APPEARED IN *PROCEEDINGS OF THE PARALLEL PROBLEM SOLVING FROM NATURE CONFERENCE (PPSN 92)*, BRUSSELS, BELGIUM, 1992, ELSEVIER PUBLISHING, 509–520.

An investigation of some properties of an "Ant algorithm"

Alberto Colorni
Dipartimento di Elettronica
Politecnico di Milano
20133 Milano, Italy

Marco Dorigo
PM-AI&R Project
Dipartimento di Elettronica
Politecnico di Milano
20133 Milano, Italy
dorigo@ipmel2.elet.polimi.it

Vittorio Maniezzo
PM-AI&R Project
Dipartimento di Elettronica
Politecnico di Milano
20133 Milano, Italy
maniezzo@ipmel2.elet.polimi.it

Abstract

We have used the metaphor of ant colonies to define "the Ant system", a class of distributed algorithms for combinatorial optimization. To test the Ant system we used the travelling salesman problem. In this paper we analyze some properties of Ant-cycle, the up to now best performing of the ant algorithms we have tested. We report many results regarding its performance when varying the values of control parameters and we compare it with some TSP specialized algorithms.

1. Introduction

Most of the work done in the distributed systems research area is justified by the rapid growth of importance of parallel computers. The interest of distributing an activity over several interacting but not centrally controlled agents is however twofold and regards both the quality of the solutions obtained and the speed of their obtainement. In fact, the underlying rationale for parallelizing an algorithm or for distributing it on a set of communicating computers is the hope to increase its speed of execution. This can be done only to a limited extent: at most, the use of a parallel computer can cut down the computational time by a factor that is linear with the number of processors.

The raw power of parallel hardware is best exploited by algorithms specifically designed for distribution, where simple activities can be performed in parallel (thus assigned to different processors) with rare local co-ordinating phases, which do not account for significant computational load. In this case, superlinear speedups have been reported [5].

In this paper we are interested in this second property of distributed systems, with special regard to the emergence of global properties from the interaction of many simple agents. The computations carried out by these simple agents can effectively be executed on parallel hardware, for instance by a transputer machine or by the simple processors of a massively parallel computer such as the Connection Machine.

In our work we call *ant* each simple agent and the distributed system (called Ant system, see [6]) is a set of ants cooperating in a common problem solving activity. A first result of our research is that the interaction of these ants generates synergetic effects. In fact, the quality of the solution obtained increases when the number of ants working on the problem increases (until an upper limit is reached) more than the corresponding quality obtained with the same number of non-communicating ants..

The problem we used to test our system is the well known Travelling Salesman Problem (TSP) often used as a benchmark for new general purpouse heuristics [1], [14], [17]. In this paper we are interested in the comparison of our heuristics with other ones, and not directly in proposing – at least at this stage of our research – a more efficient approach to the solution of TSP (which in fact has been solved optimally for problems of much higher order than those presented here).

In this paper we study Ant-cycle, a particular instance of the algorithms belonging to the Ant system class (other instances were proposed in [7]). Ant-cycle, as all the ant algorithms, is composed by a set of simple agents, ants, that cooperate to find a minimal tour.

In Ant-cycle every ant changes the system shared memory using a quantity (called *trail*) that is proportional to its global behavior. This makes Ant-cycle superior to other instances of ant algorithms (like Ant-density and Ant-quantity presented in [7]) that use strictly local information.

The properties of Ant-cycle that we repoert about are: the optimal parameters settings, how many ants to use to solve a given problem and how many cycles are required to find an optimal solution. Moreover, we compare Ant-cycle with some other specialized heuristics.

The paper is organized as follows: section 2 contains a brief description of the Ant system and of the Ant-cycle algorithm as it is currently implemented, together with the definition of the application problem (as the algorithm structure partially reflects the problem structure, we introduce them together). Section 3 reports diffusely on experiments run using the algorithm previously introduced. In section 4 we conclude discussing results obtained and some related work. We also give some hints about further research directions.

2. The Ant system

In this section we briefly summarize the main characteristics of the Ant system [6]. We first define the TSP problem.

