The Ant System Applied to the Quadratic Assignment Problem

Vittorio Maniezzo and Alberto Colorni

Abstract—In recent years, there has been growing interest in algorithms inspired by the observation of natural phenomena to define computational procedures that can solve complex problems. In this article, we describe a distributed heuristic algorithm that was inspired by the observation of the behavior of ant colonies, and we propose its use for the Quadratic Assignment Problem. The results obtained in solving several classical instances of the problem are compared with those obtained from other evolutionary heuristics to evaluate the quality of the proposed system.

Index Terms—Heuristic algorithms, distributed algorithms, ant system, evolutionary computation, quadratic assignment problem, combinatorial optimization, knowledge pooling.

1 Introduction

The Quadratic Assignment Problem (QAP) of order n consists of looking for the best allocation of n activities facilities to n locations, where the terms activity and location should be considered in their most general sense. It was first formulated in [11] and since then it has been recognized as a model of many different real situations; applications have been described concerning planning of buildings in university campuses, arrangement of departments in hospitals, minimization of the total wire length in electronic circuits, ordering of correlated data in magnetic tapes, and others [2].

Mathematically, the problem is defined by three matrices of dimension $n \times n$:

1. $D = [d_{ih}] =$ matrix of the distances (between location i and location h);
2. $F = [f_{jk}] =$ matrix of the flows (between activity j and activity k); and
3. $C = [c_{ij}] =$ matrix of the linear assignment costs (of activity j to location i).

Normally, matrices **D** and **F** are integer-valued matrices, while the linear assignment cost c_{ij} of activity j to location i is usually ignored since it does not make a significant contribution to the complexity of solving the problem.

Under these hypotheses, a permutation $\Pi : i \rightarrow \pi(i)$ can be interpreted as a particular assignment of activity $j = \pi(i)$ to location i $(i = 1, \ldots, n)$.

The cost of transferring data (or materials, etc., depending on the problem in question) between two activities can be expressed as the product of the distance between the locations to which the activities are assigned by the flow between the two activities, $d_{ih} \cdot f_{\pi(i)\pi(h)}$.

To solve the QAP, one must thus find a permutation Π of the indices $(1, 2, \ldots, n)$, which minimizes the total assignment cost:

$$\min z = \sum_{i,h=1}^{n} d_{ih} \cdot f_{\pi(i)\pi(h)}. \tag{1}$$

The problem can be reformulated to show the quadratic nature of the objective function: Solving the problem means identifying a permutation matrix X of dimension $n \times n$ (whose elements x_{ij} are 1 if activity j is assigned to location i and 0 in the other cases) such that:

$$z_{QAP} = min\ z = \sum_{i,j=1}^{n} \sum_{h,k=1}^{n} d_{ih} f_{jk} x_{ij} x_{hk} \tag{2}$$

subject to the following constraints

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, \ldots, n), \tag{3}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, \ldots, n), \tag{4}$$

$$x_{ij} \in 0, 1 \qquad (i, j = 1, \ldots, n), \tag{5}$$

which identify the matrix **X** as belonging to set Π of the permutation matrices of order n.

As the QAP is a generalization of the Traveling Salesman Problem (TSP), it is also an NP-complete problem [20].

The techniques that can be used to find the optimal solution are limited to branch and bound and cutting planes methods; with current hardware, problems of order greater than 25 cannot be solved in an acceptable time [3].

- *V. Maniezzo is with Dipartimento di Informatica, Università di Bologna, Via Sacchi 3, 47023 Cesena, Italy. E-mail: maniezzo@csr.unibo.it.*
- *A. Colorni is with Dipartimento di Elettronica e Informazione, Politecnico di Milano, Via Ponzio 34/5, 20133, Milano, Italy. E-mail: colorni@elet.polimi.it.*

Manuscript received 17 June 1994; revised 17 June 1997. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 105261.

For this reason, many heuristic algorithms have been proposed in recent years which, although not ensuring that the solution found is the best one, give good results in an acceptable computation time [17].

In this article, we propose the use of a new heuristic procedure, improving over an algorithm originally developed for the TSP, which shows the emergence of global properties following the mutual interaction among many elementary agents [4], [5], [8]. In particular, we are interested in the distribution of search activity among agents which can only perform very simple actions, so that we can easily parallelize the computational effort (see [14] for a discussion on the effectiveness of parallelization for the QAP).

Our work was inspired by research on the behavior of ant colonies [7] , where one of the problems of interest is to understand how ants, which are almost blind animals with very simple individual capacities, can, when they act together in a colony, find the shortest route between two points (e.g., the ant's nest and a source of food).

The explanation lies in how the ants transmit information on the path followed: Each of them when it moves deposits a substance, called pheromone, which can be detected by the other ants. While an ant with no information moves essentially at random, an ant which follows a path already followed by others is tempted to follow the already marked path (and the probability of this occurring depends on the intensity of the trace perceived), in turn leaving new pheromone which is added to that already existing. The emerging collective effect is a form of *autocatalytic* or *positive feedback* behavior, in that the more ants follow a particular path, the more attractive this path becomes for the next ants which should meet it. The process is characterized by a positive feedback; in fact, the probability with which an ant chooses a path increases with the number of ants which have already chosen the same path. The final result is that nearly all the ants will choose to follow the shortest path, even if each ant's decision always remains probabilistic (that is, they can also explore new paths).

