Ant System: An Autocatalytic Optimizing Process Technical Report 91-016

Vittorio Maniezzo

Ant System: An Autocatalytic Optimizing Process

Technical Report 91-016

M. Dorigo , V. Maniezzo and A. Colorni
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Piazza Leonardo da Vinci 32
20133 Milano, Italy
dorigo@elet.polimi.it

Abstract

A combination of distributed computation, positive feedback and constructive greedy heuristic is proposed as a new approach to stochastic optimization and problem solving. Positive feedback accounts for rapid discovery of very good solutions, distributed computation avoids premature convergence, and greedy heuristic helps the procedure to find acceptable solutions in the early stages of the search process. An application of the proposed methodology to the classical travelling salesman problem shows that the system can rapidly provide very good, if not optimal, solutions. We report on many simulation results and discuss the working of the algorithm. Some hints about how this approach can be applied to a variety of optimization problems are also given.

1. Introduction

In this paper we explore the emergence of global properties from the interaction of many simple agents. In particular, we are interested in the distribution of search activities over so-called "ants", i.e., agents that use very simple basic actions in order to ease the parallelization of the computational effort. Our work has been inspired by researches on the behavior of real ants ([5],[6],[12]), where one of the problems of interest is to understand how almost blind animals like ants can manage to establish shortest route paths from their colony to feeding sources and back. It was found that the media used to communicate among individuals information regarding paths and used to decide where to go consists of *pheromone trails*. A moving ant lays some pheromone (in varying quantities) on the ground, thus marking the path it follows by a trail of this substance. While an isolated ant moves essentially at random, an ant encountering a previously laid trail can detect it and decide with high probability to follow it, thus reinforcing the trail with its own pheromone. The collective behavior that emerges is a form of *autocatalytic* behavior[1] where the more the ants are following a trail, the more attractive that trail becomes for being followed. The process is thus characterized by a positive feedback loop, where the probability with which an ant chooses a path increases with the number of ants that previously chose the same path.

The algorithms that we are going to define in the next sections are a model deriving from the study of artificial ant colonies and therefore we call our system *Ant system* and the algorithms we introduce *Ant algorithms*. As we are not interested in simulation of ant colonies, but in the use of artificial ant colonies as an optimization tool, our system will have some major differences with a real (natural) one: artificial ants will have some memory, they will not be completely blind and will live in a environment where time is discrete. Nevertheless we believe that the ant colony metaphor can be useful as a didactic tool to explain our ideas.

[1] An autocatalytic [7], i.e. positive feedback, process is a process that reinforces itself, in a way that causes very rapid convergence and, if no limitation mechanism exists, leads to explosion.

A result obtained running experiments with the Ant systems was the observation of the presence of synergetic effects in the interaction of ants. In fact, the quality of the solution obtained increases when the number of ants working on the problem increases, up to reach an optimal point (see section 5.2 for details).

A major point in defining any distributed system is the definition of the communication procedure. In our algorithms a set of ants communicate by modifications of the problem representation, since at any step during the problem solving each ant gives a sign of its activity that will change the probability with which the same decision will be taken in the future. The idea is that if at a given point an ant has to choose between different options and the one actually chosen results in being particularly good, then in the future that choice will appear more desirable than it was before. We also give ants a heuristic to guide the early stages of the computational process, when experience hasn't yet accumulated into the problem structure. This heuristic automatically loses importance as the experience gained by ants, and memorized in the problem representation, increases.

A second, but no less important, result presented in this paper is the viability of autocatalytic processes as a methodology for optimization and learning. A "single-ant" autocatalytic process usually converges very quickly to a bad suboptimal solution. Luckily, the interaction of many autocatalytic processes can lead to rapid convergence to a subspace of the solution space that contains many good solutions, causing the search activity to find quickly a very good solution, without getting stuck in it. In other words, all the ants converge not to a single solution, but to a subspace of solutions; thereafter they go on searching for improvements of the best found solution.

At the present stage of understanding we do not have any proof of convergence or bound on the time required to find the optimal solution. Nevertheless simulations strongly support the above speculations. We also believe our approach to be a very promising one because of its generality (it can be applied to many different problems) and because of its effectiveness in finding very good solutions to difficult problems.

The paper is organized as follows: section 2 contains the description of the 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. Sections 3 and 4 describe three slightly different ways to apply the proposed algorithm to the chosen problem. Section 5 reports diffusely on experiments carried out with the algorithms previously introduced. In section 6 we discuss the properties of the distributed algorithm and show how it could be applied to other optimization problems. Conclusions, related and further work are contained in section 7.

2. The Ant system

We introduce in this section our approach to the distributed solution of difficult problems by many locally interacting simple agents. We call *ants* the simple interacting agents and ant-algorithms the class of algorithms we define. We first describe the general characteristics of the ant-algorithms and then introduce three of them, called *Ant-density*, *Ant-quantity* and *Ant-cycle*.

To test the ant algorithms, we decided to apply them to the well-known travelling salesman problem (TSP) [15], to have a comparison with results obtained by other heuristic approaches [11]: the model definition is influenced by the problem structure, however we will hint in section 6 that the same approach can be used to solve other optimization problems. We stress that the choice of TSP is due to its ubiquity as a benchmark for heuristics: we are interested in

the proposal of a new heuristic and in its comparison with other ones, not directly in proposing – at least in this paper – 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).

Given a set of n towns, the TSP problem can be stated as the problem of finding a minimal length closed tour that visits each town once.

We call d_{ij} the length of the path between towns i and j; in the case of euclidean TSP d_{ij} is the Euclidean distance between i and j (i.e., $d_{ij}=[(x_i-x_j)^2 + (y_i-y_j)^2]^{1/2}$). An instance of the TSP problem is given by a weighted graph (N,E), where N is the set of towns and E is the set of edges between towns, weighted by the distances.

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.

