Submitted to IEEE Transactions on Systems, Man, and Cybernetics

The Ant System: Optimization by a colony of cooperating agents

Marco Dorigo, Vittorio Maniezzo, Alberto Colorni
Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
e-mail: dorigo@elet.polimi.it
maniezzo@elet.polimi.it
colorni@elet.polimi.it

Abstract

An analogy with the way ant colonies function has suggested the definition of a new computational paradigm, which we call *Ant System*. We propose it as a viable new approach to stochastic combinatorial optimization. The main characteristics of this model are positive feedback, distributed computation, and the use of a constructive greedy heuristic. Positive feedback accounts for rapid discovery of good solutions, distributed computation avoids premature convergence, and the greedy heuristic helps find acceptable solutions in the early stages of the search process. We apply the proposed methodology to the classical Traveling Salesman Problem (TSP), and report simulation results. We also discuss parameter selection and the early setups of the model, and compare it with tabu search and simulated annealing using TSP. To demonstrate the robustness of the approach, we show how the Ant System (AS) can be applied to other optimization problems like the asymmetric traveling salesman, the quadratic assignment and job-shop scheduling. Finally we discuss the salient characteristics – global data structure revision, distributed communication and probabilistic transitions of the AS.

I. Introduction

In this paper we define a new general-purpose heuristic algorithm which can be used to solve different combinatorial optimization problems. The new heuristic has the following desirable characteristics:

- It is *versatile*, in that it can be applied to similar versions of the same problem; for example, there is a straightforward extension from the traveling salesman problem (TSP) to the asymmetric traveling salesman problem (ATSP).
- It is *robust*. It can be applied with only minimal changes to other combinatorial optimization problems such as the quadratic assignment problem (QAP) and the job-shop scheduling problem (JSP).

- It is a *population based* approach. This is interesting because it allows the exploitation of positive feedback as a search mechanism, as explained later in the paper. It also makes the system amenable to parallel implementations (though this is not considered in this paper).

These desirable properties are counterbalanced by the fact that, for some applications, the Ant System can be outperformed by more specialized algorithms. This is a problem shared by other popular approaches like simulated annealing (SA), and tabu search (TS), with which we compare the Ant System. Nevertheless, we believe that, as is the case with SA and TS, our approach is meaningful in view of applications to problems which, although very similar to well known and studied basic problems, present peculiarities which make the application of the standard best-performing algorithm impossible. This is the case, for example, with the ATSP.

In the approach discussed in this paper we distribute the search activities over so-called "*ants*," that is, agents with very simple basic capabilities which, to some extent, mimic the behavior of real ants. In fact, research on the behavior of real ants has greatly inspired our work (see [10], [11], [21]). One of the problems studied by ethologists was to understand how almost blind animals like ants could manage to establish shortest route paths from their colony to feeding sources and back. It was found that the medium used to communicate information among individuals 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 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 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.

Consider for example the experimental setting shown in Fig. 1. There is a path along which ants are walking (for example from food source A to the nest E, and vice versa, see Fig. 1a). Suddenly an obstacle appears and the path is cut off. So at position B the ants walking from A to E (or at position D those walking in the opposite direction) have to decide whether to turn right or left (Fig. 1b). The choice is influenced by the intensity of the pheromone trails left by preceding ants. A higher level of pheromone on the right path gives an ant a stronger stimulus and thus a higher probability to turn right. The first ant reaching point B (or D) has the same probability to turn right or left (as there was no previous pheromone on the two alternative paths). Because path BCD is

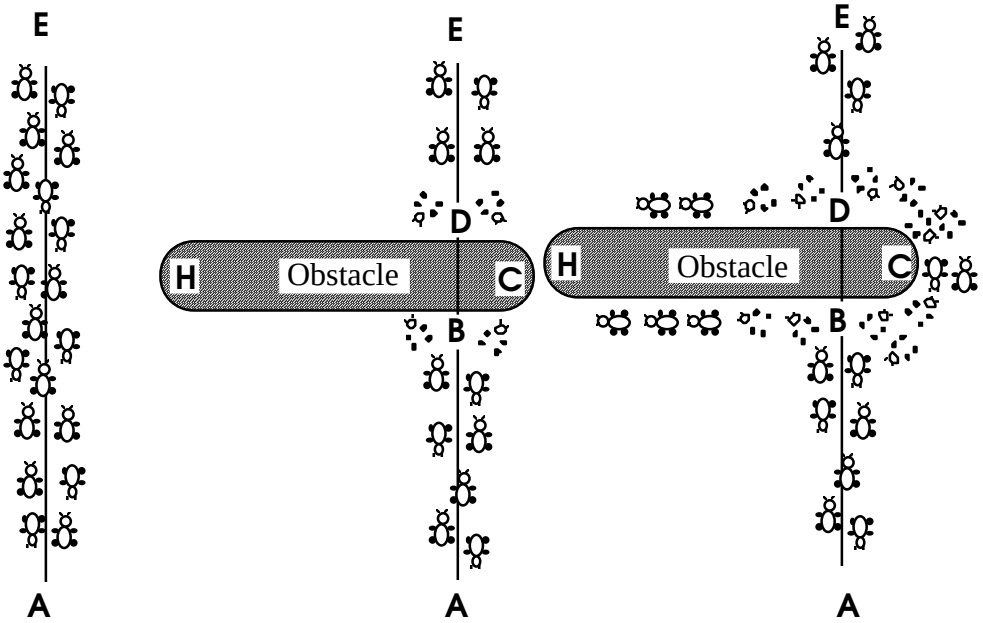
---

[1] An autocatalytic [12], 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.

shorter than BHD, the first ant following it will reach D before the first ant following path BHD (Fig. 1c). The result is that an ant returning from E to D will find a stronger trail on path DCB, caused by the half of all the ants that by chance decided to approach the obstacle via DCBA and by the already arrived ones coming via BCD: they will therefore prefer (in probability) path DCB to path DHB. As a consequence, the number of ants following path BCD per unit of time will be higher than the number of ants following BHD. This causes the quantity of pheromone on the shorter path to grow faster than on the longer one, and therefore the probability with which any single ant chooses the path to follow is quickly biased towards the shorter one. The final result is that very quickly all ants will choose the shorter path.

The algorithms that we are going to define in the next sections are models derived from the study of artificial ant colonies. Therefore we call our system *Ant System* (AS) 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,
- they will live in an environment where time is discrete.



a) b) c)

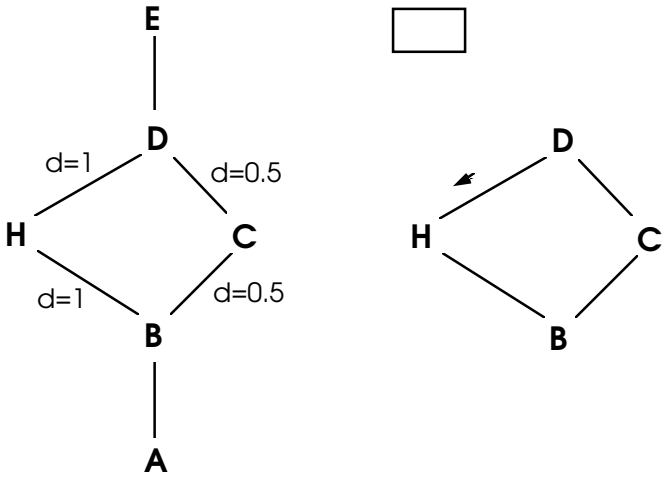
Fig. 1. a) Ants follow a path between points A and E.
b) An obstacle is interposed; ants can choose to go around it following one of the two different paths with equal probability.
c) On the shorter path more pheromone is laid down.

Nevertheless, we believe that the ant colony metaphor can be useful to explain our model. Consider the graph of Fig. 2a, which is a possible AS interpretation of the situation of Fig. 1b. To fix the ideas, suppose that the distances between D and H, between B and H, and between B and D—via C—are equal to 1, and let C be positioned half the way between D and B (see Fig. 2a). Now let us consider what happens at regular discretized intervals of time: t=0, 1, 2, ... Suppose that 30 new ants come to B from A, and 30 to D from E at each time unit, that each ant walks at a speed of 1 per time unit, and that while walking an ant lays down at time t a pheromone trail of intensity 1, which, to make the example simpler, evaporates completely and instantaneously in the middle of the successive time interval (t+1, t+2).

At t=0 there is no trail yet, but 30 ants are in B and 30 in D. Their choice about which way to go is completely random. Therefore, on the average 15 ants from each node will go toward H and 15 toward C (Fig. 2b).