The algorithm, which we will define in the next section, is inspired by the observations made on ant colonies and is thus called the Ant System. A description of the original version of this method, and of its experimental results when applied to the Traveling Salesman Problem, can be found in [8].

2 The Ant System

In this section, we introduce a new heuristic (henceforth called the Ant System) for the QAP which uses some characteristics of behavior shown in reality by ant colonies, defining a system of artificial "ants." The Ant System presented in this paper represents an improvement of the algorithm described in [8], from which it differs in several structural elements. Specifically, the Ant System algorithm is designed to be a metaheuristic to guide global search by means of an updating of a global memory structure that represents the "knowledge" acquired by the system during the history of the search. Two problem-specific modules are supposed to be used in conjunction with the global search: local search and lower bounds. Local search is useful in order to focus global search in the space of local optima so as to increase the computational efficiency of the overall process, while lower bounding is a structural component of the Ant Search, as detailed in the following.

More in detail, in the Ant System, each artificial "ant" is an agent with the following characteristics:

1. when it chooses to assign activity j to location i, it leaves a substance, called *trace* (the equivalent of the pheromone) τ_{ij} on the coupling (i, j);
2. it chooses the location to which a given activity is to be assigned with a probability, which is a function of the "potential goodness" η_{ij} of the coupling (i, j) and of the quantity of trace present on the coupling itself; and
3. to construct a complete permutation, locations and activities already coupled are inhibited until all activities have been assigned.

This heuristic uses a population of m agents which construct solutions step by step, assigning an activity to each location. When all the ants have constructed their permutations, the best assignments are rewarded so as to encourage the identification of ever better solutions in the next cycles.

To satisfy the requirement that the ants assign each activity to a different location, we associate a data structure, called a *tabu list*, to each ant. This memorizes the activities already used and stops the ant assigning them to a new location before a cycle is complete (which thus identifies a permutation). Once the permutation is completed, the tabu list is emptied and the ant is free to choose its own couplings again. Let us define $\mathbf{tabu_k}$ a vector containing the tabu list for the *k*th ant and $\mathbf{tabu_k}(s)$ as the *s*th element of the tabu list for the *k*th ant (the activity occupying the *s*th location in the assignment made by the *k*th ant).

We now see how to introduce a method to calculate the "potential goodness" η_{ij} of an assignment and, thus, the initial assignment (when there is no trace); the initial situation will then be modified by the experience acquired by the population via the trace.

The basic idea is to exploit the information given by an effective lower bound to the completion of the problem solution and use it as an indicator of the expected proficiency of a particular pairing. The goodness η_{ij} of an assignment can, in fact, be estimated as the inverse of the lower bound obtained considering that pairing.

For the particular case of the QAP, several lower bounds have been proposed [19]. The best known one and, among those we tested, the one that had the best effectiveness/computational cost ratio, is the Gilmore and Lawler bound (independently presented by Gilmore [10] and Lawler [13]). The bound is obtained by computing a value z_{GL} as

$$z_{GL} = min\ z = \sum_{i,j=1}^{n} (min \sum_{h,k=1}^{n} d_{ih} f_{jk} x_{hk}) x_{ij}, \qquad (6)$$

where the minimum is computed subject to constraints (3), (4), and (5). This corresponds to solving n^2 linear assignment problems for defining the costs of the terms in brackets, and one further linear assignment for obtaining the GL bound. Obviously, $z_{GL} \leq z_{QAP}$. Using the same bounding strategy, it is also possible to obtain a lower bound to the value of the completion of a partial assignment. In fact, suppose that the index set $\Phi = \{1, 2, \ldots, n\}$ is partitioned into two subsets Φ_1 and Φ_2, corresponding to the indices of the already assigned facilities and to the indices of the still unassigned facilities, respectively.

Similarly, suppose that the index set $\Lambda = \{1, 2, \ldots, n\}$ is partitioned into two subsets Λ_1 and Λ_2, corresponding to the indices of the already assigned locations and to the indices of the still unassigned locations, respectively. Then, (2) can be rewritten as:

$$\begin{aligned} z_{QAP} = min\ z = &\sum_{i,h\in\Phi_1} \sum_{i,h\in\Lambda_1} d_{ih} f_{jk} x_{hk} x_{ij} \\ &+ \sum_{i,h\in\Phi_1} \sum_{i,h\in\Lambda_2} d_{jh} f_{jk} x_{ij} x_{hk} \\ &+ \sum_{i,h\in\Phi_2} \sum_{j,k\in\Lambda_1} d_{ih} f_{jk} x_{ij} x_{hk} \\ &+ \sum_{i,h\in\Phi_2} \sum_{j,k\in\Lambda_2} d_{ih} f_{jk} x_{ij} x_{hk}. \end{aligned} \tag{2'}$$

Notice that the first term of the objective function is now a known constant, z_1, and the fourth term is a reduced QAP instance to which (6) can be applied to obtain a bound z_4. A lower bound z_{23} to the value of the second and third terms can be obtained [2] by solving an assignment problem defined over a cost matrix $[\chi_{lm}], l \in \Phi_2, m \in \Lambda_2$, where:

$$\chi_{lm} = \sum_{i\in\Phi_1} \sum_{j\in\Lambda_1} (d_{il} f_{jm} + d_{li} f_{mj}). \tag{7}$$