Each ant is a simple agent with the following characteristics:

- when going from town i to town j it lays a substance, called *trail*, on edge (i,j);
- it chooses the town to go to with a probability that is a function of the town distance and of the amount of trail present on the connecting edge;
- to force ants to make legal tours, transitions to already visited towns are inhibited till a tour is completed (see the tabu list in the following).

Let $\tau_{ij}(t)$ be the *intensity of trail* on edge (i,j) at time t. At each iteration of the algorithm trail intensity becomes

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

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

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

$\Delta\tau_{ij}^{k}(t,t+1)$ 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+1.

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

In order to satisfy the constraint that an ant visits n different towns (a n-town tour), we associate to each ant a data structure, called *tabu list*[2], that memorizes the towns already visited up to time t and forbids the ant to visit them again before a tour has been completed. When a tour is completed the tabu list is emptied and the ant is free again to choose its way. We define

[2] Even though the name chosen recalls tabu search, proposed in [9] and [10], 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).

tabu$_k$ a vector containing the tabu list of the k-th ant, and **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

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

where **allowed** = {j: j$\in$ **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 heuristic) and trail intensity (that says that if on edge (i,j) there has been a lot of traffic then it is highly desirable, thus implementing the autocatalytic process).

Different choices about how to compute $\Delta\tau_{ij}^{k}(t,t+1)$ and when to update the $\tau_{ij}(t)$ cause different instantiations of the ant algorithm. In the next two sections we present the three algorithms we used as experimental test-bed for our ideas, namely Ant-density, Ant-quantity and Ant-cycle.

3. The Ant-density and Ant-quantity algorithms

In the Ant-density model a quantity Q_1 of trail for every unit of length is left on edge (i,j) every time an ant goes from i to j; in the Ant-quantity model an ant going from i to j leaves a quantity Q_2/d_{ij} of trail for every unit of length.

Therefore, in the Ant-density model

$$\Delta\tau_{ij}^{k}(t,t+1) = \begin{cases} Q_1 & \text{if k-th ant goes from i to j between t and t+1} \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

and in the Ant-quantity model we have

$$\Delta\tau_{ij}^{k}(t,t+1) = \begin{cases} \dfrac{Q_2}{d_{ij}} & \text{if k-th ant goes from i to j between t and t+1} \\ 0 & \text{otherwise} \end{cases} \qquad (4)$$

From these definitions it is clear that the increase in trail intensity on edge (i,j) when an ant goes from i to j is independent of d_{ij} in the Ant-density model, while it is inversely proportional to d_{ij} in the Ant-quantity model (i.e., shorter edges are made more desirable by ants in the Ant-quantity model, thus further reinforcing the visibility factor in equation (2)).

The Ant-density and Ant-quantity algorithms are then

1 Initialize
  Set t:=0 {t is the time counter}
  For every edge (i,j) set an initial value $\tau_{ij}(t)$ for trail intensity and $\Delta\tau_{ij}(t,t+1):= 0$
  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
    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 **tabu**$_k$(s)
  Set $\Delta\tau_{ij}(t,t+1):= \Delta\tau_{ij}(t,t+1) + Q_1$ in case of the Ant-density model or
    $\Delta\tau_{ij}(t,t+1):= \Delta\tau_{ij}(t,t+1) + Q_2/d_{ij}$ in case of the Ant-quantity model
  2.2 For every edge (i,j) compute $\tau_{ij}(t+1)$ according to equation (1)

3 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 cycles}
  then
    Empty all tabu lists
    Set s:=1
    For i:=1 to n do
      For k:=1 to $b_i(t)$ do
        tabu$_k$(s):=i {after a tour the k-th ant is again in the initial position}
    Set t:=t+1
    For every edge (i,j) set $\Delta\tau_{ij}(t,t+1):=0$
    Goto step 2
  else
    Print shortest tour and Stop

In words the algorithms work as follows. At time zero an initialization phase takes place during which ants are positioned on different towns and initial values for trail intensity are set on edges. The first element of each ant tabu list is set to be equal to the starting town. Thereafter every ant moves from town i to town j choosing the town to move to with a probability that is given as a function (with parameters α and β) of two desirability measures: the first (called trail - τ_{ij}) gives information about how many ants in the past have chosen that same edge (i,j), the second (called visibility - η_{ij}) says that the closer a town the more desirable it is (setting $\alpha = 0$ we obtain a stochastic greedy algorithm with multiple starting points, setting $\alpha = 0$ and $\beta \to \infty$ we obtain the deterministic classical one).

Each time an ant makes a move, the trail it leaves on edge (i,j) is summed to trail left on the same edge in the past. When every ant has moved, transition probabilities are computed using new trail values, according to formulae (1) and (2).

After n-1 moves the tabu list of each ant will be full: the shortest path found by the m ants is computed and memorized and all tabu lists are emptied. This process is iterated until the tour counter reaches the maximum (user-defined) number of cycles NC_{MAX} or all ants make the same tour (we call this last case *uni-path* behavior: it denotes a situation in which the algorithm stops searching for alternative solutions, see section 5.1).

By examination of the algorithms we see that the computational complexity expressed as a function of the number of ants m, the number of towns n and the number of cycles NC (where

a cycle is a complete tour) is $O(NC\cdot(m\cdot n^2+n^3))$. In fact, for both the Ant-quantity and Ant-density algorithms we have that:

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

Step 2 is $O(n\cdot m+n^3)$ (step 2.1 is O(m), step 2.2 is $O(n^2)$, and they are repeated n times),

Step 3 is $O(n\cdot m+n^2)$

In our experiments, see section 5, we found that the optimal number of ants is $m=c_1\cdot n$, where c_1 is a small constant (the experimental best value is $c_1=1$). Therefore the overall complexity is $O(NC\cdot n^3)$. NC could well be a function of n (NC=NC(n)). We have investigated the form of the NC function in section 5.7.

