the convergence behavior of S-ACO, in particular investigating the cases where α was changed in step sizes of 0.25 from 1 to 2. Again, the behavior was dependent on whether solution quality based pheromone update was used or not. In the first case we found that increasing α had negative effects for the convergence behavior, while in the second case the results were rather independent of the particular value of α. In general, we found that the smaller α, the easier it is for the algorithm to converge towards the shorter path, if the number of ants is held fixed. This is intuitively clear, because large values of α tend to amplify the influence of initial random fluctuations, while small values of α tend to limit the influence of poor decisions done in the first iterations, that can therefore be more easily forgotten via the pheromone evaporation mechanism.

As in the case of real ants, *autocatalysis* and *differential path length* are at work to favor the emergence of short paths. While the results with S-ACO indicate that the differential path length effect alone can be enough to let S-ACO converge to the optimal solution on small graphs, relying on this effect as the main driving force of the algorithm comes at the price of having to use large colony sizes, which determines long simulation times. In addition, we expect the effectiveness of the differential path length effect to strongly decrease with increasing problem complexity. This is what is tested in the experiment of the next subsection.

Complex problem: Pheromone evaporation
In a second experiment we run S-ACO on the graph given in Figure 2. Based on the results for the double bridge experiment, we modified the basic configuration of S-ACO so that the ants always deposit an amount of pheromone which is the in-

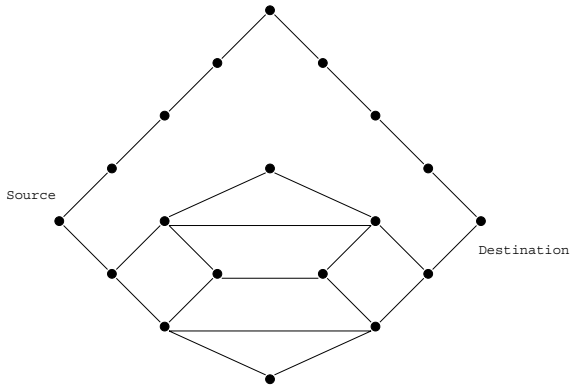

Figure 2: An extension of the double bridge setup. Here ants can choose between the upper path which is longer but once chosen can be traversed without any further decision to be made, and the lower set of paths that are shorter but require a number of decisions to be done, with the risk of entering loops.

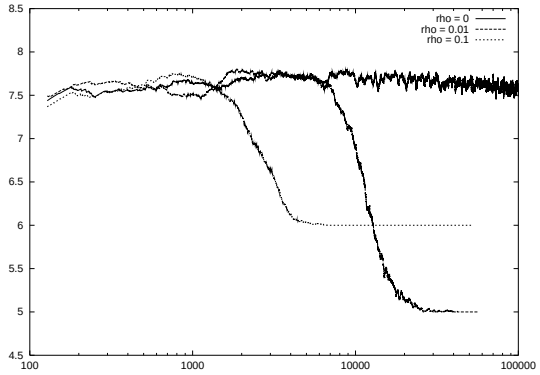

Figure 3: The graph plots the moving averages (given on the y-axis) of the ants' path length for the graph of Figure 2 as a function of the number of completed paths (given on the x-axis). We give plots for not using evaporation (`rho = 0`), small evaporation (`rho = 0.01`), and large evaporation (`rho = 0.1`). The trials were limited to 5000 steps by each single ant; $\alpha = 1$ and $m = 128$.

verse of the length (after loop elimination) of the path they generated. To evaluate the behavior of the algorithm we observe the development of the path lengths found by the ants. In particular, we plot the moving averages of the path lengths after loop elimination (moving averages are calculated using the $4{\cdot}m$ most recent paths found by the ants, where m is the number of ants). In other words, in the graph of Figure 3 a point is plotted each time an ant has completed a journey from the source to the destination and back (the number of journeys is on the x-axis), and the corresponding value of the point on the y-axis is given by the length of the ant's path after loop elimination.