A lower bound to the completion cost of a partial assignment can be thus computed as:

$$z_{LB} = z_1 + z_{23} + z_4. \tag{8}$$

On the basis of these results, to compute the attractiveness of a coupling $(i, j), i \in \Phi_2, j \in \Lambda_2$, we simply compute (8) for a partial assignment where, apart from the already specified couplings, we also tentatively locate facility i in location j. Therefore, we tentatively set: $\Phi_1 = \Phi_1 \cup \{i\}, \Lambda_1 = \Lambda_1 \cup \{j\}, \Phi_2 = \Phi_2 \setminus \Phi_1$ and $\Lambda_2 = \Lambda_2 \setminus \Lambda_1$, compute z_{LB} accordingly, and set $\eta_{ij} = z_{LB}$.

As an example, we consider Nugent's problem of order 5 [18] a problem arising in a hospital layout location context, with the distance **D** and flow **F** matrices shown below:

$$\mathbf{D} = \begin{matrix} 0 & 1 & 1 & 2 & 3 \\ 1 & 0 & 2 & 1 & 2 \\ 1 & 2 & 0 & 1 & 2 \\ 2 & 1 & 1 & 0 & 1 \\ 3 & 2 & 2 & 1 & 0 \end{matrix} \qquad \mathbf{F} = \begin{matrix} 0 & 5 & 2 & 4 & 1 \\ 5 & 0 & 3 & 0 & 2 \\ 2 & 3 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 5 \\ 1 & 2 & 0 & 5 & 0 \end{matrix}$$

The following execution trace shows how a solution is constructed. Assume for simplicity that solutions are constructed assigning facilities to locations of increasing indices, that is, first the facility to assign to location 1 is chosen, then the facility to assign to location 2, and so on. The construction goes on as follows. At the root node, no facility is assigned, so there are five possible assignments of facilities to location 1, whose costs are:

1. $z_1 = 0, z_{12} = 32, z_4 = 22$
2. $z_1 = 0, z_{12} = 24, z_4 = 28$
3. $z_1 = 0, z_{12} = 10, z_4 = 43$
4. $z_1 = 0, z_{12} = 18, z_4 = 32$
5. $z_1 = 0, z_{12} = 18, z_4 = 33$

The corresponding five lower bounds are therefore:

$$z_{LB} = 54\ 52\ 53\ 50\ 51$$

The choice goes to assigning facility 4 to location 1.

At the second level, one needs to define the assignment to location 2. Being one facility assigned, only four possibilities remain, whose costs are:

1. $z_1 = 8,\ z_{12} = 32,\ z_4 = 10$
2. $z_1 = 0,\ z_{12} = 56,\ z_4 = 6$
3. $z_1 = 0,\ z_{12} = 42,\ z_4 = 18$
5. $z_1 = 10,\ z_{12} = 16,\ z_4 = 24$

The corresponding four lower bounds are therefore:

$$z_{LB} = 50\ 62\ 60\ 50$$

The choice goes to assigning facility 1 to location 2.

At the third level, one needs to define the assignment to location 3. The three remaining possibilities have the following costs:

2. $z_1 = 28,\ z_{12} = 46,\ z_4 = 0$
3. $z_1 = 16,\ z_{12} = 50,\ z_4 = 4$
4. $z_1 = 22,\ z_{12} = 22,\ z_4 = 6$

The corresponding three lower bounds are therefore:

$$z_{LB} = 74\ 70\ 50$$

Facility 5 is thus chosen for location 3. The two remaining assignments are then considered explicitly, i.e., without going through lower bound computations.

One can thus find the complete permutation: Facility 4 is assigned to location 1, facility 1 to location 2, and so on, obtaining the permutations (4, 1, 5, 2, 3) of cost equal to 50. This happens to be the optimal cost for this trivial problem but, moving to more challenging instances, there is obviously no guarantee that the optimum will be found at the end of the construction of the first solution.

In the Ant System, the permutation is constructed probabilistically, using the Monte Carlo method, on the basis both of the η_{ij} values and of the values of the τ_{ij} variables, representing the trace levels. We define,

in fact, $\tau_{ij}(t+1)$ as the *trace intensity* (pheromone in the case of real ants) associated to the location i-facility j coupling.

The population has m ants, with k the generic ant $(k = 1, \ldots, m)$. The probability that the kth ant assigns facility j to location i is given by:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\alpha \cdot \tau_{ij}(t) + (1-\alpha)\cdot \eta_{ij}}{\sum_{r \notin tabu_k} (\alpha \cdot \tau_{ir}(t) + (1-\alpha)\cdot \eta_{ir})} & \text{if } j \notin tabu_k \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

with $0 \leq \alpha \leq 1$.

In constructing the permutation, we start from the location of index 1 and we assign a facility to it by choosing probabilistically from all the available facilities; at the second step, we assign to the second location a facility by choosing probabilistically among those that were not already assigned, and so on. The procedure is repeated for all n locations. The solution construction is repeated m times, as many times as there are ants in the population.

The parameter α allows the user to define the relative importance of the trace $\tau_{ij}(t)$ with respect to the desirability η_{ij}. Thus, the probability $p_{ij}^{k}(t)$ is a compromise between the desirability of a coupling (as indicated by the lower bound to the cost of a solution containing that assignment) and the trace intensity (if there was already a high "passage" of ants on coupling (i, j), then this coupling is probably very desirable).