4. The Ant-cycle algorithm

In this case we introduced a major difference with respect to the two previous systems. Here $\Delta\tau_{ij}^k$ is not computed at every step, but after a complete tour (n steps). The value of $\Delta\tau_{ij}^k(t,t+n)$ is given by

$$\Delta\tau_{ij}^k(t,t+n)=\begin{cases}\dfrac{Q_3}{L^k} & \text{if k-th ant uses edge (i,j) in its tour}\\ 0 & \text{otherwise}\end{cases} \qquad (5)$$

where Q_3 is a constant and L^k is the tour length of the k-th ant. This corresponds to an adaptation of the Ant-quantity approach, where trails are updated at the end of a whole cycle instead of after each single move. We expect this algorithm to have a better performance than the previously defined Ant-density and Ant-quantity because here global information about the value of the result (i.e., the tour length) is used.

The value of the trail is also updated every n steps according to a formula very similar to (1)

$$\tau_{ij}(t+n) = \rho_1\cdot\tau_{ij}(t)+\Delta\tau_{ij}(t,t+n) \qquad (1')$$

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

and ρ_1 is different from ρ because the equation is no more updated at every step but only after a tour (n steps).

The Ant-cycle algorithm is then

```
1  Initialize:
     Set t:=0                                  {t is the time counter}
     For every edge (i,j) set an initial value τij(t) for trail intensity and Δτij(t,t+n):= 0
     Place bi(t) ants on every node i          {bi(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 bi(t) do
           tabuk(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, as results from its 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,s+1) in the tabu list of ant k}
$\Delta\tau_{hl}(t+n):=\Delta\tau_{hl}(t+n) + Q_3/L^k$
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 the number of cycles}
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 (remember NC could be NC(n)). In fact we have that:

Step 1 is $O(n^2+m)$
Step 2 is $O(n^2 \cdot m)$
Step 3 is $O(n \cdot m)$
Step 4 is $O(n^2)$
Step 5 is $O(n \cdot m)$

Also for the Ant-cycle algorithm we found a linear relation between the number of towns and the best number of ants. Therefore the complexity of the algorithm is $O(NC \cdot n^3)$.

5. Computational results

We implemented the three algorithms and investigated their relative strengths and weaknesses by experimentation. Since we have not yet developed a mathematical analysis of the models, which would yield the optimal parameter setting in each situation, we ran several simulations to collect statistical data for this purpose. During this phase we found Ant-cycle to be superior to the other two algorithms. We therefore tried to deepen our understanding of it by analyzing the effects of several variations of the basic algorithm, like defining a unique versus different starting points for every ant, increasing the importance of the ant that found the best tour or adding noise to the computation of the probabilities. These results are described in the next subsections, together with a brief comparison with alternative heuristics for the TSP.

5.1 Parameters setting

The parameters considered here are those that affect directly or indirectly the computation of the probability in formula (2): α, β, ρ, Q_h (h=1, 2, 3). 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, Q_h=100; in each experiment only one of the values was changed), over ten simulations for each setting in order to achieve some statistical information about the average evolution. The values tested were: $\alpha \in \{0, 0.5, 1, 5\}$, $\beta \in \{0, 1, 2, 5, 10, 20\}$, $\rho \in \{0.3, 0.5, 0.7, 0.9, 0.999\}$ and $Q_h \in \{1, 100, 10000\}$; α and β have been tested over different sets of values because of the different sensitivity the algorithms have shown for them. Preliminary results, obtained on small-scale problems, have been presented in [3] and [4]; the tests reported here are based, where not otherwise stated, on the Oliver30 problem, a 30-cities problem described in [13], for which a tour of length 424.635 was found using genetic algorithms[3]. The same result is also often obtained by the Ant system, which can also yield better outcomes. In order to allow the comparison with other approaches (see section 5.6) the tour lengths have been computed both as real numbers and as integers[4]. All the tests have been carried out for NC_{MAX} = 5000 cycles and were averaged over ten 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 three algorithms show a different sensitivity to the parameters.

Ant-density shows for β a monotonic decrease of the tour length up to β=10. After this value the average length starts to increase.

β	0	1	2	5	**10**	20
avg. length	881.56	456.98	455.52	431.37	426.74	428.79

The tests on the other parameters show that α has an optimum around 1

α	0	0.5	**1**	2	5
avg. length	578.52	464.99	456.98	508.44	695.25

that ρ should be set as high as possible

[3] Genetic algorithms [13] seem to be a very good general heuristic for combinatorial optimization problems, at least as good as simulated annealing [17].