INSTANCE: Consider a sequence of n towns $c_1, c_2, ..., c_n$, and for each pair (c_i, c_j) a distance $\mathbf{d}(c_i, c_j)$.

QUESTION: Find a permutation π of towns the minimizes the quantity

$$\left(\sum_{i=1}^{n-1} \mathbf{d}(c_{\pi(i)},c_{\pi(i+1)})\right) + \mathbf{d}(c_{\pi(n)},c_{\pi(1)})$$

Let $b_i(t)$ (i=1, ..., n) be the number of ants in town i at time t and let $m = \sum_{i=1}^{n} b_i(t)$ be the total number of ants (m doesn't depend on t).

In Ant-cycle an ant is an agent that:

- in the time interval between t and t+1 moves from town i to town j; the town j to go to is chosen with a probability that is a function of the distance $\mathbf{d}(c_i, c_j)$ and of the amount of trail present on the connecting edge;
- visits only towns that were not visited by it in the preceeding steps; this property, implemented to force ants to make legal tours, holds until a tour is completed; then the "ant memory" is reset and the ant is free again;
- after a tour (i.e., after it has visited all the towns) lays a substance, called *trail*, on each edge (i,j) visited.

Let $\tau_{ij}(t)$ be the *intensity of trail* on edge (i,j) at time t. After each ant has completed a tour[1], trail intensity becomes

$$\tau_{ij}(t+n)=\rho\cdot\tau_{ij}(t)+\Delta\tau_{ij}(t,t+n) \qquad (1)$$

where

ρ is a coefficient such that (1 - ρ) represents the evaporation of trail,

$$\Delta\tau_{ij}(t,t+n) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t,t+n)$$

$\Delta\tau_{ij}^{k}(t,t+n)$ is the quantity per unit of length of trail substance (pheromone in real ants) laid on edge (i,j) by the k-th ant between time t and t+n and is given by the following formula

$$\Delta\tau_{ij}^{k}(t,t+n)=\begin{cases}\dfrac{Q}{L^{k}} & \text{if k-th ant uses edge (i,j) in its tour}\\ 0 & \text{otherwise}\end{cases}$$

where Q is a constant and L^k is the tour length of the k-th ant..

The coefficient ρ must be set to a value <1 to avoid unlimited accumulation of trail. The intensity of trail at time 0, $\tau_{ij}(0)$, should be set to arbitrarily chosen values (in our experiments the same small value is chosen for every edge (i,j)).

A data structure, called *tabu list*[2], is associated to each ant in order to avoid that ants visit a town more than once: in the *tabu list* are memorized the towns already visited up to time t and ants are forbidden to visit them again before they have completed a tour. When a tour is completed the tabu list is emptied and every ant is free again to choose its way. We define $\mathbf{tabu}_k$ a vector containing the tabu list of the k-th ant, and $\mathbf{tabu}_k(s)$ the s-th element of the tabu list of the k-th ant (i.e., the s-th town visited by ant k in the current tour).

We call *visibility* η_{ij} the quantity $1/d_{ij}$, and define the transition probability from town i to town j for the k-th ant as

$$p_{ij}(t) = \begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum_{j\in \mathbf{allowed}}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}} & \text{if } j\in \mathbf{allowed}\\ 0 & otherwise\end{cases} \qquad (2)$$

where **allowed** = {j: $j\notin \mathbf{tabu}_k$} and where α and β are parameters that allow a user to control the relative importance of trail versus visibility. Therefore the transition probability is a trade-off between visibility (which says that close towns should be chosen with high probability, thus implementing a greedy constructive heuristics) and trail intensity (which says that if on edge (i,j) there has been a lot of traffic then it is highly desirable).

[1] A tour is completed in n time intervals.

[2] Even though the name chosen recalls tabu search, proposed in [«@Glover89»] and [«@Glover90»], there are substantial differences between our approach and tabu search algorithms. We mention here: (i) the absence of any aspiration function, (ii) the difference of the elements recorded in the tabu list, permutations in the case of tabu search, nodes in our case (our algorithms are constructive heuristics, which is not the case of tabu search).