At t=1 the 30 new ants that come to B from A find a trail of intensity 15 on the path that leads to H, laid by the 15 ants that went that way from B, and a trail of intensity 30 on the path to C, obtained as the sum of the trail laid by the 15 ants that went that way from B and by the 15 ants that reached B coming from D via C (Fig. 2c). The probability of choosing a path is therefore biased, so that the expected number of ants going toward C will be the double of those going toward H: 20 versus 10 respectively. The same is true for the new 30 ants in D which came from E.

This process continues until all of the ants will eventually choose the shortest path.


E
D
d=1
d=0.5
H
C
d=1
d=0.5
B
A
D
H
C
B

Given a set of n towns, the TSP 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 is given by a graph (N,E), where N is the set of towns and E is the set of edges between towns (a fully connected graph in the Euclidean TSP).

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:

- it chooses the town to go to with a probability that is a function of the town distance "visibility" and amount of trail present on the connecting edge;
- to force the ant to make legal tours, transitions to already visited towns are disallowed until a tour is completed (this is controlled by a tabu list);
- when it completes a tour, it 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. Each ant at time t chooses the next town, where it will be at time t+1. Therefore, if we call an *iteration* of the AS algorithm the m moves carried out by the m ants in the interval (t, t+1), then every n iterations of the algorithm (which we call a cycle) each ant has completed a tour. At this point the trail intensity is updated according to the following formula

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

where

ρ is a coefficient such that (1 - ρ) represents the *evaporation* of trail between time t and t+n,

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2}$$

where $\Delta\tau_{ij}^{k}$ 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; it is given by

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{if k - th ant uses edge } (i, j) \text{ in its tour (between time t and t + n)} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

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 (see note 1). The intensity of trail at time 0, $\tau_{ij}(0)$, can be set to arbitrarily chosen values; in our experiments for every edge (i,j) we set $\tau_{ij}(0)=c$, with c a small positive constant.

In order to satisfy the constraint that an ant visits all the n different towns, we associate with each ant a data structure called the *tabu list*[2], that saves the towns already visited up to time t and forbids the ant to visit them again before n iterations (a tour) have been completed. When a tour is completed, the tabu list is used to compute the ant's current solution (i.e., the distance of the path followed by the ant). The tabu list is then emptied and the ant is free again to choose. We define $\mathbf{tabu_k}$ the vector containing the tabu list of the k-th ant, $tabu_k$ the set obtained from the elements of $\mathbf{tabu_k}$, and $\mathbf{tabu_k}(s)$ the s-th element of the list (i.e., the s-th town visited by the k-th ant in the current tour).

We call *visibility* η_{ij} the quantity $1/d_{ij}$. This quantity is not modified during the run of the AS, as opposed to the trail which instead changes according to the previous formula (1).

We define the transition probability from town i to town j for the k-th ant as

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{k \in allowed_k} [\tau_{ik}(t)]^{\alpha} \cdot [\eta_{ik}]^{\beta}} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \qquad (4)$$

where $allowed_k = \{N - tabu_k\}$ and where α and β are parameters that 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 at time t (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).

---

[2] Even though the name chosen recalls tabu search, proposed in [17,18], there are substantial differences between our approach and tabu search algorithms. We mention here: (i) the absence in the AS of any aspiration function, (ii) the difference of the elements recorded in the tabu list, permutations in the case of tabu search, nodes in the AS (our algorithms are constructive heuristics, which is not the case of tabu search).

III. The algorithms

Given the definitions of the preceding section, the so-called *ant-cycle* algorithm is simply stated as follows. At time zero an initialization phase takes place during which ants are positioned on different towns and initial values $\tau_{ij}(0)$ for trail intensity are set on edges. The first element of each ant's tabu list is set to be equal to its starting town. Thereafter every ant moves from town i to town j choosing the town to move to with a probability that is a function (with parameters α and β, see formula (4)) of two desirability measures. The first, the trail $\tau_{ij}(t)$, gives information about how many ants in the past have chosen that same edge (i,j); the second, the visibility η_{ij}, says that the closer a town the more desirable it is. Obviously, setting $\alpha = 0$, the trail level is no longer considered, and a stochastic greedy algorithm with multiple starting points is obtained.

After n iterations all ants have completed a tour, and their tabu lists will be full; at this point for each ant k the value of L_k is computed and the values $\Delta\tau_{ij}^{k}$ are updated according to formula (3). Also, the shortest path found by the ants (i.e., $\min_{k} L_k$, k = 1, ..., m) is saved and all the 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 *stagnation behavior* because it denotes a situation in which the algorithm stops searching for alternative solutions. We investigate this situation in Section IV.

Formally the *ant-cycle* algorithm is:

1. Initialize:
   Set t:=0 {t is the time counter}
   Set NC:=0 {NC is the cycles counter}
   For every edge (i,j) set an initial value $\tau_{ij}(t)$ for trail intensity and $\Delta\tau_{ij}= 0$
   Place the m ants on the n nodes

2. Set s:=1 {s is the tabu list index}
   For k:=1 to m do
   Place the starting town of the k-th ant in $\mathbf{tabu_k}(s)$

3. Repeat until tabu list is full {this step will be repeated (n-1) times}
   Set s:=s+1
   For k:=1 to m do
   Choose the town j to move to, with probability $p_{ij}^{k}(t)$ given by equation (4)
   {the k-th ant is now on town i=$\mathbf{tabu_k}$(s-1) at time t}
   Move the k-th ant to the town j
   Insert town j in $\mathbf{tabu_k}(s)$

4. For k:=1 to m do
   Compute the length L_k of the tour described by $\mathbf{tabu_k}$
   Update the shortest tour found

For every edge (i,j)
For k:=1 to m do

$$\Delta\tau_{i,j}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{if } (i,j) \in \text{tour described by } \mathbf{tabu_k} \\ 0 & \text{otherwise} \end{cases}$$

$$\Delta\tau_{ij} := \Delta\tau_{ij} + \Delta\tau_{ij}^{k};$$

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

6. If (NC < NC_{MAX}) and (not stagnation behavior)
   then
   Empty all tabu lists
   Goto step 2
   else
   Print shortest tour
   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 step 1 is $O(n^2+m)$, step 2 is $O(m)$, step 3 is $O(n^2\cdot m)$, step 4 is $O(n^2\cdot m)$, step 5 is $O(n^2)$, step 6 is $O(n\cdot m)$. Because we have experimentally found a linear relation between the number of towns and the best number of ants (see Section V-A), the complexity of the algorithm is $O(NC\cdot n^3)$.

We also experimented with two other algorithms of the AS, which we called *ant-density* and *ant-quantity* algorithms [6]. They differ in the way the trail is updated. In these two models each ant lays its trail at each step, without waiting for the end of the tour. In the *ant-density* model a quantity Q of trail 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/d_{ij} of trail on edge (i,j) every time it goes from i to j. Therefore, in the *ant-density* model we have

$$\Delta\tau_{ij}^{k} = \begin{cases} Q & \text{if the k - th ant goes from i to j between time t and t + 1} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

and in the *ant-quantity* model we have

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{d_{ij}} & \text{if the k - th ant goes from i to j between time t and t + 1} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

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).

IV. Experimental study 1: Parameter setting and basic properties

We implemented the three algorithms (*ant-cycle*, *ant-density* and *ant-quantity*) of the AS 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 simulations to collect statistical data for this purpose.

The parameters considered here are those that affect directly or indirectly the computation of the probability in formula (4):

- α: the relative importance of the trail, $\alpha \bullet 0$;
- β: the relative importance of the visibility, $\beta \bullet 0$;
- ρ: trail persistence, $0 \bullet \rho < 1$ ($1-\rho$ can be interpreted as trail evaporation);
- Q: quantity of trail laid by ants.

The number m of ants has always been set equal to the number n of cities (see Section V-A for the explanation). We tested several values for each parameter while all the others were held constant (over ten simulations for each setting in order to achieve some statistical information about the average evolution). The default value of the parameters was $\alpha=1$, $\beta=1$, $\rho=0.5$, $Q=100$. In each experiment only one of the values was changed, except for α and β, which have been tested over different sets of values, as discussed at the end of this section. The values tested were: $\alpha \in \{0, 0.5, 1, 2, 5\}$, $\beta \in \{0, 1, 2, 5\}$, $\rho \in \{0.3, 0.5, 0.7, 0.9, 0.999\}$ and $Q \in \{1, 100, 10000\}$. Preliminary results, obtained on small-scale problems, have been presented in [6], [7], and [12], [13]; all the tests reported in this section are based, where not otherwise stated, on the *Oliver30* problem, a 30-cities problem described in [34][3]. All the tests have been carried out for NC_{MAX} = 5000 cycles and were averaged over ten trials.