[4] In this case distances between towns are integer numbers and are computed according to the standard code proposed in TSPLIB 1.0 (Reinelt G., TSPLIB 1.0, Institut für Mathematik, Universität Augsburg, 1990).

ρ	0.3	0.5	0.7	0.9	**0.999**
avg. length	649.86	530.58	456.98	431.31	429.09

and that the system is little affected by Q_1, never being able to improve significantly the quite unsatisfactory solution obtained in standard condition. The importance of the quantity Q_h (h=1,2,3) resulted to be uninfluential in all the three algorithms.

The experiments with Ant-density show that this system enters the uni-path behavior only for $\beta \geq 2$, usually within the first 200-300 cycles.

Ant-quantity shows a different sensitivity to the parameters; β is still very important, as shown in the following table

β	0	1	2	5	10	**20**	30
avg. length	454.72	441.85	436.77	431.60	428.52	427.95	438.83

where a decrease of the average tour length with increasing β is still exhibited up to β=20.

Also in this case we have that a too high or a too low α can worsen the performance, but a minor sensitivity to trails (α=0.5) could improve the result obtained with standard configuration (α=1).

α	0	**0.5**	1	5
avg. length	649.45	430.70	441.85	478.48

The tests on ρ show that keeping a strong memory of past experience is a good policy here, as it was in Ant-density, since again higher ρ yield better results.

ρ	0.3	0.5	0.7	0.9	**0.999**
avg. length	555.24	451.51	441.85	426.48	426.25

Ant-quantity is more prone to uni-path behavior than Ant-density, in fact we observed uni-path behavior for $\alpha \geq 1$ and for $\beta \geq 1$. In all the other cases the system kept on the exploration activity.

Results obtained with Ant-cycle show that α presents an optimal range around 1, β between 2 and 5 and ρ around 0.5.

The average of the tests are, for β:

β	0	0.5	1	2	**5**	10	20
avg. length	848.31	452.62	427.44	424.63	424.25	428.35	438.88

for α:

α	0	0.5	**1**	2
avg. length	651.27	533.49	427.44	456.11

and for ρ:

ρ	0.3	0.5	0.7	0.9
avg. length	427.85	426.86	427.44	428.28

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

In the three algorithms ρ is the only parameter that has qualitatively different behaviors. Its best value is as high as possible (i.e., very close to 1) in Ant-density and Ant-quantity, and ρ=0.5 in Ant-cycle. This fact can be explained as follows: Ant-density and Ant-quantity use only strictly local information, i.e., the visibility η_{ij} (local by definition) and τ_{ij} that in these cases is also local because doesn't contain any information about the final tour length of the ants that laid it in the past. The amount of trail on each edge is therefore a direct consequence of the local greedy rule. Conversely, in the Ant-cycle algorithm trail actually laid is a function of the total tour length (see formula (5)) and the algorithm uses therefore global information on the result of sequences of local moves. In other words what happens is that in the first two algorithms τ_{ij} is only a reinforcement of η_{ij}, while in Ant-cycle, as global information about total tour length is used to compute the amount of trail to deposit, τ_{ij} represents a different type of information with respect to η_{ij}.

All the three algorithms mainly use the greedy heuristic to guide search in the early stages of computation, but it is only Ant-cycle that as computation runs can start exploiting the global information contained in the values τ_{ij} of trail. This explains both the better performance of Ant-cycle and 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. In Ant-density and Ant-quantity this forgetting capability is neither necessary nor useful because there is no "second stage" in the search process in which global information can be exploited: the two algorithm always continue to use the same strategy based on the greedy heuristic.

We investigated also the behavior of the Ant-cycle algorithm for different combination of parameters α and β (in this experiment every run was stopped after 2500 cycles, i.e., after every ant had made 2500 tours). The results are summarized in Fig.1. We identified three different zones: for high values of α and not too high values of β the algorithm enters very quickly the uni-path behavior without finding very good solutions (this situation is represented by the symbol ■ in Fig.1); if we don't give enough importance to trail (i.e., we set α to a low value) or we give too a high importance to the greedy rule (high values to β) then the algorithm doesn't find very good solutions in the number of cycles used in the experiment (the symbol used for this situation is ∞). Very good solutions are found for α and β values in the central area (where the symbol used is ✖). In this case we found that different parameter combinations (i.e., (α=1, β=1),(α=1, β=2),(α=1, β=5),(α=0.5, β=5)) resulted in the same performance level: same result (the shortest tour known on the Oliver30 problem) in approximately the same number of cycles.

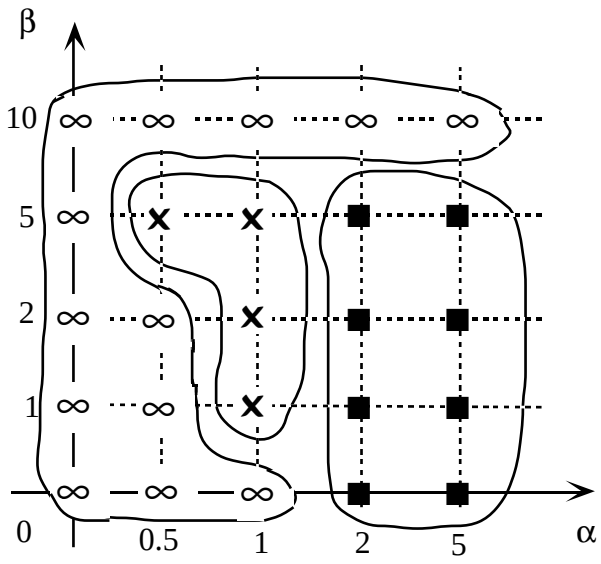

Fig.1 - Ant-cycle behavior for different combinations of α-β parameters
- ✕ - the algorithm finds very good solutions without entering the uni-path behavior