Trail levels are updated after all the ants have constructed their solutions. The update is made according to the following:

$$\tau_{ij}(t+1) = \rho \tau_{ij}(t) + \Delta\tau_{ij}, \quad (10)$$

where ρ is a coefficient that represents the trace's *persistence* ($1 - \rho$ represents the evaporation) and:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \quad (11)$$

$\Delta\tau_{ij}^{k}$ being the quantity of trace left on the coupling (i, j) by the kth ant at the end of the construction of its permutation. The trace's initial intensity, $\tau_{ij}(0)$, can be set to a small and positive arbitrary value.

The coefficient ρ must be fixed to a value <1 to avoid an unlimited accumulation of trace. Concerning the quantity of trace left by the ants, different choices for the calculation of $\Delta\tau_{ij}^{k}$ determine the realization of slightly different algorithms. In the current version of the Ant System, $\Delta\tau_{ij}^{k}$ is given by the value Q/L_k if the kth ant has chosen coupling (i, j), and by the value 0 otherwise: Q is the current upper bound, i.e., the best solution found at the current iteration, while L_k is the value of the objective function obtained by the kth ant.

In this way, the best solutions (with a corresponding low L_k value) must be characterized by more trace on the couplings which determine low values of the objective function.

The basic algorithm, which uses the calculation of the bounds and which we will indicate by AS, is the following:

1. `t:=0`
   `Initialize the trace matrix`
   `Calculate the upper and lower bounds for the whole problem and the desirabilities` η_{ij}
   `Put` m `ants on node 1`
2. `For` $k := 1$ `to` m
   `Repeat {for each location}`
   `Choose, with probability given by equation (9), the facility to assign from those not yet assigned.`
   `Put the chosen facility in the tabu list of the k-th ant`
   `Until the tabu list is full`
   `{this cycle is repeated n times}`
   `End-for`
   `For k:=1 to m`
   `Carry the solution to its local optimum and compute` L_k
   `{the local search procedure is described at the end of this section}`
   `Update the best permutation found`
   `End-for`
3. `For each coupling (i,j) calculate` $\Delta\tau_{ij}$ `according to equation (11)`
   `Update the trace matrix according to equation (10)`
4. `If not (END_TEST)`
   `Empty the tabu lists of all the ants`
   `goto 2`
   `Else`
   `Print the best permutation and STOP`

The END_TEST is usually made either on a maximum number of iterations (steps from 2 to 5) or on a maximum CPU time allowed.

The algorithm's performance depends on the values of parameters ρ (trace persistence coefficient), α (importance of the trace), and m (number of ants). An experimental analysis for parameters setting will be presented in Section 3.

One can calculate an estimate of the complexity of the Ant System algorithm. After the initializations of complexity $O(n^3)$, as it implies the solution of a linear assignment problem, one must choose which facility will be assigned to the currently considered location: Probabilities are calculated according to (9) and the choice in probability is made between the facilities not yet assigned; the whole has complexity $O(n^2)$. To construct an entire permutation, one must thus perform $O(n^3)$ operations. Each complete iteration (m ants) thus requires a number of operations, $O(m \cdot n^3)$. When all the ants have constructed their solution, the trace matrix must be updated: $O(n^2)$ operations are required for this updating. The total complexity of an iteration of the algorithm is thus $O(m \cdot n^3)$.

As is the case for most constructive heuristics (see, for example, GRASP [15]) for the Ant System, efficiency improvements also can be achieved by using a local search procedure as a standard element of the overall algorithm. We thus designed a two-phase algorithm. The

first phase constructs solutions one element after the other, following the ant path. When an ant has constructed its basic permutation, a second phase of local search (see step 2) is activated and the trace is then added to the common data structure.

The local search procedure we implemented is a simple deterministic procedure. The cost of all possible exchanges is evaluated starting from the permutation obtained by the ant and choosing the exchange which most improves the objective function (see [21] for an efficient implementation of the variation due to an exchange).

The local search procedure is then the following:

```
Change:=true
While (change=true) do
   Explore the neighborhood of solution
        s(k) constructed by ant k and save the
        best adjacent solution s'(k)
   If f(s'(k))<f(s(k))
   then s(k):= s'(k)
   else change:=false
End-while
```

The complete exploration of the neighborhood of a solution requires a number of operations $O(n^2)$: In fact, the neighborhood consists of $n(n-1)/2$ permutations which can be obtained with exchanges of pairs of elements and evaluating the cost variation, once initialized the relevant data structures, requires a constant operation time [21]; the local search step, for medium-large problems, could become rather onerous in terms of computation time.

3 Experimental Results

The algorithm presented in this paper was coded in Fortran 77 and tested on a Pentium 166 MHz machine, running under DOS. The computational testing of the new algorithm was carried out by applying the code to standard test problems from the literature and comparing the results to those of an established heuristic running under identical experimental conditions.

Before comparing our code, we had to identify a good parameter setting. As a complete analysis of the model which suggests the optimum values of the parameters in each situation has not been developed, we performed several simulations, testing the algorithm on five different problems with various values of the control parameters α (importance of the trace) and ρ (trace persistence coefficient). We also studied how the number m of ants can influence the overall performance.