The Ant-cycle algorithm is then

The Ant-cycle algorithm

1 Initialize:
- Set t:=0 {t is the time counter}
- For every edge (i,j) set an initial value $\tau_{ij}(t)$ and set $\Delta\tau_{ij}(t,t+n):= 0$
- Place $b_i(t)$ ants on every node i {$b_i(t)$ is the number of ants on node i at time t}
- Set s:=1 {s is the tabu list index}
- For i:=1 to n do
  - For k:=1 to $b_i(t)$ do
    - $\mathbf{tabu}_k(s):=i$ {starting town is the first element of the tabu list of the k-th ant}

2 Repeat until tabu list is full {this step will be repeated (n-1) times}
- 2.0 Set s:=s+1
- 2.1 For i:=1 to n do {for every town}
  - For k:=1 to $b_i(t)$ do {for every k-th ant on town i still not moved}
    - Choose the town j to move to, with probability $p_{ij}(t)$ given by equation (2)
    - Move the k-th ant to j {this instruction creates the new values $b_j(t+1)$}
    - Insert node j in $\mathbf{tabu}_k(s)$

3 For k:=1 to m do {for every ant}
- Compute L^k {it results from the tabu list}
- For s:=1 to n-1 do {scan the tabu list of the k-th ant}
  - Set (h,l):=($\mathbf{tabu}_k(s)$,$\mathbf{tabu}_k(s+1)$) {(h,l) is the edge connecting town s and s+1 in the tabu list of ant k}
  - $\Delta\tau_{hl}(t+n):=\Delta\tau_{hl}(t+n) + \dfrac{Q}{L^k}$

4 For every edge (i,j) compute $\tau_{ij}(t+n)$ according to equation (1)
Set t:=t+n
For every edge (i,j) set $\Delta\tau_{ij}(t,t+n):=0$

5 Memorize the shortest tour found up to now
If (NC < NC_{MAX}) or (not all the ants choose the same tour) {NC is the number of algorithm cycles; in NC cycles are tested NC·m tours}
- then
  - Empty all tabu lists
  - Set s:=1
  - For i:=1 to n do
    - For k:=1 to $b_i(t)$ do
      - $\mathbf{tabu}_k(s):=i$ {after a tour the k-th ant is again in the initial position}
  - Goto step 2
- else
  - Print shortest tour and Stop

The complexity of the Ant-cycle algorithm is $O(NC\cdot n^2\cdot m)$ if we stop the algorithm after NC cycles. In fact we have that:

Step 1 is $O(n^2+m)$
Step 2 is $O(n^2\cdot m)$

Step 3 is O(n·m)
Step 4 is O(n^2)
Step 5 is O(n·m)

3. Experimental testing

We implemented the algorithm and investigated its strengths and weaknesses by experimentation: we ran several simulations to collect statistical data for this purpose. These results are described in the next subsections, together with a brief comparison with alternative heuristics for the TSP.

3.1 Best values of the parameters of the algorithm

The parameters considered here are those that affect directly or indirectly[3] the computation of the probability in formula (2): α, β, ρ. The number m of ants has always been set equal to the number n of cities. We tested several values for each parameter, all the other being constant (the default value of the parameters was α=1, β=1, ρ=0.7; in each experiment only one of the values was changed). The values tested were: α∈{0, 0.5, 1, 2}, β∈{0.5, 1, 2, 5, 10, 20} and ρ∈{0.3, 0.5, 0.7, 0.9}. Preliminary results, obtained on small scale problems, have been presented in [7] and [6]; the tests reported here are based, where not otherwise stated, on the Oliver30 problem, a 30-cities problem described for example in [23], for which a tour of length 424.635 was found using genetic algorithms. The same result is also often obtained by Ant-cycle, which can also yield better outcomes. In order to allow the comparison with other approaches the tour lengths have been computed both as real numbers and as integers. (In this case distances between towns are integer numbers and are computed according to the standard code proposed in [21].) All the tests have been carried out for NC_{MAX} = 5000 cycles and were averaged over 10 trials.