- ∞ - the algorithm doesn't find very good solutions without entering the uni-path behavior
- ■ - the algorithm doesn't find very good solutions and enters the uni-path behavior

The results obtained in this experiment are consistent with our understanding of the algorithm: a high value for α means that trail is very important and therefore ants tend to choose edges chosen by other ants in the past. This is true until the value of β becomes very high: in this case even if there is a high amount of trail on a edge, an ant always has a high probability of choosing another town that is very near.

High values of β and/or low values of α make the algorithm very similar to a stochastic multigreedy algorithm.

In Fig.2 we present the new optimal tour[5] found using the - experimentally determined - optimal set of parameters values for the Ant-cycle algorithm (α=1, β=2, ρ=0.5, Q_3=100). This tour results to be of length 423.74 and presents two inversions with respect to the best tour published in [20].

[5] This result is not exceptional since big dimensional TSP problems are now solved to optimality with special-purpose algorithms. Nevertheless, it is interesting to note that our algorithm can consistently find very good solutions to published problems (see also section 5.7)

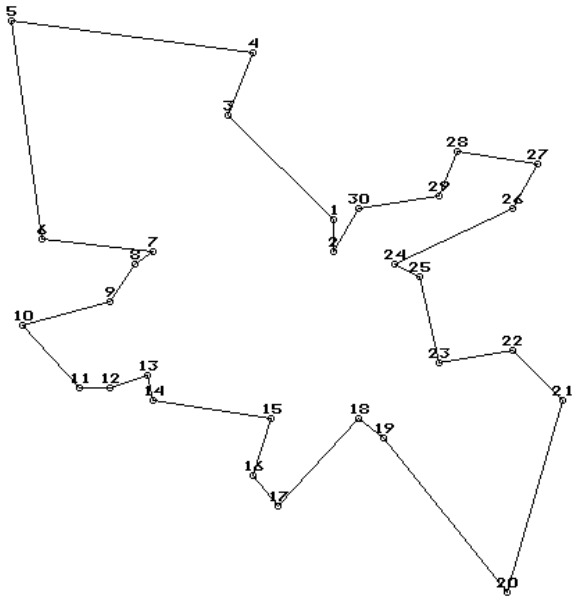

Fig.2 - The new best found tour[6] obtained with 342 iterations of Ant-cycle for the Oliver30 problem (α=1, β=2, ρ=0.5, Q_3=100), real length = 423.74, integer length = 420 (cfr. note 4).

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

- within the range of parameter optimality the algorithm always finds a very good solution, most of the times one that is better than the best found using genetic algorithms;
- the algorithm finds good solutions very quickly (see Fig.3); 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 to problems with increasing dimensions and we found the sensitivity of the parameters optimal values to the problem dimension to be very low.

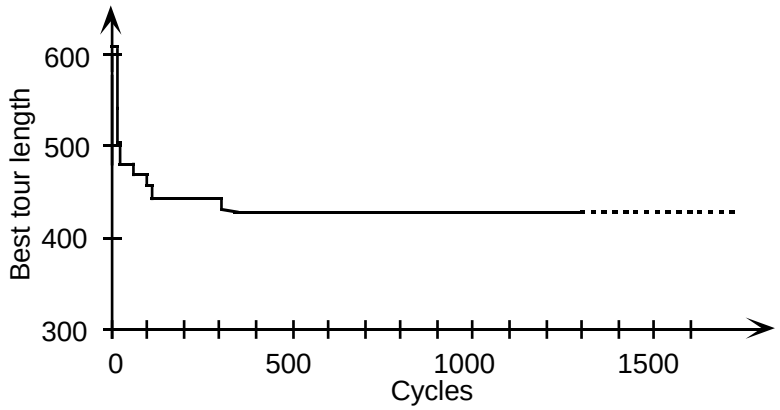

Fig.3 - The algorithm finds very good values for Oliver30 very quickly and the new optimal value (423.74) after 342 cycles. In this figure cycles correspond to complete tours.

On the Oliver30 problem Ant-cycle reached the former best-known result [20] with a frequency statistically higher than that of the other two algorithms and, on the whole, identification of good tours is much quicker; moreover, it is the only one which can identify the new optimal tour.

[6] The previous best found tour, published in [20], has real length of 424.63 and integer length of 421.

We partially tested the algorithm on the Eilon50 and Eilon75 problems [8] 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.

5.2 Number of ants

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 4x4 grid of evenly spaced points: this is a problem with a priori known optimal solution (160 if we put to 10 the edge length, see Fig.4).

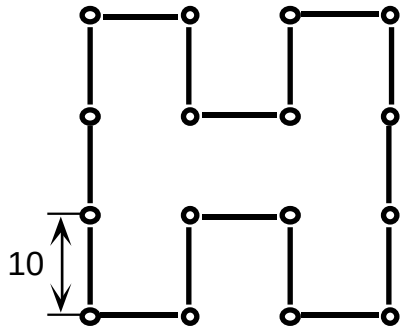

Fig.4 - An optimal solution for the 4x4 grid problem

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. As already said, this test, as the following ones, was conducted only for the Ant-cycle algorithm. The results are shown in Fig.5: 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 average 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). It is interesting to note that:

- the algorithm has been consistently able to identify the optimum with any number $m \geq 4$ of ants;
- 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 the computational load caused by the management of progressively more ants that causes the overall efficiency, measured in one-ant cycles, to decrease when increasing number of ants;
- 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 was used in the assessment of the computational complexity (see sections 3 and 4).

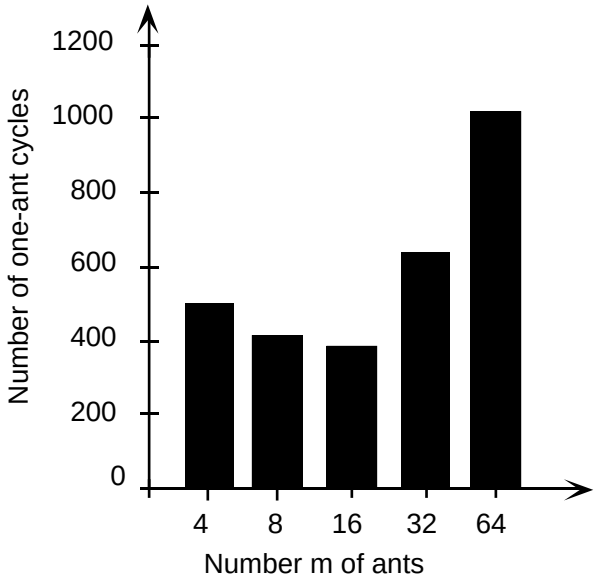

Fig.5 - Number of one-ant cycles required to reach optimum as a function of the total number of ants for the 4x4 grid problem

A second set of tests has been carried out with 16 cities randomly distributed (16 cities random graph). Again we found that the optimal performance was reached with 8-16 ants, a number of ants comparable with the dimension (measured in number of cities) of the problem to be solved.

5.3 Which town should ants start from?

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[7]. We used Ant-cycle applied to the 16 cities random graph and the 4x4 grid described in the previous subsection, and to the Oliver30 problem. In all cases uniformly distributing ants resulted in better performance.

In the case of the 16 cities random graph, we run 16 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 an average of 64.4 cycles vs. the average 57.1 needed in case of uniformly distributed ants.

In the case of the 4x4 grid problem experiments showed that an average of 28.2 cycles were needed to identify the optimum vs. 26.9 in case of uniformly distributed ants (in order to compare this data with those of Fig.5, the number of cycles must be multiplied by 16 to obtain the number of one-ant cycles).