The problems chosen for the purpose of setting the algorithm parameters were: the Nugent problems [18] of dimension from 15 to 30, the Elshafei problem ([9]) of dimension 19, and a Krarup problem of dimension 30 ([12]). The optimal solution is known for all these problems up to dimension 20 while, for the two larger ones (Nugent 30 and Krarup 30), the best solutions found in the literature, as reported by Burkard et al. [3], were considered for the comparison.

We tested various values for each parameter (in a cœteris paribus framework) on five different simulations for each choice.

The values tested were: $\alpha = 0.3, 0.5, 0.9$ and $\rho = 0.7, 0.9, 0.95, 0.99$. We kept $\alpha = 0.5, \rho = 0.9$ as default values.

As well as solving the problems, we were also interested in studying the behavior of the ant population with regard to a possible "stagnation," a situation in which all the ants reconstruct the same solution; this situation indicates that the system has stopped exploring new possibilities and that the best solution found up to that point will probably not be improved any further. With some parameter values it was observed that, after many cycles, all the ants made the same couplings despite the algorithm's stochastic nature: This behavior is due to a much greater trace level on some couplings than on others. From this high trace level, it follows that the probability that an ant chooses a new coupling is very low and, thus, stagnation is produced.

The value 0.9 for α (independent of the other parameters) quickly led the ant population to stagnation around the suboptimum solutions. With parameter α at value 0.5, good solutions were found for all the problems (about 0 to 3 percent away from the best solution known), without observing stagnation of the population: This means that, at each cycle, new solutions belonging to a promising subset were tried.

Low values of parameter ρ reduce the algorithm's efficiency: It takes longer to find good solutions; the best results were obtained for $\rho = 0.95$.

The number of ants used does not seem to have a decisive influence on the overall performance, on condition that a quantity at least the same as the dimension (n) of the problem is used. This agrees with results obtained by a previous version of the Ant System applied to the TSP [8].

With the most effective parameters ($\alpha = 0.5$, $\rho = 0.95$, $m = n$) the basic algorithm AS found the optimal or best known solutions of all problems.

Table 1 gives the best known results, the Gilmore-Lawler lower bound, the average of the objective function of 500 randomly generated solutions, and the best results given by the Ant System in 10 minutes runs.

To evaluate the performance of the algorithm proposed, we compared it with one of the best performing metaheuristics so far proposed for the QAP, namely GRASP, in the version presented by Li et al. [15]. Both the Ant System and GRASP were run for 10 minutes on each problem instance.

The comparative computational experiments were carried out on problem instances taken from the QAPLIB library [3], plus one instance (MC33), which will be introduced in Section 4. In order to have a significant test suite, we used all instances of the QAPLIB database (by the time of writing of this paper) containing problems of dimension 20 to 40: Lower dimensions imply too easy problems, bigger dimensions lead to the need to augment the time limit of 10 minutes in order to have meaningful results.

TABLE 1
Best Known Results (Best), Gilmore-Lawler Lower Bound (GL Bound), Random Average Value (Random), and Results Obtained by the Ant System (AS) for the Problems Examined

	Nugent (15)	Nugent (20)	Nugent (30)	Elshafei (19)	Krarup (30a)
Best	1150	2570	6124	17212548	88900
GL bound	963	2057	4539	11971900	68360
Random	1588	3403	8120	58993040	134641
AS	1150	2570	6124	17212548	88900

For GRASP, the parameters used were the same as those that were used in Li et al. [14], that is, $\alpha = 0.5$ and $\beta = 0.1$, except for MaxIter, which was set to 2,048 as in Li et al. [15].

All experiments were run on the same Pentium PC 166 MHz machine mentioned at the beginning of this section. The results are presented in Table 2, which shows the following columns:

- PROBL: problem identifier; an asterisk indicates that the optimal solution for the problem is known.
- GL: Gilmore and Lawler bound.
- OPT/BK: optimal or best known solution.
- GRASP-best: best result obtained by GRASP over five runs of 10 minutes each.
- GRASP-%error: percentage error of the best solution obtained by GRASP.
- GRASP-t.best: average, over five runs, of the CPU time (in seconds) needed by GRASP to produce its best solutions.
- ANT-best: best result obtained by the Ant System over five runs of 10 minutes each.
- ANT-%error: percentage error of the best solution obtained by the Ant System.
- ANT-t.best: average, over five runs, of the CPU time (in seconds) needed by the Ant System to produce its best solutions.

The last two rows of Table 2 present:

- AVG: Average percentage distance from the optimum (or best known) solution computed over all 45 problems;
- MAX: maximal percentage distance from the optimum (or best known) solution computed over all 45 problems.

Table 2 shows that, under the mentioned experimental conditions, the Ant System has a better performance, in terms of quality of the best solution found, than GRASP on the problem tested: It finds a greater number of best known solutions, it has a smaller average percentage error and a smaller maximum error. On no problem did GRASP find a solution that improved over that found by the Ant System. Moreover, the time needed to find its best solution is on the average slightly smaller for the Ant System (138.99 seconds) than for GRASP (143.83 seconds), even though on individual problems GRASP could be more efficient than the Ant System.

4 A Real-World Testcase

In this section, we propose a real assignment problem, which can be modeled as QAP of order 33. The problem is the optimum allocation of services in the offices of a multinational company located in Milan, Italy, as described in [16].