Beside the tour length, we were interested also in investigating the *uni-path behavior*, i.e., the situation in which all the ants make the same tour: this would indicate that the system has ceased to explore new possibilities and therefore the best tour achieved so far will not be improved any more. With some parameters settings in fact we observed that, after several cycles, all the ants followed the same tour despite the stochastic nature of the algorithms: this was due to a much higher trail level on the edges composing that tour than on all the others. This high trail level makes the probability that an ant chooses an edge not belonging to the tour very low.

The average of 10 trials obtained testing different values of α, β and ρ are respectively presented in Figs. 1, 2 and 3. The obtained results show that α presents an optimal range around 1, β between 1 and 5 and ρ around 0.5.

Ant-cycle enters the uni-path behavior only for α≥2; in all the other cases we always observed an ongoing exploration of different alternatives, even after more than NC_{MAX}=5000 cycles.

The algorithm mainly uses the greedy heuristic to guide search in the early stages of computation, but as computation runs it can start exploiting the global information contained in the values τ_{ij} of trail. This explains the value ρ=0.5: the algorithm needs to have the possibility to forget part of the experience gained in the past in order to better exploit new incoming global information.

[3] Experiments have shown that quantity Q doesn't influence the algorithm performance.

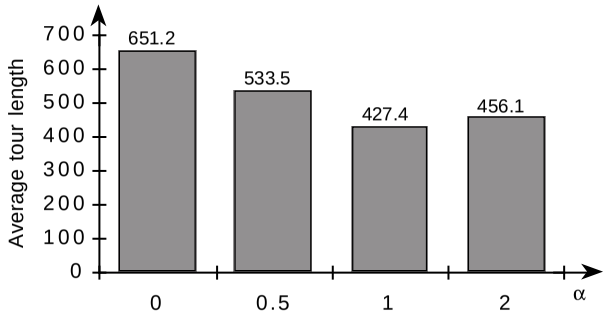

Fig.1 - Average tour length with increasing values of alpha

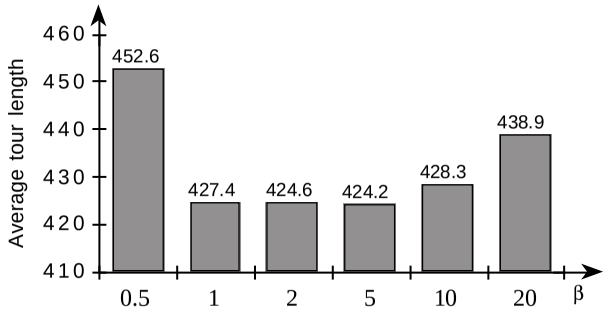

Fig.2 - Average tour length with increasing values of beta

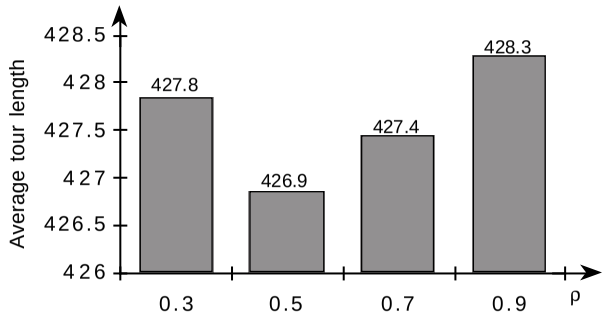

Fig.3 - Average tour length with increasing values of ro

The major strengths of the Ant-cycle algorithm can be summarized in the following points:

- with the best parameter values the algorithm always finds a very good solution, most of the times one that is better than the best one found using genetic algorithms;
- the algorithm finds good solutions very quickly; nevertheless it doesn't enter the uni-path behavior (in which all ants choose the best found tour), viz. the ants continue to search for new possibly better tours;

- we tested the Ant-cycle algorithm on problems with increasing dimensions and we found the sensitivity of the parameters optimal values to the problem dimension to be very low. Specifically, we let the algorithm run on the Eilon50 (see Fig.4) and Eilon75 problems [11] on a limited number of runs and with the number of cycles constrained to NC_{MAX}=3000. Under these restrictions we never got the best-known result, but a quick convergence to satisfying solutions was maintained for both the problems.