To compare the three models we first experimentally determined the parameters best values for each algorithm, and then we ran each algorithm ten times using the best parameters set. Results are shown in Table I. Parameter Q is not shown becasue its influence was found to be negligible.

---

[3] In [34] genetic algorithms were applied to solve the Oliver30 problem; they could find a tour of length 424.635. The same result was often obtained by *ant-cycle*, which also found a tour of length 423.741.

Table I. Comparison among ant-quantity, ant-density, and ant-cycle. Averages over 10 trials.

	Best parameter set	Average result	Best result
ant-density	α=1, β=5, ρ=0.99	426.740	424.635
ant-quantity	α=1, β=5, ρ=0.99	427.315	426.255
ant-cycle	α=1, β=5, ρ=0.5	424.250	423.741

Both the *ant-density* and the *ant-quantity* models have given worse results than those obtained with *ant-cycle*. The reason is to be found in the kind of feedback information which is used to direct the search process. *Ant-cycle* uses global information, that is, its ants lay an amount of trail which is proportional to how good the solution produced was. In fact, ants producing shorter paths contribute a higher amount of trail than ants whose tour was poor. On the other side, both *ant-quantity* and *ant-density* use local information. Their search is not directed by any measure of the final result achieved. Therefore, it is not surprising that they gave worse performance results (details can be found in [6]).

The optimal value $\rho = 0.5$ in *ant-cycle* can be explained by the fact that the algorithm, after using the greedy heuristic to guide search in the early stages of computation, starts exploiting the global information contained in the values τ_{ij} of trail. *Ant-cycle* needs therefore to have the possibility to forget part of the experience gained in the past in order to better exploit new incoming global information.

Given that we found *ant-cycle* to be superior to the other two algorithms, we decided to deepen our understanding of the *ant-cycle* alone. Figures 3, 4, and 5 present traces of a typical run of the *ant-cycle* algorithm applied to the Oliver30 problem. In particular, Fig.3 shows the length of the best found tour at each cycle, and Fig.4 the standard deviation of the tour lengths of the population at each cycle of the same run. Note how in the early cycles the AS identifies good tours which are subsequently refined in the rest of the run. Since the standard deviation of the population's tour lengths never drops to zero, we are assured that the algorithm actively searches solutions which differ from the best-so-far found, which gives it the possibility of finding better ones. The search for better solutions is carried on in selected regions of the search space determined by the trail resulting from preceding cycles. This can be observed in Fig.5, in which the vertical axis shows the average node branching of the problem's graph. Although the graph is initially fully connected, those arcs whose trail level falls below a (very small) value ε, which makes their probability of being chosen by ants negligible, are removed. The node branching of node i is therefore given by the number of edges which exit from node i and which have a trail level higher than ε. Note how at the beginning of the run an ant could go

from any node to any other (except for tabu list constraints), while at the end the possible choices are significantly reduced.

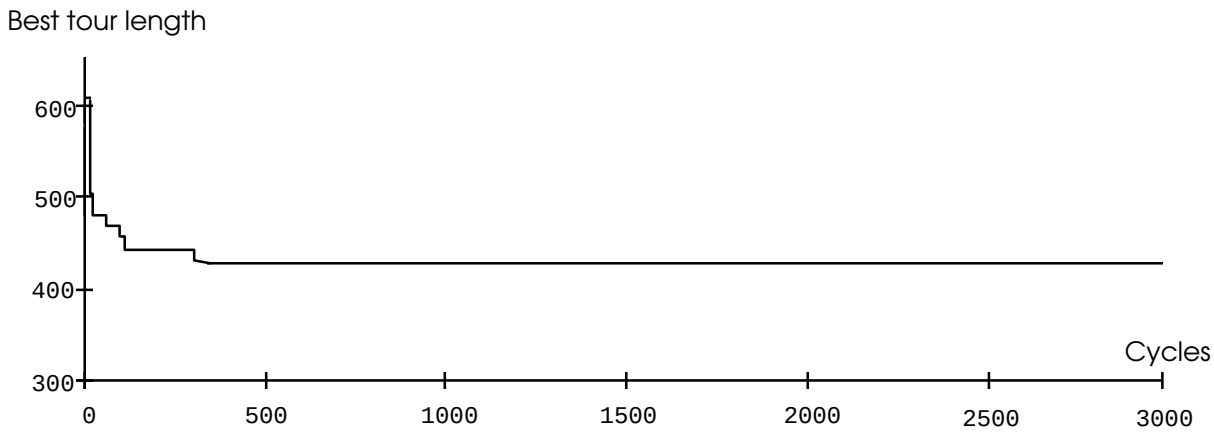


Fig. 3. Evolution of best tour length (Oliver30). Typical run.

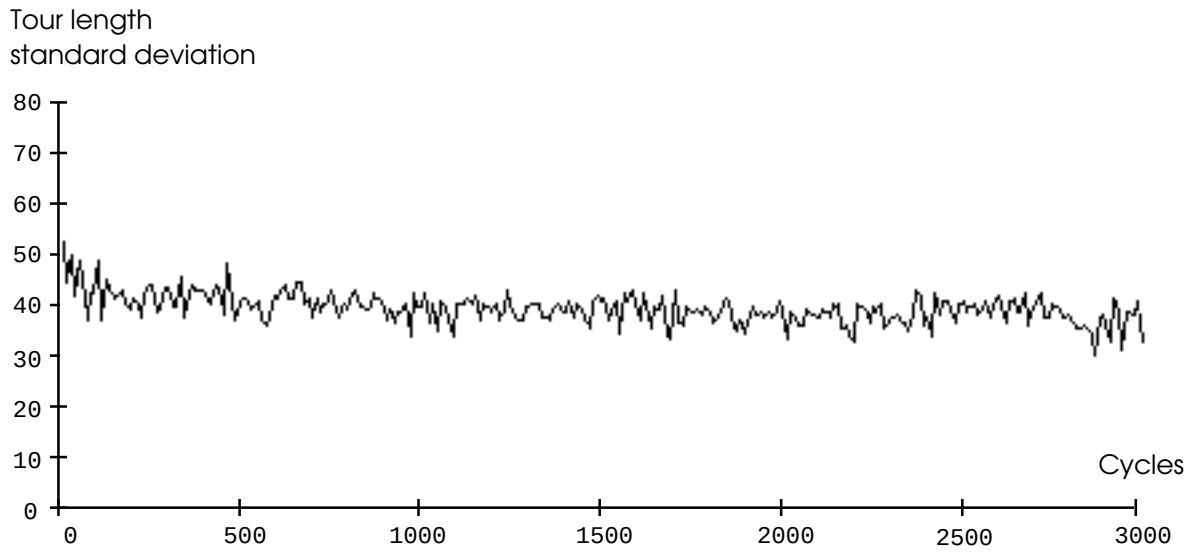


Fig. 4. Evolution of the standard deviation of the population's tour lengths (Oliver30). Typical run.

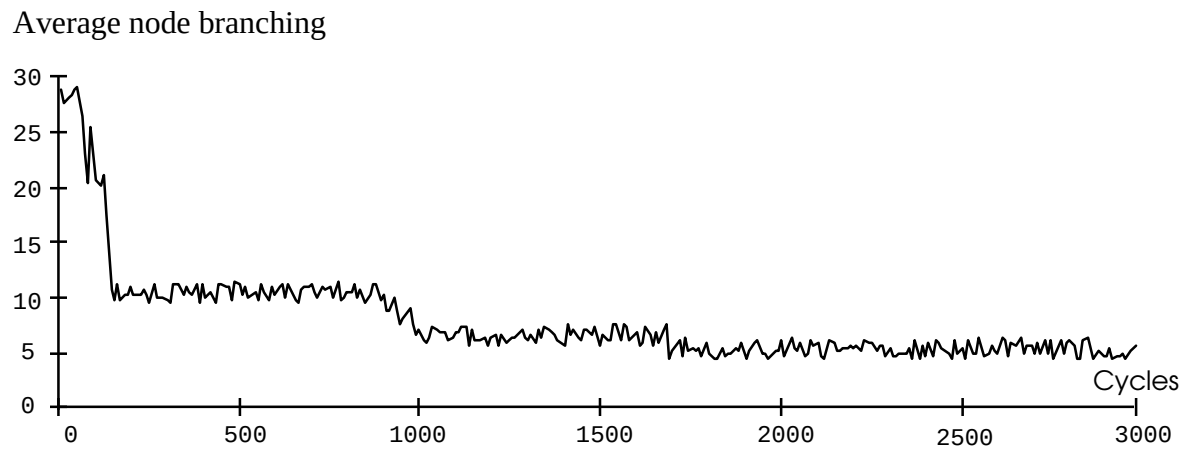
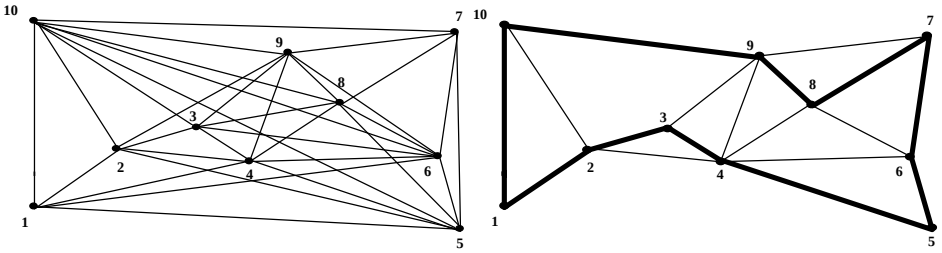


Fig. 5. Evolution of the average node branching of the problem's graph (Oliver30). Typical run.