In this experiment we focus on the influence that pheromone trail evaporation has on the convergence behavior of S-ACO. We have run experiments with S-ACO and different settings for the evaporation rate of $\rho \in \{0, 0.01, 0.1\}$ ($\alpha = 1$ and $m = 128$ in all experiments). If $\rho = 0$, no pheromone evaporation takes place. Note that an evaporation rate of $\rho = 0.1$ is rather large, because evaporation takes place after every iteration of the S-ACO algorithm: after 10 S-ACO iterations, which corresponds to the smallest number of steps that an ant needs to build the shortest path and to come back to the source, roughly 65% of the pheromone trail on each arc evaporates, while with $\rho = 0.01$ this evaporation is reduced to around 10%.

Figure 3 gives the observed moving averages. Although the graphs are relative to one single run of the algorithm, they are representative of the typical algorithm behavior. If no evaporation is used, the algorithm does not converge: neither to the shortest path nor to any longer path, as can be understood by observing that the moving average is around 7.5, which does not correspond to the length of any path. With these parameter settings, this result typically does not change if the run last a much higher number of iterations. With pheromone evaporation, the behavior of S-ACO is significantly different. After a short transitory phase, S-ACO converges to a single path: either the shortest one (the moving average takes the value 5 for $\rho = 0.01$) or the path of length 6 for $\rho = 0.1$. A closer examination of the results revealed that at convergence all ants built loop-free paths of the indicated length.

In further experiments with S-ACO on this graph we have made the following general observations:

- Without solution quality based pheromone update, S-ACO gets trapped in the strongly suboptimal solution of length 8; the larger the parameters α or ρ are chosen, the faster S-ACO converges to this suboptimal solution. We conjecture that the reason for the suboptimal behavior is that ants are easily trapped

in cycles in the lower part of the graph while, once entered the upper part of the graph, they can easily reach the destination node. This has the effect of decreasing the power of the differential path length effect. This example shows once again that pheromone update based on the solution quality is important for convergence towards good solutions.

- The pheromone evaporation rate ρ can be critical. In particular, we observed that S-ACO often converged to suboptimal paths when evaporation was set to a too high value. For example, in 15 trials with ρ set to 0.2, S-ACO converged one time to a path of length eight, one time to a path of length seven and two times to a path of length six, while with setting ρ to 0.01, S-ACO converged in all trials to the shortest path.

- Large values of α generally result in a worse behavior of S-ACO because they give excessive importance to the initial random fluctuations.

In general, we noticed that as problems become more complex, the parameter settings of S-ACO become increasingly important to obtain convergence to the optimal solution.

4 Conclusions

This article reports results of experiments run with S-ACO, a simplified ACO algorithm for finding shortest paths in graphs. The experimental results allow to derive important conclusions for the application of ACO algorithms to much more difficult combinatorial optimization problems. These are that (i) the differential path length effect alone is not effective enough to allow to solve effectively large optimization problems, (ii) solution quality based pheromone update by the ants is important to allow a fast convergence of S-ACO, (iii) large values for parameter α lead to a strong emphasis of initial, random fluctuations and to bad algorithm behavior, and (iv) pheromone evaporation is important when trying to solve more complex problems.

We expect that more detailed studies of S-ACO will lead to a better understanding of ACO features which are important to solve complex, real-world optimization problems.

Acknowledgments

Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Senior Research Associate. This work was partially supported by the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, NJ, 1999.

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz. Inspiration for optimization from social insect behavior. *Nature*, 406:39–42, 2000.

[3] M. Dorigo. *Optimization, Learning and Natural Algorithms* (in Italian). PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992. pp. 140.

[4] M. Dorigo, E. Bonabeau, and G. Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8):851–871, 2000.

[5] M. Dorigo and G. Di Caro. The Ant Colony Optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw Hill, London, UK, 1999.

[6] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[7] M. Dorigo and L. M. Gambardella. Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[8] M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.

[9] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41, 1996.

[10] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76:579–581, 1989.