The offices available are clustered into units, which are the elements of the three following buildings:

I. TOWER: A building of six identical floors, each divided into three units, numbered from 1 to 18 (three per floor).
II. BUILDING A: A three-floor construction near to the TOWER building, with direct pedestrian connections at the level of the first two floors (as well as the outside passage) and with three units per floor, numbered from 19 to 27.
III. BUILDING B: a construction with several floors, the first three of which are available for the company in question, detached from the previous buildings and connected to them by footpaths. Two units are available on each usable floor, numbered from 28 to 33. The whole is shown in Fig. 1.

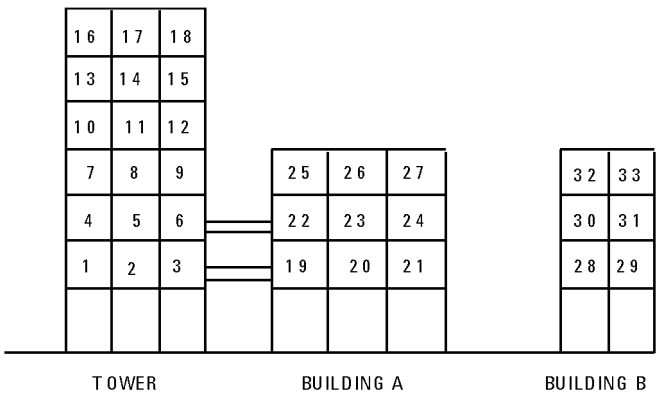

Fig. 1. Position of the units in the three buildings available.

TABLE 2
Comparison of Results Obtained by GRASP and by the Ant System

			GRASP			ANT		
PROBL	GL	OPT/BK	best	%error	t.best	best	%error	t.best
BUR26A	5189430	5426670	5426670	0.00	11.38	5426670	0.00	21.07
BUR26B	3626950	3817852	3817852	0.00	59.45	3817852	0.00	35.03
BUR26C	5165550	5426795	5426795	0.00	5.16	5426795	0.00	19.09
BUR26D	3609470	3821225	3821225	0.00	15.12	3821225	0.00	19.40
BUR26E	5155820	5386879	5386879	0.00	17.63	5386879	0.00	20.53
BUR26F	3601290	3782044	3782044	0.00	5.05	3782044	0.00	11.23
BUR26G	9368990	10117172	10117172	0.00	222.58	10117172	0.00	18.67
BUR26H	6547900	7098658	7098658	0.00	37.58	7098658	0.00	5.67
CHR20A*	2150	2192	2232	1.82	509.45	2192	0.00	331.20
CHR20B*	2196	2298	2434	5.92	195.00	2362	2.79	374.57
CHR20C*	8601	14142	14142	0.00	9.23	14142	0.00	29.49
CHR22A*	5924	6156	6298	2.31	200.66	6156	0.00	314.68
CHR22B*	5936	6194	6354	2.58	213.41	6254	0.97	161.75
CHR25A*	2765	3796	3884	2.32	114.95	3796	0.00	236.29
ESC32A	35	130	132	1.54	7.03	130	0.00	226.47
ESC32B	96	168	168	0.00	2.80	168	0.00	40.59
ESC32C	350	642	642	0.00	0.00	642	0.00	0.08
ESC32D	106	200	200	0.00	1.92	200	0.00	2.13
ESC32E*	0	2	2	0.00	0.00	2	0.00	0.05
ESC32F*	0	2	2	0.00	0.00	2	0.00	0.08
ESC32G*	0	6	6	0.00	0.00	6	0.00	0.07
ESC32H	257	438	438	0.00	3.41	438	0.00	2.64
HAD20*	6166	6922	6922	0.00	2.80	6922	0.00	158.71
LIPA20A*	3667	3683	3683	0.00	0.99	3683	0.00	107.32
LIPA20B*	27076	27076	27076	0.00	0.66	27076	0.00	0.00
LIPA30A*	13147	13178	13178	0.00	46.43	13178	0.00	54.85
LIPA30B*	151426	151426	151426	0.00	7.31	151426	0.00	0.00
LIPA40A*	31497	31538	31893	1.13	306.38	31859	1.02	281.00
LIPA40B*	476581	476581	476581	0.00	6.21	476581	0.00	0.00
KRA30A	68360	88900	88900	0.00	292.03	88900	0.00	199.06
KRA30B	69065	91420	91710	0.32	267.80	91420	0.00	140.02
NUG20*	2057	2570	2570	0.00	2.53	2570	0.00	119.28
NUG30	4539	6124	6150	0.42	522.97	6124	0.00	180.75
ROU20*	599948	725522	725522	0.00	164.55	725522	0.00	244.54
SCR20*	86766	110030	110030	0.00	157.47	110030	0.00	46.09
STE36A	7124	9526	9698	1.81	275.77	9598	0.76	295.23
STE36B	8653	15852	15998	0.92	179.62	15892	0.25	212.81
STE36C	6393630	8239110	8312752	0.89	141.87	8265934	0.33	321.16
TAI20A*	580674	703482	703482	0.00	483.80	703482	0.00	160.07
TAI25A	962417	1167256	1182914	1.34	354.51	1173672	0.55	206.17
TAI30A	1504690	1818146	1846888	1.58	264.62	1844676	1.46	331.72
TAI35A	1951210	2422002	2467946	1.90	531.32	2461764	1.64	231.59
TAI40A	2492850	3139370	3208452	2.20	324.84	3203836	2.05	252.83
THO30	90578	149936	149936	0.00	215.88	149936	0.00	287.50
THO40	143804	240516	243320	1.17	184.39	242108	0.66	312.46
UFFICI	209821	339416	339416	0.00	249.39	339416	0.00	379.65
AVG				0.66			0.27	
MAX				5.92			2.79	