The same process can be observed in the graphs of Fig. 6, where the AS was applied to a very simple 10-cities problem (CCA0, from [20]), and which depict the effect of ants search on the trail distribution. In the figure the length of the edges is proportional to the distances between the towns; the thickness of the edges is proportional to their trail level. Initially (Fig. 6a) trail is uniformly distributed on every edge, and search is only directed by visibilities. Later on in the search process (Fig. 6b) trail has been deposited on the edges composing good tours, and is evaporated completely from edges which belonged to bad tours. The edges of the worst tours actually resulted to be deleted from the problem graph, thus causing a reduction of the search space.



a) b)

Fig. 6. Evolution of trail distribution for the CCA0 problem.
a) Trail distribution at the beginning of search.
b) Trail distribution after 100 cycles.

Besides the tour length, we also investigated the *stagnation behavior*, i.e. the situation in which all the ants make the same tour. This indicates that the system has ceased to explore new possibilities and no better tour will arise. With some parameter settings we observed that, after several cycles, all the ants followed the same tour despite the stochastic nature of the algorithms because of a much higher trail level on the edges comprising that tour than on all the others. This high trail level made the probability that an ant chooses an edge not belonging to the tour very low. For an example, see the Oliver30 problem, whose evolution of average branching is presented in Fig. 7. In fact, after 2500 cycles circa, the number of arcs exiting from each node sticks to the value of 2, which – given the symmetry of the problem – means that ants are always following the same cycle.

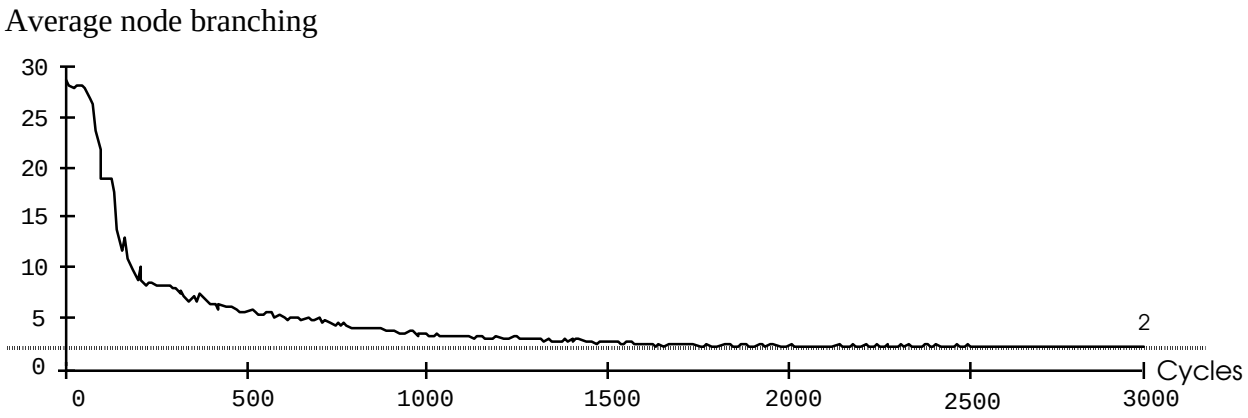
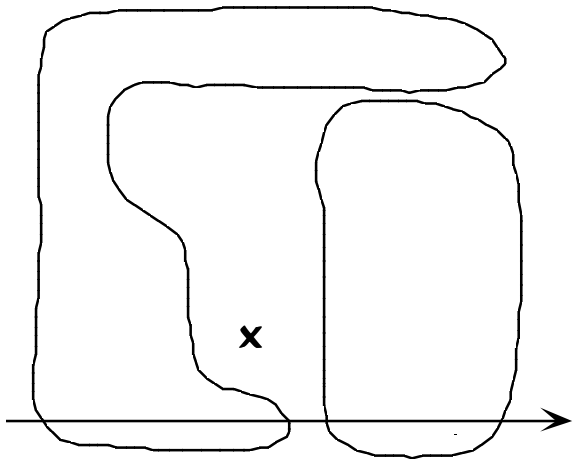


Fig. 7. Average node branching of a run going to stagnation behavior (Oliver30). Typical run obtained setting α=5 and β=2.

This led us to also investigate the behavior of the *ant-cycle* algorithm for different combination of parameters α and β (in this experiment we set NC_{MAX}=2500). The results are summarized in Fig. 8, which was obtained running the algorithm ten times for each set of parameters, averaging the results and ascribing each averaged result to one of the three following different classes.

- *Bad solutions and stagnation*. For high values of α the algorithm enters the stagnation behavior very quickly without finding very good solutions. This situation is represented by the symbol ∅ in Fig. 8;
- *Bad solutions and no stagnation*. If enough importance was not given to the trail (i.e., α was set to a low value) then the algorithm did not find very good solutions. This situation is represented by the symbol ∞.
- *Good solutions*. 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: the same result (the shortest tour known on the Oliver30 problem) was obtained in approximately the same number of cycles.



● - The algorithm finds the best known solution without entering the stagnation behavior.
∞ - The algorithm doesn't find good solutions without entering the stagnation behavior.
∅ - The algorithm doesn't find good solutions and enters the stagnation 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. On the other hand, low values of α make the algorithm very similar to a stochastic multigreedy algorithm.

In Fig. 9 we present the new optimal tour[4] found using the experimentally determined optimal set of parameters values for the *ant-cycle* algorithm, α=1, β=5, ρ=0.5, Q=100. This tour is of length 423.741 and presents two inversions, 2–1 and 25–24, with respect to the best tour published in [34].

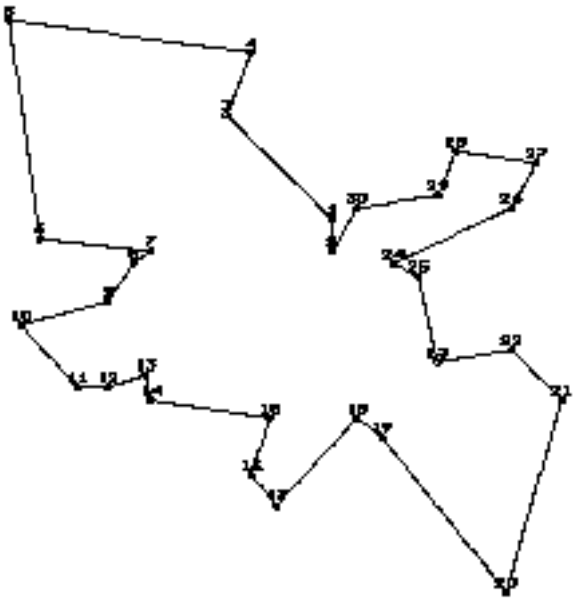


Fig. 9. The new best found tour obtained with 342 cycles of the *ant-cycle* algorithm for the Oliver30 problem (α=1, β=5, ρ=0.5, Q=100), real length = 423.741, integer length = 420.

The major strengths of the *ant-cycle* algorithm can be summarized as:

- Within the range of parameter optimality the algorithm always finds very good solutions for all the tested problems (Oliver30 and other problems which will be presented later).
- The algorithm quickly finds good solutions (see Fig. 10; for a comparison with other heuristics, see Section VI); nevertheless it doesn't exhibit stagnation behavior, i.e. the ants continue to search for new possibly better tours.