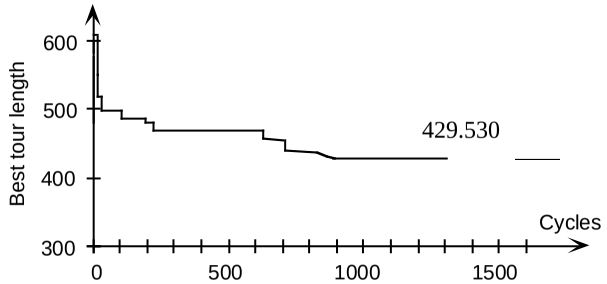

Fig.4 - The algorithm quickly finds very good values for the Eilon50 problem. In this figure cycles correspond to complete tours.

3.2 Best values of the parameters of the system

A set of experiments was run, in order to assess the impact of the number of ants on the efficiency of the solving process. In this case, the test problem involved finding a tour in a 5x5 grid of evenly spaced points: this is a problem with a priori known optimal solution (254.14 if we put to 10 the edge length, see Fig.5).

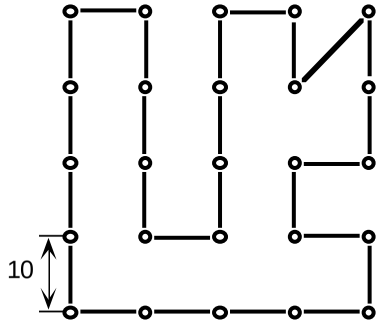

Fig.5 - An optimal solution for the 5x5 grid problem

In this case we determined the average number of cycles needed in each configuration to reach the optimum, if the optimum could be reached within 2000 cycles. The results are shown in Fig.6: on the abscissa there is the total number of ants used in each set of runs, on the ordinate the so called *one-ant cycles*, i.e., the number of cycles required to reach the optimum, multiplied by the number of ants used (in order to evaluate the efficiency per ant, hence the name, and to have comparable data). The algorithm has been able to identify the optimum with any number $m \geq 6$ of ants. It is interesting to note that:

- there is a synergetic effect in using more ants, up to an optimality point given by n=m; the existence of this optimality point is due to a tradeoff between the increased efficiency and the computational load caused by the management of progressively more ants. This causes the overall efficiency, measured in one-ant cycles, to decrease when the number of ants increases beneath the optimality point;

- tests on a set of r x r grid problems (r = 4, 5, 6, 7, 8) have shown that the optimal number of ants is close to the number of cities (n≈m): this property, combined with the assessment of the computational complexity developed in section 2, transforms the complexity of the algorithm used in our experiments from $O(NC \cdot n^2 m)$ into $O(NC \cdot n^3)$.

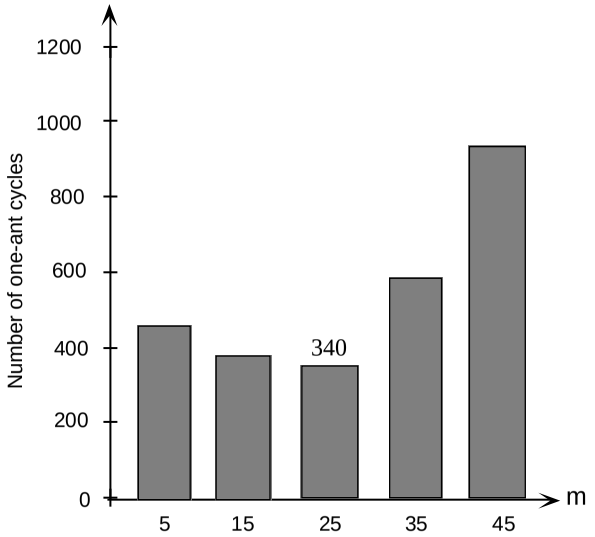

Fig.6 - Number of one-ant cycles required to reach optimum as a function of the total number m of ants, for the 5x5 grid problem

A second set of tests has been carried out with 25 cities randomly distributed (25 cities random graph). Again we found that the optimal performance was reached with 15÷25 ants, a number of ants comparable with the dimension (measured in number of cities) of the problem to be solved.