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 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.

[7] 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).

5.4 Elitist strategy

We call elitist strategy (because in some way it resembles the elitist strategy used in genetic algorithms, reproducing with probability 1 the best individual found) the modified algorithm in which at every cycle the trail laid on the edges belonging to the so far best found tour is reinforced by a quantity $e \cdot Q_3/L^*$, where e is the number of elitist ants[8] and L^* is the length of the best found tour. The idea is that the trail of the best tour so far identified (L^*), so reinforced, will direct in probability the search of all the other ants towards the edges that compose it.

The test were carried out again on the Oliver30 problem (the run was stopped after NC_{MAX} = 2500 cycles) and indicated that there is an optimal range for the number of elitist ants: below it, increasing their number results in better tours discovered and/or in the best tour being discovered earlier, above it, the elitist ants force the exploration around suboptimal tours in the early phases of the search, so that a decrease of performance results. Fig.6 shows the outcome of a test where this behavior is evident.

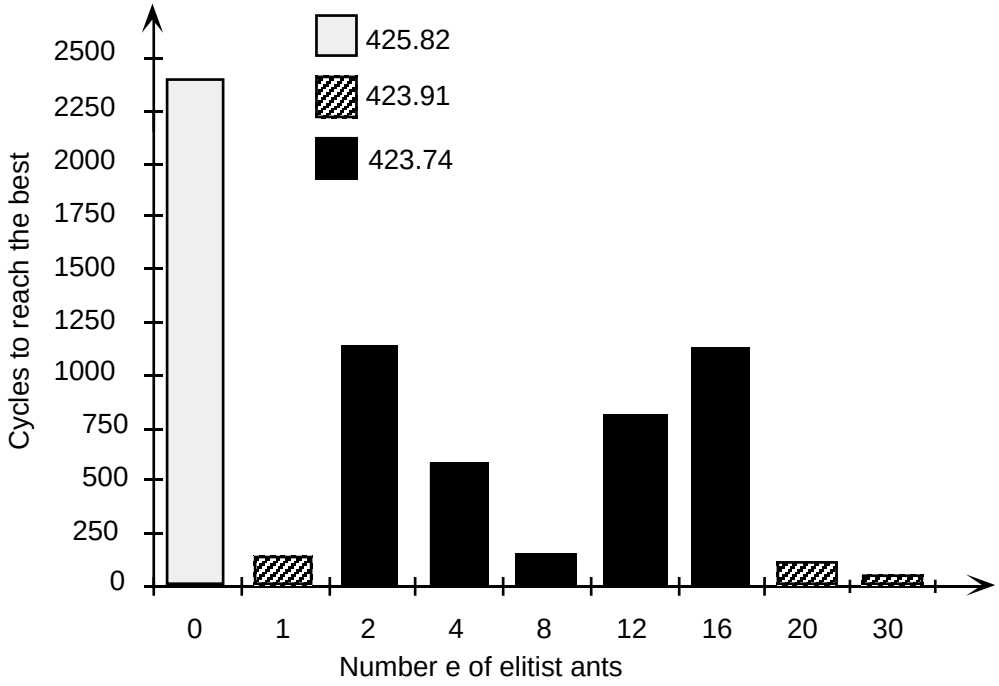

Fig.6 - Number of cycles required to reach a local optimum related to the number of elitist ants used

5.5 Noisy transition probability

We used in some experiments a slightly different transition rule including noise, given by the following formula:

[8] In our case the effect of an individual, i.e. an ant with its tabu list defining a tour, is to increment the value of the trail on edges belonging to its tour; therefore the equivalent of saving an individual is in our case to sum its contribution to the contribution of all other ants in the following cycle.

$$p_{ij}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)\cdot(1+\varepsilon_{ij}(\sigma))]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum_{j\in \mathbf{allowed}} [\tau_{ij}(t)\cdot(1+\varepsilon_{ij}(\sigma))]^{\alpha}\cdot[\eta_{ij}]^{\beta}} & \text{if } j\in \mathbf{allowed} \\ 0 & \text{otherwise} \end{cases} \qquad (2')$$

where **allowed** = {j: j∈ **tabu**$_k$} and $\varepsilon_{ij}(\sigma)$ is a noise function (random variable with zero mean and standard deviation σ).

This set of tests was carried out in order to assess the usefulness of formula (2') compared to formula (2), to calculate transition probabilities. The original idea was to evaluate the robustness of the procedure with respect to the intensity level of the trail, and to see whether early convergence could be avoided. The second objective turned out to be intrinsically met by the Ant-cycle algorithm, which naturally tend not to present the uni-path behavior. As test problem we used the 4x4 grid problem and we noticed that low noise values do not hamper the identification of the optimum – although they do not help either – while higher values lead to a significant worsening of the performance. In Fig.7 we display the average number of cycles, over 10 tests for each value, needed to reach the optimum with different values of noise.

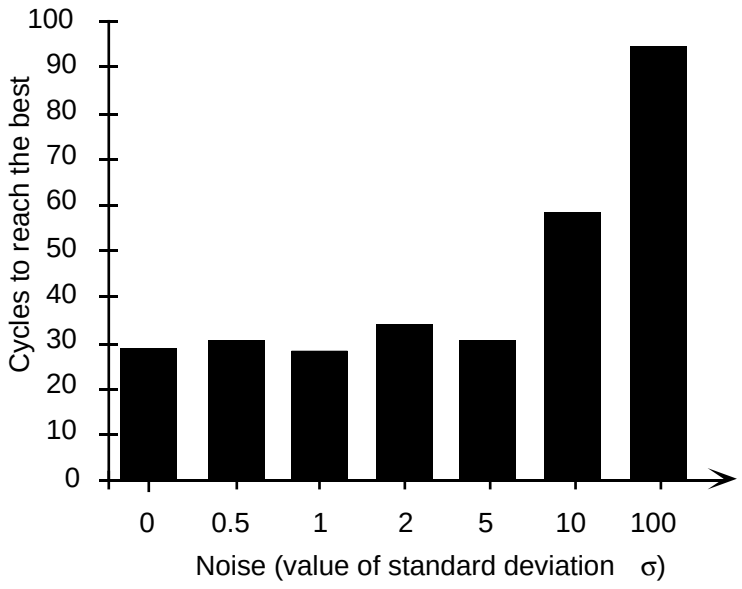

Fig.7 - Number of cycles required to reach optimum related to the different values of noise

5.6 Time required to find optimal solutions

The algorithm complexity presented in section 4, $O(NC\cdot n^3)$, doesn't say anything about the actual time required to reach the optimum. The experiment presented in this section is devoted to investigate the relation between NC and n, i.e., NC=NC(n). 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.4). 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	Best solution	Relative dimension of search space	Average number of cycles to find optimum	Time required to find optimum[9] (seconds)
4 x 4	160	1	5.6	8
5 x 5	254.1	$\approx 10^{11}$	13.6	75
6 x 6	360	$\approx 10^{28}$	60	1020
7 x 7	494.1	$\approx 10^{49}$	320	13440
8 x 8	640	$\approx 10^{75}$	970	97000

Observing Table I we note that the time required to find optimum is proportional to the the number of cycles necessary to find it multiplied by n^3 (in this case n = 16, 25, 36, 49, 64) confirming in this way the complexity analysis presented in section 4.

5.7 Comparison with other approaches

We compared the results of Ant-cycle with those obtained, on the same Oliver30 problem, by the other heuristics contained in the package "Travel" [2]. 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 [16], 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).

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

	basic[10]	2-opt	Lin-Kernighan[11]
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
Random	1212	663	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, Q_3=100, e=5), our algorithm consistently finds a very

[9] Tests were run on a IBM-compatible PC.
[10] The name "basic" means the basic heuristic, with no improvement.
[11] The Lin-Kernighan algorithm found solutions of length 420 or 421 depending on the starting solution provided by the basic algorithm.