[4] This result is not competitive with results obtained by special-purpose algorithms [2].

- With increasing dimensions the sensitivity of the parameter values to the problem dimension has been found to be very low.

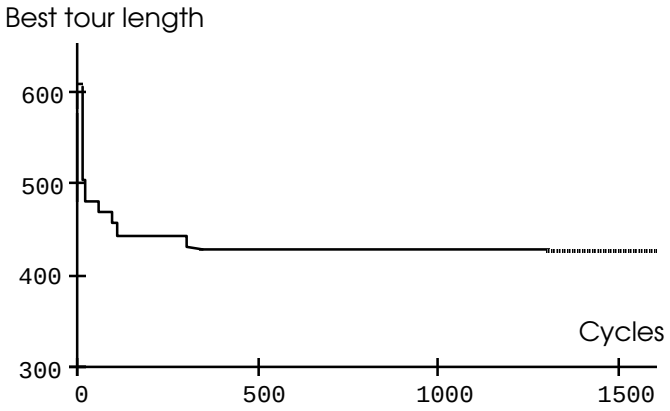


Fig. 10. The algorithm finds good values for Oliver30 very quickly and the new optimal value (423.741) after NC=342 cycles.

We partially tested the *ant-cycle* algorithm on the Eilon50 and Eilon75 problems [14] with a limited number of runs and with a number of cycles bounded by NC_{MAX}=3000. Under these restrictions we never got the best-known result, but a quick convergence to satisfactory solutions was maintained for both the problems.

V. Experimental study 2: Extensions and advanced properties

In this section we discuss experiments which have deepened our understanding of the *ant-cycle* algorithm. We study how synergy affects the algorithm performance (Section V-A). We compare the performance of *ant-cycle* when all the ants are initially positioned on a unique starting point with the performance obtained when each ant starts from a different town (Section V-B). Finally, we study the effects of an *elitist strategy* which increases the importance of the ant that found the best tour (Section V-C), and the change in performance of the AS when the problem dimension increases (Section V-D).

A. Synergistic effects

We ran a set of experiments to assess both the impact of the number m of ants, and the importance of communication through trail, 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 each edge has length 10, see Fig. 11).

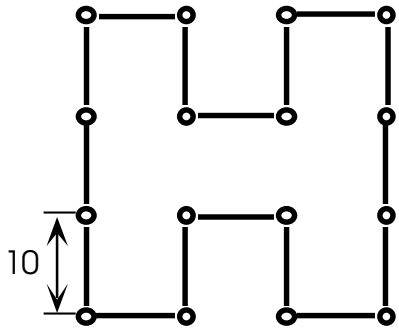
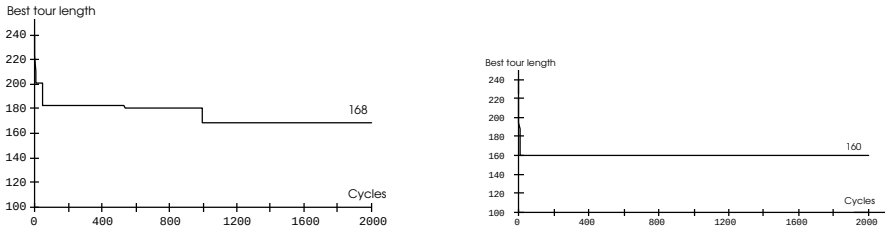


Fig. 11. An optimal solution for the 4x4 grid problem.

The result was that there is a synergistic effect in using many ants and using the trail communication system; that is, a run with n ants is more search-effective with communication among ants than with no communication. In case of communicating ants, there is an "optimality point" given by n•m in which the synergistic effects reach a maximum. The results are shown in Figs. 12 and 13.

In Fig. 12 we compare a situation in which ants do not communicate (α=0), with a situation in which they communicate (α=1). Results show that communication is indeed exploited by the algorithm. In Fig. 13 we report on an experiment in which the 4x4 grid problem was solved with m ∈ {4, 8, 16, 32, 64}. The abscissa shows the total number of ants used in each set of runs, the ordinate shows the so-called *one-ant cycles*, that is, the number of cycles required to reach the optimum, multiplied by the number of ants used (in order to evaluate the efficiency per ant, and have comparable data). The algorithm has always been able to identify the optimum with any number m • 4 of ants. Tests run on a set of r x r grid problems (r = 4, 5, 6, 7, 8) have substantiated our hypothesis 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 (Section III).



a) α=0 b) α=1

Fig. 12. Synergy: Communication among ants (α>0) improves performance.

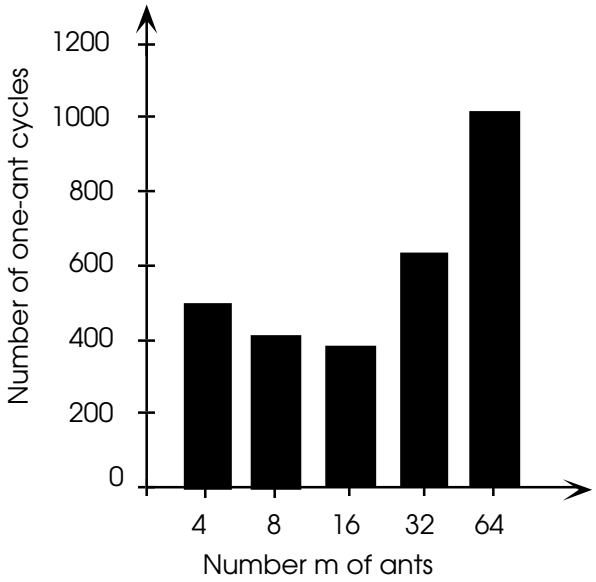


Fig. 13. Number of one-ant cycles required to reach optimum as a function of the total number of ants for the 4x4 grid problem. Results are averaged over five runs.

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 comparable with the dimension of the problem to be solved.

B. Initialization

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[5]. This experiment was run in order to know whether particular, possibly problem-specific, ant distributions are more effective than the uniform distribution, which in the previous experiments was used as the default starting configuration of ants. We used *ant-cycle* applied to the 16 cities random graph, to the 4x4 grid, and to the Oliver30 problem. In all three cases, uniformly distributing ants resulted in better performance.

We also tested whether an initial random distribution of the ants over the cities performed better than a uniform one; results show that there is no significant difference between the two choices, even though the random distribution obtained slightly better results.

---

5 We say ants are uniformly distributed if there is, at time t=0, the same integer number of ants on every town (this forces m to be a multiple of n).

C. Elitist strategy

We use the term "elitist strategy" (because in some way it resembles the elitist strategy used in genetic algorithms [19]) for the modified algorithm in which at every cycle the trail laid on the edges belonging to the best-so-far tour is reinforced more than in the standard version. We added to the trail of each arc of the best tour a quantity $e \cdot Q/L^*$, where e is the number of elitist ants[6] and L^* is the length of the best found tour. The idea is that the trail of the best tour, so reinforced, will direct the search of all the other ants in probability toward a solution composed by some edges of the best tour itself.

The test were carried out again on the Oliver30 problem (the run was stopped after $NC_{MAX} = 2500$ cycles) and results 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 in performance results. Fig. 14 shows the outcome of a test on the Oliver30 problem where this behavior is evident.

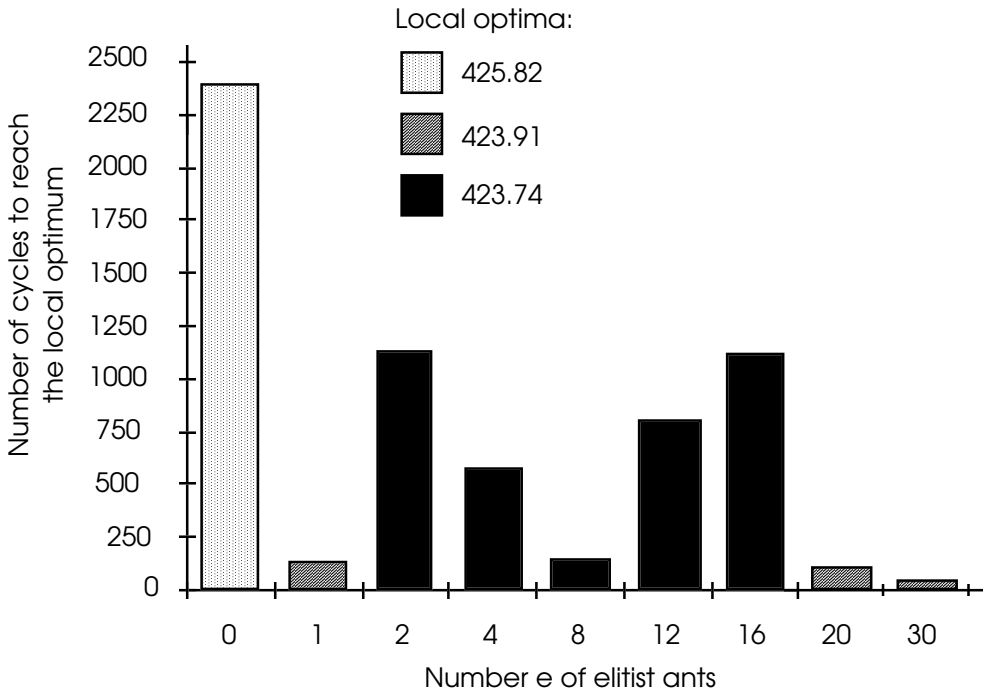


Fig. 14. Number of cycles required to reach a local optimum related to the number of elitist ants used (Oliver30). Results are averaged over five runs.