Distance matrix

0	9	12	54	56	60	66	68	72	78	80	84	90	92	96	102	104	108	40	40	43	82	82	85	104	104	107	234	236	246	248	258	260
	0	15	56	58	62	68	70	74	80	82	86	92	94	98	104	106	110	42	42	45	84	84	87	106	106	109	236	238	248	250	260	262
		0	60	62	66	72	74	78	84	86	90	96	98	102	108	110	114	46	46	49	88	88	91	110	110	113	240	242	252	254	264	266
			0	9	12	54	56	60	66	68	72	78	80	84	90	92	96	82	82	85	40	40	43	92	92	95	246	248	258	260	270	272
				0	15	56	58	62	68	70	74	80	82	86	92	94	98	84	84	87	42	42	45	94	94	97	248	250	260	262	272	274
					0	60	62	66	72	74	78	84	86	90	94	98	102	88	88	91	46	46	49	98	98	101	252	254	264	266	276	278
						0	9	12	54	56	60	66	68	72	78	80	84	94	94	97	82	82	85	134	134	137	258	260	270	272	282	284
							0	15	56	58	62	68	70	74	80	82	86	96	96	99	84	84	87	136	136	139	260	262	272	274	284	286
								0	60	62	66	72	74	78	84	86	90	100	100	103	88	88	91	140	140	143	264	266	276	278	288	290
									0	9	12	54	56	60	66	68	72	106	106	109	94	94	97	146	146	149	270	272	282	284	294	296
										0	15	56	58	62	68	70	74	108	108	111	96	96	99	148	148	151	272	274	284	286	296	298
											0	60	62	66	72	74	78	112	112	115	100	100	103	152	152	155	276	278	288	290	300	302
												0	9	12	54	56	60	118	118	121	106	106	109	158	158	161	282	284	294	296	306	308
													0	15	56	58	62	120	120	123	108	108	111	160	160	163	284	286	296	298	308	310
														0	60	62	66	124	124	127	112	112	115	164	164	167	288	290	300	302	312	314
															0	9	12	130	130	133	118	118	121	170	170	173	294	296	306	308	318	320
																0	15	132	132	135	120	120	123	172	172	175	296	298	308	310	320	322
																	0	136	136	139	124	124	127	176	176	179	300	302	312	314	324	326
																		0	6	8	60	60	63	72	72	75	192	194	204	206	216	218
																			0	6	60	60	63	72	72	75	192	194	204	206	216	218
																				0	63	63	66	75	75	78	195	197	207	209	219	221
																					0	6	8	60	60	63	204	206	216	218	228	230
																						0	6	60	60	63	204	206	216	218	228	230
																							0	63	63	66	207	209	219	221	231	233
																								0	6	8	216	218	228	230	240	242
																									0	6	216	218	228	230	240	242
																										0	219	221	231	239	243	245
																											0	8	70	72	82	84
																												0	72	74	84	86
																													0	8	70	72
																														0	72	74
																															0	8
																																0

Fig. 2. The distance matrix of the MC33 instance.

The distance matrix is made of the times (in seconds) spent by an employee to move from location i to location $\mathrm{h}(\mathrm{i}, \mathrm{h} = 1, \ldots, 33)$.

For simplicity, the distances between the units on the various floors of each building are considered as identical, even though sometimes there are mandatory paths which may cause small differences. The distances are estimated on the basis of the conditions of normal activity of the offices themselves (waiting times for the service lifts and/or any use of alternative routes, walkways, or stairs).

As "flow between activities" we decided to use the number of personal contacts necessary on average in a week by the employees of various offices, weighted according to the qualification of the person involved (the employees were assigned weight 1 and the managers weight 2), thus trying to correlate the movements to the effective burden in terms of working costs. The matrix of the flows between the various activities was obtained by quali-/quantitative indications obtained from all the managers of the various services. The distance and flow matrices are reported in Fig. 2 and Fig. 3.

The objective function of the permutation (3, 4, 5, 14, 16, 17, 25, 26, 15, 24, 8, 9, 10, 2, 11, 1, 6, 7, 29, 31, 30, 18, 22, 23, 20, 21, 19, 27, 28, 32, 33, 12, 13) corresponding to the current location of the offices in the units was initially calculated: It produces a value of 438,114 man-seconds per week ($\approx$ 121.7 man-hours).