We tested whether there is any difference between the case in which all ants at time t=0 are in the same city and the case in which they are *uniformly* distributed[4]. We used Ant-cycle applied to the 25 cities random graph and the 5x5 grid previously described, and to the Oliver30 problem. In all cases uniformly distributing ants resulted in better performance.

In the case of the 25 cities random graph, we run 25 experiments (each one repeated five times) in which all the ants were positioned, at time t=0, on the same city (in the first experiment all ants were on town 1, in the second on town 2, and so on). We obtained that in all the cases the ants were able to identify the optimum, but they needed less cycles in the case of uniformly distributed ants.

Also in the case of the 5x5 grid problem experiments showed that the number of cycles needed to identify the optimum was lower in case of uniformly distributed ants.

In the case of the Oliver30 problem with 30 ants starting from the same city (runs were done using optimal parameters values), we noticed that the ants were never able to identify the

[4] We say ants are uniformly distributed if there is, at starting point, the same integer number of ants on every town (this excludes the possibility to have m which is not a multiple of n).

optimum (average value of the best tour found: 438.43), and that after some hundreds of cycles all the ants followed one of a very small set of tours.

We also tested whether an initial uniform distribution of the ants over the cities performed better than a random one; results show that there is little difference between the two choices, even though the random distribution obtained slightly better results.

3.3 Practical complexity

The algorithm complexity presented in sections 2 and 3.2, $O(NC \cdot n^3)$, doesn't say anything about the actual time required to reach the optimum. Results are reported in Table I for the case of similar problems with increasing dimensions (r x r grids with the edge length set to 10 as in Fig.5). It is interesting to note that, up to problems with 64 cities, Ant-cycle always found the optimal solution. Moreover both the number of cycles required to find it and the computational time actually used grow more slowly than the dimension of the search space, suggesting again that the algorithm uses a very effective search strategy.

Table I - Time required to find optimum as a function of problem dimension

Problem (n=r x r)	n^3	Best solution	Relative[5] dimension of search space	Average number of cycles to find optimum	Time required to find optimum[6] (seconds)
4 x 4	4096	160	1	5.6	$8 \cong 10$
5 x 5	15625	254.1	$\approx 10^{11}$	13.6	$75 \cong 10^2$
6 x 6	46656	360	$\approx 10^{28}$	60	$1020 \cong 10^3$
7 x 7	117649	494.1	$\approx 10^{49}$	320	$13440 \cong 10^4$
8 x 8	262144	640	$\approx 10^{75}$	970	$97000 \cong 10^5$

We compared the results of Ant-cycle with those obtained, on the same Oliver30 problem, by the other heuristics contained in the package "Travel" [4]. This package represents the distances among the cities as an integer matrix and so, in order to enable the comparison, we implemented an analogous representation in our system.

The results are shown in Table II, where in the first column there is the length of the best tour identified by each heuristic, in the second and third columns the improvement on the corresponding first column solution as obtained by the 2-opt (exhaustive exploration of all the permutations obtainable from the basic one by exchanging 2 cities) and the Lin-Kernighan heuristics [19], respectively.

Note how Ant-cycle consistently outperformed 2-opt, while its efficacy – i.e., the effectiveness it has in finding very good solutions – can be compared with that of Lin-Kernighan (even if our algorithm requires a longer computational time).

[5] The relative search space is given by $\frac{n!}{16!}$.

[6] Tests were run on a IBM-compatible PC.

Table II - Performance of Ant-cycle compared with other approaches

	basic[7]	2-opt	Lin-Kernighan[8]
Ant-cycle	420	-	-
Near Neighbour	587	437	420/421
Far Insert	428	421	420/421
Near Insert	510	492	420/421
Space Filling Curve	464	431	420/421
Sweep	486	426	420/421

As a general comment of all the tests, we like to point out that, given a good parameter setting (for instance α=1, β=2, ρ=0.5), our algorithm consistently finds a very good solution, the optimal one in case of BAYG29[9] or the new best known one in case of Oliver30, and finds rather quickly satisfying solutions (it usually identifies for Oliver30 the new best-known solution of length 423.74 in less than 400 cycles, and it takes only ≈100 cycles to reach values under 430). In any case exploration continues, as it is testified by the non-zero variance of the lengths of the tours followed by the ants in each cycle and by the fact that the average of the ants tour lengths gets never equal to the best tour found, but remains somewhat above it, thus indicating that tours around the best found are tested.