6 In our case the effect of an ant is to increment the value of the trail on edges belonging to its tour; therefore in our case the equivalent of “saving” an individual is to reinforce its contribution.

D. Increasing the problem dimensions

The algorithm complexity presented in Section III, $O(NC \cdot n^3)$, says nothing about the actual time required to reach the optimum. The experiment presented in this section is devoted to investigating the efficiency of the algorithm for increasing problem dimensions. Results are reported in Table II for the case of similar problems with increasing dimensions (r x r grids with the edge length set to 10, as in Fig. 11). It is interesting to note that, up to problems with 64 cities, the algorithm always found the optimal solution.

Table II. Time required to find optimum as a function of problem dimension. Results are averaged over five runs.

Problem	n (dimension)	Best solution	Average number of cycles to find the optimum	Time required to find the optimum[7] (seconds)
4 x 4	16	160	5.6	8
5 x 5	25	254.1	13.6	75
6 x 6	36	360	60	1020
7 x 7	49	494.1	320	13440
8 x 8	64	640	970	97000

VI. Comparison with other heuristics

In this section we compare the efficacy of our algorithm to that of other heuristics, both tailored and general-purpose.

A. Comparison with TSP-tailored heuristics

In this section we compare *ant-cycle* with the heuristics contained in the package "Travel" [4]. This package represents the distances between the cities as an integer matrix and so we implemented an analogous representation in our system[8]. The results of the comparisons on Oliver30 are shown in Table III, where the first column is the length of the best tour identified by each heuristic, and the second column is the improvement on the solution as obtained by the 2-opt heuristic (the 2-opt heuristic is an exhaustive exploration of all the permutations obtainable by exchanging 2 cities). Comparisons have been carried out also with the Lin-

7 Tests were run on a IBM-compatible PC with 80286 Intel processor.

8 In this case distances between towns are integer numbers and are computed according to the standard code proposed in [31].

Kernighan [27] improvement of the first-column solutions, which has been able to reduce the length of any tour to 420 (or 421, depending on the starting solution provided by the basic algorithms).

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. On the other hand, our algorithm requires a much longer computational time than any other tested special-purpose heuristic.

Table III. Performance of the *ant-cycle* algorithm compared with other approaches. Results are averaged over ten runs, and rounded to the nearest integer.

	basic[9]	2-opt	L-K
Ant-cycle	420	-	-
Near Neighbor	587	437421	
Far Insert	428	421420	
Near Insert	510	492420	
Space Filling Curve	464	431421	
Sweep	486	426420	
Random	1212	663421	

As a general comment of all the tests, we would like to point out that, given a good parameter setting (for instance α=1, β=5, ρ=0.5, Q=100, e=8), our algorithm consistently found the best known solution for the Oliver30 problem, and converged quickly towards satisfactory solutions. It always identified for Oliver30 the best-known solution of length 423.741 in less than 400 cycles, and it took only •100 cycles to reach values under 430. The algorithm never fell into the stagnation behavior. In fact, the average branching was always greater than 2, and the average length of tours was never equal to the best tour found but remained somewhat above it. This indicates that the ants followed different tours.

B. Comparison with general-purpose heuristics

We also compare *ant-cycle* with other general-purpose heuristics. This comparison is more fair to the AS, which in fact is a general-purpose heuristic, and not a specialized algorithm for the TSP. To run the comparisons, we implemented a Simulated Annealing (SA) [1], and a Tabu Search (TS) [16], [18]; we let each of them run 10 times on the Oliver30 data. SA used the annealing function $T(t+1)=\alpha T(t)$, with α=0.99; TS was implemented with tabu list length varying in [20, 50]. TS and SA, and the AS as well, were allowed to run for 1 hour on a IBM-compatible PC with 80386 Intel processor. The results are presented in Table IV.

9 The name "basic" means the basic heuristic, with no improvement.

Table IV. Performance of AS compared to TS and SA on the Oliver30 problem.
Results are averaged over ten runs using integer distances.

	Best	Average	Std.dev.
AS	420	420.4	1.3
TS	420	420.6	1.5
SA	422	459.8	25.1

Results show that the AS for this problem was as effective as TS and better than SA, when running under the same hardware and time constraints.

VII. Generality of the approach

As we said in Section I, the AS is both versatile and robust. Versatility is exemplified by the ease with which AS can be applied to the asymmetric TSP (ATSP), a particular kind of TSP (Section VII-A). Robustness is exemplified by the possibility of using the same algorithm, although appropriately adapted, to solve other combinatorial optimization problems like the quadratic assignment problem (QAP), and the job-shop scheduling problem (JSP) (Section VII-B).

A. Versatility: The ATSP

The asymmetric traveling salesman problem is a TSP in which the distance between two nodes is not symmetric (i.e., in general $d_{ij} \bullet d_{ji}$). The ATSP is more difficult than the TSP; in fact, while symmetric TSP can be solved optimally even on graphs with several thousand nodes, ATSP instances, and particularly ATSP instances where the distance matrix is almost symmetric, can be solved to the optimum only on graphs with a few dozen nodes [26], [16].

The application of the AS to the ATSP is straightforward, as no modifications of the basic algorithm are necessary. The computational complexity of a cycle of the algorithm remains the same as in the TSP application, as the only differences are in the distance and trail matrices which are no longer symmetric. We chose as test problem the RY48P problem [16], a very difficult problem instance with a distance distribution that is hard to solve even with tailored heuristics and branch and bound procedures. We ran AS 5 times on it, each time for 4000 cycles. The average length of the best found tour was 14899, that is 3.3% longer than the optimal one. The average number of cycles to find this result was 1517.

B. Robustness: QAP and JSP

Let's now consider the robustness of the AS approach. Many combinatorial problems can be solved by the AS. To apply the autocatalytic algorithm to a combinatorial problem requires defining:

1) an appropriate graph representation with search by many simple agents for the problem;
2) the autocatalytic (i.e. positive) feedback process;
3) the heuristic that allows a constructive definition of the solutions (which we also call "*greedy force*");
4) the constraint satisfaction method (that is, 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 AS that could effectively handle the relative problem. The most difficult (and *ad hoc*) tasks to face when applying the AS are to find an appropriate graph representation for the problem to be solved and a greedy force as heuristic.

1) Quadratic Assignment Problem:

A QAP of order n is the formalization of the problem that arises when trying to assign n facilities to n locations.

Formally the problem is usually defined by two nxn (symmetric) matrices:

D = $\{d_{ij}\}$ is the distance between location i and location j;

F = $\{f_{hk}\}$ is the flow (of information, products or some other quantity) between facility h and facility k;

A permutation π can be interpreted as an assignment of facility $h=\pi(i)$ to location i, for each i=1,..,n. The problem is then to identify a permutation π of both row and column indexes of the matrix **F** that minimizes the total cost:

$$\min z = \sum_{i,j=1}^{n} d_{ij} f_{\pi(i)\pi(j)}$$

To apply AS to QAP we used the same algorithm as in the case of the TSP, after having studied an approximation of the QAP objective function that allows a problem representation on the basis of a single matrix. The QAP objective function was expressed by a combination of the "potential vectors" of distance and flow matrices. The potential vectors, $\mathcal{D}$ and $\mathcal{F}$, are the row sums of each of the two matrices. Consider the following example:

$$\mathbf{D} = \begin{matrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 4 & 5 \\ 2 & 4 & 0 & 6 \\ 3 & 5 & 6 & 0 \end{matrix} \rightarrow \mathcal{D} = \begin{matrix} 6 \\ 10 \\ 12 \\ 14 \end{matrix} \qquad \mathbf{F} = \begin{matrix} 0 & 60 & 50 & 10 \\ 60 & 0 & 30 & 20 \\ 50 & 30 & 0 & 50 \\ 10 & 20 & 50 & 0 \end{matrix} \rightarrow \mathcal{F} = \begin{matrix} 120 \\ 110 \\ 130 \\ 80 \end{matrix}$$

From the two potential vectors, a third matrix **S** is obtained, where each element is computed as $s_{ih}=d_i \cdot f_h$, d_i and f_h being elements of the potential vectors.

$$\mathbf{S} = \begin{matrix} 720 & 660 & 780 & 480 \\ 1200 & 1100 & 1300 & 800 \\ 1440 & 1320 & 1560 & 960 \\ 1680 & 1540 & 1720 & 1120 \end{matrix}$$

The ants choose the node to move to using the inverse of the values of **S** as visibility data, $\eta_{ih}=1/s_{ih}$, thus interpreting each element s_{ih} as the choice of assigning to location i the facility h (so that, for example, 480 on the first row is linked to the choice of assigning to location 1 facility 4). The algorithm then goes on as in the case of the TSP, with trail laid down on the arc of the digraph whose "cost" matrix is **S**.