good solution, the optimal one in case of BAYG29[12] 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.

6. Discussion

Results presented in the preceding section suggest that the algorithm could be an effective optimization tool. In this section we try to give some insights about the way the algorithm works and to show how it could be applied to other NP-hard combinatorial problems.

A first way to explain the effect of applying the algorithm to the TSP problem is the following.

Consider the transition matrix **p**(t): every element $p_{ij}(t)$ is the transition probability from town i to town j at time t as defined by equation (2). At time t=0 each $p_{ij}(0)$ is proportional to η_{ij}, i.e., closer towns are chosen with higher probability. As the process evolves, **p**(t) changes its elements according to (1) and (2). The process can therefore be seen as a space deformation, in which distance is reduced between towns which are connected by edges with a high amount of traffic, and, conversely, distance is incremented between towns connected by edges with low traffic levels. From simulations we observed that the matrix **p**(t), at least in the range of optimality for our parameters, converges to a state[13] that is very close to stationary (i.e., variations in the transition matrix **p**(t) are very small). When this state is reached the behavior of the ants is dependent on the kind of transition matrix obtained. We observed two situations: in the most rare one, occurring - as we saw in section 5 - only for particular parameter settings, only one transition probability is significantly higher than zero in every row and therefore all the ants choose the same edge at each step and no new tour is searched. In the most common situations instead, most of the rows have two (sometimes more) transition probabilities with a value higher than zero. In these cases search never stops, even if the dimension of the space searched is highly reduced, with respect to the initial situation. Consider for example Fig.8, obtained as steady-state transition matrix for a randomly generated 10-town problem, where the area of each circle is proportional to the corresponding value of the transition probability. An ant in town 1 has a very high probability to go either to town 5 (near 50%) or to town 2 (near 35%), and a low probability of choosing any other edge (a similar analysis holds for ants in any other town; from town 9 and 0, for example, any destination is equally probable).

[12] This is a 29 cities problem proposed in TSPLIB 1.0 (cfr. note 4).
[13] The stochastic process that rules the evolution of the matrix **p**(t) is a Markov process with infinite memory.

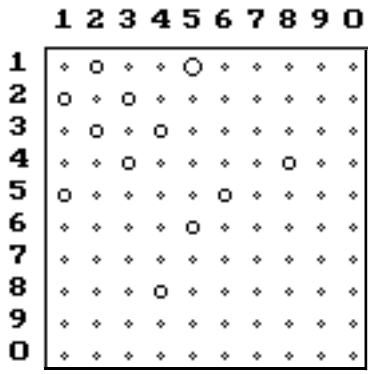

Fig.8 - Steady-state transition matrix for a randomly generated 10-town problem (Ant-cycle)

Another way to interpret how the algorithm works 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 and will practically always lead to bad final results. The reason the greedy rule doesn't work is that greedy local improvements 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; bad paths chosen only because obliged by constraints satisfaction (remember the tabu list) will be chosen only by few ants and therefore the trail over them remains low.

We have seen that the ants cooperate by exchanging a particular kind of information, 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 set of ants obtaining in this way a new trail matrix. We then redistribute this matrix to all the nodes.

Let's now consider 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 three well-known combinatorial optimization problems – Satisfiability (SAT), Quadratic Assignment (QAP) and Job-Shop Scheduling (JSP) – each time obtaining an adapted version of the Ant-system that could effectively handle the relative problem. The most difficult (and *ad hoc*) tasks to face when trying to apply the Ant-system are to find an appropriate graph representation for the problem to be solved and a greedy force as heuristic. This required the introduction of many edges between each pair of nodes in the case of QAP, of one more node in the case of JSP, of suitable constraints between each pair of nodes in the case of SAT.

7. Conclusions, related work and future investigations

This paper introduces a new search methodology based on a *distributed autocatalytic process* and its application to the solution of a classical optimization problem. Our main contributions are:

(i) we introduce positive feedback as a powerful search and optimization tool
(ii) we show how synergetic effects can arise in distributed systems.

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).

We reported many simulation results that illustrate the power of the approach. Moreover we presented an example of how the system can be applied to other optimization problems: we believe the approach can be extended to a broader class of problems.