4. Conclusions

Results presented in the preceding sections suggest that the algorithm could be an effective optimization tool. The Ant system uses many simple interacting agents and a fast search algorithm based on positive feedback, without getting trapped in local minima; moreover augmenting the number of agents has a synergetic effect on the system performance, until an upper limit is reached.

A way to explain the effect of applying the algorithm to the TSP problem is to imagine to have some kind of probabilistic superimposition of effects: each ant, if isolated (i.e., if α=0), would move with a local, greedy rule. This greedy rule guarantees only locally optimal moves: it doesn't work because that greedy local improvements very often lead to very bad final steps (an ant is constrained to make a closed tour and therefore choices for the final steps are constrained by early steps). So the tour followed by an ant ruled by a greedy policy is composed by some parts that are very good and some others that are not. If we now consider the effect of the simultaneous presence of many ants, then each one contributes to a part of the trail distribution: good sets of paths will be followed by many ants and therefore they receive a great amount of trail, which in turn will make the path even more attractive (thus giving rise to a positive feedback – or *autocatalytic* – process - see also [10]); bad paths chosen only because obliged by constraints satisfaction (the tabu list) will be chosen only by few ants and therefore the trail over them remains low. When agents interact it looks like the greedy force can give the right suggestions to the autocatalytic process and let it converge on very good, often optimal, solutions very quickly, without getting stuck in local optima. Bad tours become highly improbable, and the algorithm searches only in the neighbourhood of good solutions.

We have seen that the ants cooperate by exchanging a particular kind of information, the trail, that is memorized in the problem structure. As no direct communication is necessary, and only local information is used to take decisions, the algorithm is very well suited to parallelization. We are currently designing the parallel version of Ant-cycle for a transputer architecture: we intend to give a set of ants to each transputer and to merge, every n steps, the trail left by each

[7] The name "basic" means the basic heuristic, with no improvement.

[8] The Lin-Kernighan algorithm found solutions of length 420 or 421 depending on the starting solution provided by the basic algorithm.

[9] This is a 29 cities problem proposed in TSPLIB 1.0 (cfr. note 4).

set of ants obtaining in this way a new trail matrix. We then redistribute this matrix to all the nodes.

With respect to the generality of our approach, we believe that many combinatorial problems can be faced by the Ant system. In order to apply the autocatalytic algorithm to another combinatorial problem, we must find an appropriate representation for:

1) the problem (to be represented as a graph searched by many simple agents),

2) the autocatalytic process,

3) the heuristic that allows a constructive definition of the solutions (the "greedy force"),

4) the constraint satisfaction method (viz. the tabu list).

This has been done for two well known combinatorial optimization problems – Quadratic Assignment (QAP) and Job-Shop Scheduling (JSP) – each time obtaining an adapted version of the Ant system that could effectively handle the corresponding problem.

Related work can be classified in the following major areas:

(i) studies of social animals behavior,

(ii) research in "natural algorithms",

(iii) stochastic optimization.

Research on behavior of social animals is to be considered as a source of inspiration and as a useful metaphor to explain our ideas. In particular, the work done on foraging behavior of real ants ([8], [9], [15]) has been our main source of inspiration. We believe that, especially if we are interested to design inherently parallel algorithms, observation of natural systems can be an invaluable source of inspiration. Neural networks [22], genetic algorithms [16], evolution strategies [20], immune networks [2], [3], simulated annealing [18] are only some of the proposed models with a "natural flavour". Main characteristics, at least partially shared by members of this class of algorithms, are the use of a natural metaphor, the inherent parallelism, the stochastic nature and adaptivity, the use of positive feedback, the capacity to learn (i.e., to improve performance on the basis of past experience). All this work in "natural optimization" can be inserted in the more general research area of stochastic optimization, in which the quest for optimality is traded for computational efficiency.