We compared AS, and a version of the AS with a local optimizing routine, with many other well know heuristics (see [28] for more details). Experiments were run on IBM-compatible PCs with a 80286 Intel processor, and were stopped after one hour time. The test problems used are those known as Nugent problems [29], Elshafei [15], and Krarup [25]. As can be seen in Table V, the performance of AS was always very good. Ant system always found a result within 5% of the best known, while AS with local optimization always found, except for the Nugent 30 problem, the best known solution.

Table V. Comparison of the AS with other heuristic approaches. Results are averaged over five runs. Best known results are in bold.

	Nugent (15)	Nugent (20)	Nugent (30)	Elshafei (19)	Krarup (30)
Best known	**1150**	**2570**	**6124**	**17212548**	**88900**
Ant System (AS)	**1150**	**2570**	6232	18122850	92490
AS with local optimization	**1150**	**2570**	6128	**17212548**	**88900**
Simulated Annealing	**1150**	**2570**	6128	17937024	89800
Tabu Search	**1150**	2688	**6124**	**17212548**	90090
Genetic Algorithm	1160	2654	6784	17640548	108830
Evolution Strategy	1168	**2570**	6308	19600212	97880
Sampling & Clustering	**1150**	2598	6154	**17212548**	**88900**

2) Job-shop Scheduling Problem

The JSP can be described as in the following. A set of M machines and a set of J jobs are given. The j-th job (j=1, ..., J) consists of an ordered sequence (chain) of operations from a set O={... o_{jm} ...}. Each operation o_{jm}

$\in$ O belongs to job j and has to be processed on machine m for d_{jm} consecutive time instants. N=|O| is the total number of operations. The problem is to assign the operations to time intervals in such a way that no two jobs are processed at the same time on the same machine and the maximum of the completion times of all operations is minimized [22].

To apply the AS to JSP we chose the following representation. A JSP with M machines, J jobs and operation set O is represented as an undirected, weighted graph Q=(O',A), where O'=O"{o_0} and A is the set of arcs that completely connect the nodes of O and that connect o_0 with the first operation of each job. Note that graph Q is not the graph with cliques representing machines that is usually utilized to represent the JSP. Node o_0 is necessary in order to specify which job will be scheduled first, in case several jobs have their first operation on the same machine. We have therefore N+1 nodes and $\frac{N(N-1)}{2}+|J|$ arcs, where all the nodes are pairwise connected except o_0, which is connected only to the first operation of each job. Each arc is weighted by a pair of numbers, {τ_{kl}, η_{kl}}. The first, τ_{kl}, is the trail level, while the second is the visibility η_{kl}, and is computed according to a desirability measure derived from a greedy problem specific heuristic like the Longest Processing Time or the Shortest Completion Time. The order in which the nodes are visited by each ant specifies the proposed solution. For instance, consider a 3x2 problem (3 jobs, 2 machines): it would be represented in our system by the graph presented in Fig. 15. We suppose the first machine processes operations 1, 3, 5, and the second one the others.

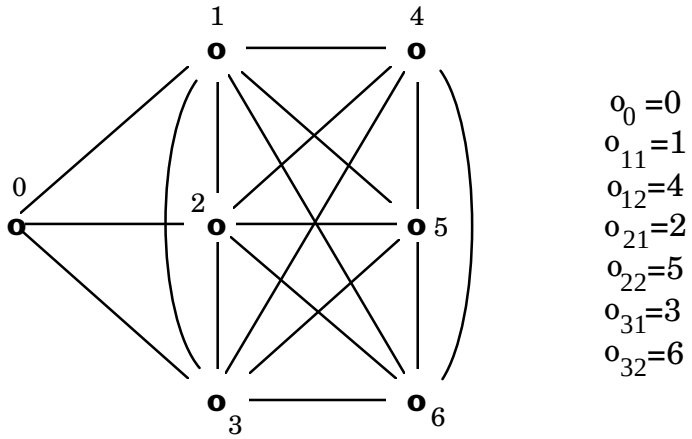


Fig. 15. AS graph for a 3 jobs and 2 machines JSP.

All ants are initially in o_0; later on they have to identify at each step a feasible permutation of the remaining nodes. To cope with this problem, transition probabilities have to be slightly modified with respect to those computed according to formula (4): in order to have a feasible permutation it is in fact necessary to define the set of allowed nodes in any step not only through the tabu list, but also in a problem-dependent way. For each ant k, let G_k be the set of all the nodes still to be visited and S_k the set of the nodes allowed at the next step.

Initially G_k={1, 2, 3, 4, 5, 6} and S_k={1, 2, 3}. Transition probabilities are computed on the basis of formula (4), where the set of allowed nodes is equal to S_k. When a node is chosen, it is appended to the tabu list and deleted from G_k and from S_k; if the chosen node is not the last in its job then its immediate successor in the job chain is added to S_k. This procedure ensures the possibility to always produce a feasible solution, possibly the optimal one. The process is iterated until $G_k=\varnothing$. At the end, the order of the nodes in the permutation given by the tabu list specifies the solution proposed by ant k. The trails can thus be computed in the usual way and they are laid down as specified by the ant cycle algorithm.

For example, suppose that an ant yielded the solution π=(0, 1, 4, 6, 2, 3, 5); this would direct the order of the operations imposing the precedences {(1,3), (3,5), (1,5)} and {(4,6), (6,2), (4,2)}, respectively.

This approach has been implemented and successfully applied to JSP instances of dimension 10x10 and 10x15 (10 jobs, 15 machines). For each of these problems we always obtained a solution within 10% of the optimum [8], which can be considered a promising result.

VIII. Discussion on some AS characteristics

A major issue in defining any distributed system is the definition of the communication protocol. In the AS a set of ants communicate by modifications of a global data structure: after each tour the trail left on each ant's tour will change the probability with which the same decision will be taken in the future. A heuristic also guides ants in the early stages of the computational process, when experience has not yet accumulated into the problem structure. This heuristic automatically loses importance (remember the coefficient ρ related to evaporation) as the experience gained by ants, and saved in the problem representation, increases.

One way to explain the behavior of AS on the TSP problem is the following. Consider the transition matrix $\mathbf{p}^k(t)$ of ant k: every element $p_{ij}^k(t)$ is the transition probability from town i to town j at time t as defined by equation (4). At time t=0 each $p_{ij}^k(0)$ is proportional to η_{ij}, i.e., closer towns are chosen with higher probability. As the process evolves, $\mathbf{p}^k(t)$ changes its elements according to (1) and (4). The process can therefore be seen as a space deformation, in which path cost is reduced between towns which are connected by edges with a high amount of traffic, and, conversely, path cost is incremented between towns connected by edges with low traffic levels. From simulations we observed that the matrix $\mathbf{p}^k(t)$, at least in the range of optimality for our parameters, converges to a state[10] that is very close to stationary (i.e., variations in the transition matrix $\mathbf{p}^k(t)$ are very small). When this state is reached the behavior of the ants is dependent on the

[10] The stochastic process that rules the evolution of the matrix $\mathbf{p}^k(t)$ is a Markov process with infinite memory.

kind of transition matrix obtained. We observed two situations: in the most rare one, occurring (as we saw in Section IV) for particular parameter settings, only two transition probabilities are 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 only a few transition probabilities with a significant value. In these cases search never stops, even if the number of significant transitions is highly reduced, with respect to the initial situation. Consider for example Fig. 16, obtained as the steady-state transition matrix for a randomly generated 10-town problem: 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 towns 9 and 10, for example, any destination is equally probable.