The general idea underlying this model is that of a population of agents each one guided by an autocatalytic process pressed by a greedy force. Were an agent alone, both the autocatalytic process and the greedy force would tend to make the agent converge to a suboptimal tour with exponential speed. 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. We have speculated that this behavior can be due to the fact that information gained by agents during the search process is used to modify the problem representation and in this way to reduce the dimension of the space considered by the search process. Even if no tour becomes unfeasible, bad tours become highly improbable, and the algorithms search only in the neighbourhood of good solutions.

Related work can be classified in the following major areas:

(i) studies of social animals behavior;
(ii) research in "natural algorithms";
(iii) stochastic optimization.

As already pointed out the research on behavior of social animals is to be considered as a source of inspiration and as a useful metaphor to explain our ideas. 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 [19], genetic algorithms [13], evolution strategies [18], immune networks [1], simulated annealing [14] 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 adaptativity, the use of positive feedback, the capacity to learn (i.e., to improve performance on the basis of past experience). Our algorithm can then be well considered to be a new member of this class. 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.

We believe that further work can be done along the following main research directions:

- theoretical investigation of the proposed model (properties of convergence, complexity, ...);
- evaluation of the generality of the approach, through an investigation on which classes of problems can be solved by this algorithm;
- evaluation of the scalability of the approach by testing it on bigger problems using a parallel computer;

- study of the implications that our model can have on artificial intelligence, particularly in the pattern recognition and machine learning fields.

Acknowledgments

We would like to thank Thomas Bäck, Hughes Bersini, Frank Hoffmeister, Mauro Leoncini, Francesco Maffioli, Bernard Manderik, Giovanni Manzini, Daniele Montanari, Hans-Paul Schwefel and Frank Smieja for useful comments on a early version of this paper.

References

[1] H.Bersini, F.J.Varela, "The Immune Recruitment Mechanism: a Selective Evolutionary Strategy," *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, 1991.

[2] S.C.Boyd, W.R.Pulleyblank, G.Cornuejols, "Travel," Software Package, Carleton University, 1989.

[3] A.Colorni, M.Dorigo, V.Maniezzo "Distributed Optimization by Ant Colonies," *Proc. of the First European Conference on Artificial Life*, Paris, France, December 11-13, 1991.

[4] A.Colorni, M.Dorigo, V.Maniezzo, "Positive feedback as a search strategy," Technical Report n. 91-016 Politecnico di Milano, 1991.

[5] J.L.Denebourg, J.M.Pasteels, J.C.Verhaeghe, "Probabilistic Behaviour in Ants: a Strategy of Errors?," *J. Theor. Biol.*, vol.105, pp. 259-271, 1983.

[6] J.L.Denebourg, S.Goss, "Collective patterns and decision-making," *Ethology, Ecology & Evolution*, vol.1, pp. 295-311, 1989.

[7] M.Dorigo, *Optimization, Learning and Natural Algorithms*, Ph.D. Thesis, Politecnico di Milano, in press.

[8] S.Eilon, T.H.Watson-Gandy, N.Christofides, "Distribution management: mathematical modeling and practical analysis," *Operational Research Quarterly*, vol.20, pp.37-53, 1969.

[9] F.Glover, "Tabu Search — Part I," *ORSA Journal on Computing*, vol.1, no.3, pp.190-206, 1989.

[10] F.Glover, "Tabu Search — Part II," *ORSA Journal on Computing*, vol.2, no.1, pp.4-32, 1990.

[11] B.Golden, W.Stewart, "Empiric analysis of heuristics," in *The Travelling Salesman Problem*, E.L.Lawler, J.K.Lenstra, A.H.G.Rinnooy-Kan, D.B.Shmoys eds., New York:Wiley, 1985.

[12] S.Goss, R.Beckers, J.L.Denebourg, S.Aron, J.M.Pasteels, "How Trail Laying and Trail Following Can Solve Foraging Problems for Ant Colonies," in *Behavioural Mechanisms of Food Selection*, R.N.Hughes ed., NATO ASI Series, Vol. G 20, Berlin:Springer-Verlag, 1990.

[13] J.H.Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: The University of Michigan Press, 1975

[14] S.Kirkpatrick, C.D.Gelatt, M.P.Vecchi, "Optimization by Simulated Annealing" *Science* , vol.220, pp.671-680, 1983.

[15] E.L.Lawler, J.K.Lenstra, A.H.G.Rinnooy-Kan, D.B.Shmoys eds., *The Travelling Salesman Problem*, New York:Wiley, 1985.

[16] S.Lin, B.W.Kernighan, "An effective Heuristic Algorithm for the TSP," *Operations Research*, vol.21, pp.498-516, 1973.

[17] C.Peterson, "Parallel Distributed Approaches to Combinatorial Optimization: Benchmark Studies on Traveling Salesman Problem," *Neural Computation*, vol.2, pp.261-269, 1990.

[18] I.Rechenberg, *Evolutionsstrategie*, Fromman-Holzbog, Stuttgart, Germany, 1973.

[19] D.E.Rumelhart, J.L.McLelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cogniton*, Cambridge, MA: MIT Press, 1986

[20] D.Whitley, T.Starkweather, D.Fuquay, "Scheduling Problems and Travelling Salesmen: the Genetic Edge Recombination Operator," *Proc. of the Third Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, 1989.