References

1. Aarts, E. and Korst, J. *Simulated Annealing and Boltzmann Machines*, John Wiley (1989).

2. Bersini, H. Hints for Adaptive Problem Solving Gleaned from Immune Networks. In *Proceedings First International Conference on Parallel Problem Solving from Nature (PPSN I)*, Schwefel, H.P. and Männer, R., Springer-Verlag, 1990.

3. Bersini, H. and Varela, F.J. The Immune Recruitment Mechanism: a Selective Evolutionary Strategy. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, UCSD - San Diego - CA, July 1991.

4. Boyd, S.C., Pulleyblank, W.R., and Cornuejols, G., *Travel*, Software Package - Carleton University, January 1989.

5. Clearwater, S.H., Hogg, T., and Huberman, B.A. Cooperative Problem Solving. Tech. Rept. Xerox Palo Alto Research Center, 1992.

6. Colorni, A., Dorigo, M., and Maniezzo, V. Positive Feedback as a Search Strategy. Tech. Rept. 91-16 Politecnico di Milano - Department of Electronics, Milano - Italy, November, 1991.

7. Colorni, A., Dorigo, M., and Maniezzo, V. Distributed Optimization by Ant Colonies. In *Proceedings First European Conference on Artificial Life*, MIT Press/Bradford Books, Paris, December 1991.

8. Denebourg, J.L., Pasteels, J.M., and Verhaeghe, J.C. Probabilistic Behaviour in Ants: a Strategy of Errors?. *Journal of Theoretical Biology 105*(1983).

9. Denebourg, J.L. and Goss, S. Collective Patterns and Decision-making. *Ethology, Ecology & Evolution 1*(1989).

10. Dorigo, M. *Optimization, Learning and Natural Algorithms*, Ph.D. dissertation, Politecnico di Milano - PM-AI & R Project, Milano - Italy, February 1992.

11. Eilon, S., Watson-Gandy, T.H., and Christofides, N. Distribution Management: mathematical modeling and practical analysis. *Operational Research Quarterly 20*(1969).

12. Glover, F. Tabu Search — Part I. *ORSA Journal on Computing 1*, 3 (1989).

13. Glover, F. Tabu Search — Part II. *ORSA Journal on Computing 2*, 1 (1990).

14. Goldberg, D.E. and Lingle, R. Alleles, Loci, and the Traveling Salesman Problem. In *Proceedings First International Conference on Genetic Algorithms*, Morgan Kaufmann, Carnegie-Mellon University - Pittsburg - PA, July 1985.

15. Goss, S., Beckers, R., Denebourg, J.L., Aron, S., and Pasteels, J.M. *Behavioural Mechanisms of Food Selection*, Springer-Verlag: Berlin, Vol. G20, NATO ASI (1990).

16. Holland, J.H. *Adaptation in Natural and Artificial Systems*, Ann Arbor: The University of Michigan Press (1975).

17. Hopfield, J.J. and Tank, D.W. Neura Computation of Decisions in Optimization Problems. *Biological Cybernetics 52*(1985).

18. Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P. Optimization by Simulated Annealing. *Science 220*(1983).

19. Lin, S. and Kernighan, B.W. An Effective Heuristic Algorithm for the TSP. *Operations Research 21*(1973).

20. Rechenberg, I. *Evolutionsstrategie*, Fromman-Holzbog, Stuttgart, Germany (1973).

21. Reinelt, G., *TSPLIB 1.0*, Institut für Mathematik, Universität Augsburg, Germany, 1990.

22. Rumelhart, D.E. and McLelland, J.L. *Parallel Distributed Processing: Explorations in the Microstructure of Cogniton*, MIT Press (1986).

23. Whitley, D., Starkweather, T., and Fuquay, D. Scheduling Problems and Travelling Salesmen: the Genetic Edge Recombination Operator. In *Proceedings Third International Conference on Genetic Algorithms*, Morgan Kaufmann, George Mason University, 1989.