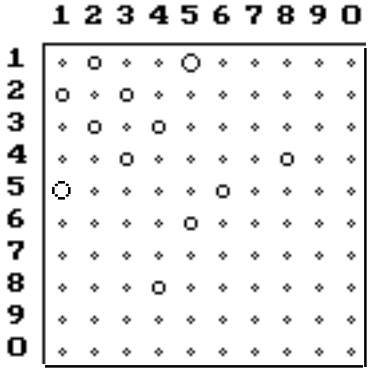


Fig. 16. The steady-state transition matrix for a randomly generated 10-town problem.

Another way to interpret how the algorithm works is to imagine having some kind of probabilistic superimposition of effects: each ant, if isolated (that is, 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 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 of some (initial) parts that are very good and some (final) parts that are not. If we now consider the effect of the simultaneous presence of many ants, then each one contributes to the trail distribution. Good parts of paths will be followed by many ants and therefore they receive a great amount of trail. On the contrary, bad parts of paths are chosen by ants only when they are obliged by constraint satisfaction (remember the tabu list); these edges will therefore receive trail from only a few ants.

IX. Conclusions

This paper introduces a new search methodology based on a *distributed autocatalytic process* and its application to the solution of a classical optimization problem. The general idea underlying the Ant System paradigm is that of a population of agents each guided by an autocatalytic process induced by a greedy force. Were an agent alone, both the autocatalytic process and the greedy force would tend to converge to a suboptimal tour with exponential speed. When agents interact it appears that the greedy force can give the right suggestions to the autocatalytic process and facilitate quick convergence to very good, often optimal, solutions without getting stuck in local optima. We have speculated that this behavior could 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 region of the space considered by the search process. Even if no tour is completely excluded, bad tours become highly improbable, and the algorithms search only in the neighborhood of good solutions.

The main contributions of this paper are the following.

(i) We employ positive feedback as a powerful search and optimization tool. The idea is that if at a given point an agent (ant) has to choose between different options and the one actually chosen results to be good, then in the future that choice will appear more desirable than it was before[11].

(ii) We show how synergy can arise and be useful in distributed systems. In the AS the effectiveness of the search carried out by a given number of cooperative ants is greater than that of the search carried out by the same number of ants, each one acting independently from the others.

(iii) We show how to apply the AS to different combinatorial optimization problems. After introducing the AS by an application to the TSP, we show how to apply it to the ATSP, the QAP, and the JSP.

At the present stage of our investigation we do not have any proof of convergence or bound on the time required to find the optimal solution. Nevertheless simulations strongly support the following conjecture: with good parameter settings the algorithm consistently converges in a polynomial number of steps to satisfactory solutions. We believe our approach to be a very promising one because of its generality (it can be applied to many different problems, see Section VII), and because of its effectiveness in finding very good solutions to difficult problems.

---

[11] Reinforcement of this nature is used by the reproduction-selection mechanism in evolutionary algorithms [23], [30], [33]. The main difference is that in evolutionary algorithms it is applied to favour (or disfavor) complete solutions, while in AS it is used to build solutions.

Related work can be classified in the following major areas:

- studies of social animal behavior;
- research in "natural heuristic algorithms";
- 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 in designing inherently parallel algorithms, observation of natural systems can be an invaluable source of inspiration. Neural networks [32], genetic algorithms [23], evolution strategies [30], [33], immune networks [3], simulated annealing [24] are only some examples of models with a "natural flavor". The main characteristics, which are at least partially shared by members of this class of algorithms, are the use of a natural metaphor, inherent parallelism, stochastic nature, adaptivity, and the use of positive feedback. Our algorithm can be considered as a new member of this class. All this work in "natural optimization" [12] fits within the more general research area of stochastic optimization, in which the quest for optimality is traded for computational efficiency.

Acknowledgments

We would like to thank two of the reviewers for the many useful comments on the first version of this paper. We also thank Thomas Bäck, Hughes Bersini, Jean-Louis Denebourg, Frank Hoffmeister, Mauro Leoncini, Francesco Maffioli, Bernard Manderik, Giovanni Manzini, Daniele Montanari, Hans-Paul Schwefel and Frank Smieja for the discussions and the many useful comments on early versions of this paper.

References

[1] E.H.L.Aarts, J.H.M.Korst, *Simulated Annealing and Boltzmann Machines*, John Wiley, 1988.

[2] J.J.Bentley, "Fast Algorithms for Geometric Traveling Salesman Problems," *ORSA Journal on Computing*, 4 (4), 387–411, 1992.

[3] H.Bersini, F.J.Varela, "The Immune Recruitment Mechanism: a Selective Evolutionary Strategy," *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, 520–526, 1991.

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

[5] R.E.Burkhard, "Quadratic Assignment Problems," European Journal of Operations Research, 15, 283–289, 1984.

[6] A.Colorni, M.Dorigo, V.Maniezzo, "Distributed Optimization by Ant Colonies," *Proceedings of the First European Conference on Artificial Life*, Paris, France, 134–142, 1991.

[7] A.Colorni, M.Dorigo, V.Maniezzo, "An Investigation of some Properties of an Ant Algorithm," *Proceedings of the Parallel Problem Solving from Nature Conference (PPSN 92)*, Brussels, Belgium, Elsevier Publishing, 509-520, 1992.

[8] A.Colorni, M.Dorigo, V.Maniezzo, M.Trubian, "Ant system for Job–Shop Scheduling," *Belgian Journal of Operations Research, Statistics and Computer Science*, 1994, in press.

[9] A.Colorni, M.Dorigo, F.Maffioli, V.Maniezzo, G.Righini, M.Trubian, "Heuristics from Nature for Hard Combinatorial Problems," *Technical Report No. 93–025*, Dip. Elettronica e Informazione, Politecnico di Milano, Italy, 1993.

[10] J.L.Denebourg, J.M.Pasteels, J.C.Verhaeghe, "Probabilistic Behaviour in Ants: a Strategy of Errors?," *Journal of Theoretical Biology*, 105, 259–271, 1983.

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

[12] M.Dorigo, *Optimization, Learning and Natural Algorithms*, Ph.D. Thesis, Dip. Elettronica e Informazione, Politecnico di Milano, Italy, 1992 (in Italian).

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

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

[15] Elshafei A.E., "Hospital Layout as a Quadratic Assignment Problem", *Operational Research Quarterly*, 28, 167–179, 1977.

[16] M.Fischetti, P.Toth, "An Additive Bounding Procedure for the Asymmetric Travelling Salesman Problem," Mathematical Programming, 53, 173–197, 1992.

[17] F.Glover, "Tabu Search—Part I," *ORSA Journal on Computing*, 1 (3), 190–206, 1989.

[18] F.Glover, "Tabu Search—Part II," *ORSA Journal on Computing*, 2 (1), 4–32, 1990.

[19] D.E.Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[20] 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.

[21] 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.

[22] R.L.Graham, E.L.Lawler, J.K.Lenstra, A.H.G.Rinnooy Kan, "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey", in *Annals of Discrete Mathematics*, 5, 287-326, 1979.

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

[24] S.Kirkpatrick, C.D.Gelatt, M.P.Vecchi, "Optimization by Simulated Annealing," *Science*, 220, 671–680, 1983.

[25] J. Krarup, P.M. Pruzan, "Computer-aided Layout Design", Mathematical *Programming Study*, 9, 85–94, 1978.

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

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

[28] V. Maniezzo, L.Muzio, A.Colorni, M.Dorigo, "The Ant System Applied to the Quadratic Assignment Problem," *Tech.Rep.94-058*, Dip. Elettronica e Informazione, Politecnico di Milano, Milano, 1992 (in Italian). Submitted to Rivista di Ricerca Operativa.

[29] C.E. Nugent, T.E. Vollmann, J. Ruml, "An Experimental Comparison of Techniques for the Assignment of Facilities to Locations," *Operations Research*, 16, 150–173, 1968.

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

[31] G.Reinelt, *TSPLIB 1.0*, Institut für Mathematik, Universität Augsburg, 1990.

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

[33] H.P. Schwefel, "Evolutionsstrategie und Numerische Optimierung," Ph.D. Thesis, Technische Universität Berlin, 1975. Also available as *Numerical Optimization of Computer Models*, J.Wiley, 1981.

[34] D.Whitley, T.Starkweather, D.Fuquay, "Scheduling Problems and Travelling Salesman: the Genetic Edge Recombination Operator," *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 1989.