Flow matrix

0	10	20	4	7	24	30	6	8	2	4	6	5	10	4	10	0	14	6	6	0	10	0	4	12	4	4	1	10	0	0	0	0
	0	2	1	1	4	8	2	2	0	4	0	0	4	2	2	0	6	2	2	0	2	0	0	0	0	2	1	10	0	4	0	0
		0	13	7	20	24	10	10	0	2	4	0	10	0	4	1	10	2	4	1	4	1	2	2	7	0	0	36	2	0	0	0
			0	8	7	2	2	2	0	0	1	0	4	0	0	0	4	0	0	0	0	0	0	5	3	0	0	5	2	1	0	0
				0	2	1	1	1	0	0	0	1	2	0	0	0	3	0	0	0	0	0	0	2	1	0	0	3	0	0	0	0
					0	20	10	10	4	16	14	9	20	8	12	0	24	8	8	0	10	0	4	6	7	0	0	10	2	0	0	0
						0	4	4	0	0	4	3	12	0	0	0	16	0	0	0	0	0	20	24	20	10	7	36	13	20	7	10
							0	0	1	0	4	0	8	0	0	0	0	0	0	0	0	0	0	4	0	0	0	4	0	0	0	0
								0	1	0	4	0	0	0	0	0	12	0	0	0	0	0	0	4	0	0	0	4	0	0	0	0
									0	0	0	0	6	0	10	3	8	0	8	2	8	3	0	0	2	0	1	4	1	0	1	0
										0	8	5	4	4	4	0	4	4	2	0	2	0	0	4	0	0	0	2	0	0	0	0
											0	3	10	20	4	0	0	4	0	0	0	0	0	4	0	0	0	4	0	2	1	0
												0	0	4	0	0	10	24	4	0	6	0	0	4	0	0	0	4	0	2	1	0
													0	30	40	20	6	4	4	0	4	0	4	4	5	4	3	10	4	4	1	5
														0	10	0	0	4	0	0	0	0	0	4	4	2	0	4	1	4	3	0
															0	27	0	0	4	0	4	0	4	4	3	0	0	4	3	2	0	3
																0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	5
																	0	30	30	11	30	11	6	6	7	4	4	10	4	4	1	7
																		0	10	0	10	0	0	4	4	2	0	4	1	4	3	0
																			0	27	10	0	4	4	2	0	0	4	3	2	0	3
																				0	0	0	0	0	0	0	0	2	0	0	0	5
																					0	27	4	4	3	0	0	4	3	2	0	3
																						0	0	0	0	0	0	1	0	0	0	5
																							0	10	13	4	3	10	7	4	1	1
																								0	20	4	1	4	0	0	0	0
																									0	3	2	13	5	7	3	3
																										0	0	6	7	10	7	0
																											0	3	5	7	5	0
																												0	13	24	11	7
																													0	20	8	6
																														0	13	13
																															0	0
																																0

Fig. 3. The flow matrix of the MC33 instance.

If this datum is compared with the average value of a random arrangement (565,541 man-seconds, calculated as the average between 100 permutations generated at random), one can conclude that the actual logistic situation allows a "saving" of about 22.5 percent as compared to a random allocation.

The best permutation found with the Ant System algorithm, as reported in the last row of Table 2, has a value of 339,416 man-seconds (≈ 94.3 man-hours). This solution is 22.5 percent better than the current logistic situation (obviously, this datum must be taken with due care, as the current assignment derives not only from cost considerations, but also from other less quantifiable objectives such as personal preferences, prestige of a location, ...).

5 Conclusions

In this work, we presented a distributed heuristic algorithm, the Ant System, applied to the Quadratic Assignment Problem.

The main point in each distributed system is the definition of the communication procedure among agents. In our algorithm, a set of ants communicates by modifying the problem's representation as, at each step of the processing, each ant leaves a sign of its activity which changes the probability with which the decisions will be made in future. The idea is that, if an

ant in a given state must choose between different options and, having made a choice, that choice results as particularly good, then in the future that choice must appear more desirable whenever the state and the options are the same.

The ants are given a heuristic to guide the initial steps of the computation process when the information on the problem structure given by the trace has not yet accumulated. This initial heuristic then automatically loses importance (by means of trace accumulation) when the experience acquired by the ants, saved in the trace matrix, grows.

The result presented in this work is, thus, about the use of an autocatalytic process as a method for optimization and learning. The process of an individual ant quickly converges to a possibly poor solution; the interaction of many autocatalytic processes can instead lead to convergence toward a region of the space containing good solutions so that very good solutions can be found by means of local optimization (without, however, being stuck on it). In other words, the ant population does not converge on a single solution, but on a set of (good) solutions; the ants continue their search to further improve the best solution found.

The results obtained showed the Ant System's competitive performance on all test problems.

APPENDIX A

DISTANCE AND FLOW MATRICES FOR THE COMPANY PROBLEM

Fig. 2. and Fig. 3 illustrate the distance and flow matrices, respectively, for the Italian company problem.

REFERENCES

[1] A. Bruengger, J. Clausen, A. Marzetta, and M. Perregaard, "Joining Forces in Solving Large-Scale Quadratic Assignment Problems in Parallel," DIKU technical report, Univ. of Copenhagen, Denmark, 1996.

[2] R.E. Burkard, "Quadratic Assignment Problems," *European J. Operational Research,* vol. 15, pp. 283-289, 1984.

[3] R.E. Burkard, S.E. Karisch, and F. Rendl, "QAPLIB—A Quadratic Assignment Problem Library," Technical Report No. 287, Technical Univ. of Graz, Austria, 1994.

[4] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed Optimization by Ant Colonies," *Proc. ECAL '91, European Conf. Artificial Life,"* Paris, Elsevier, pp. 134-142, 1991.

[5] A. Colorni, M. Dorigo, and V. Maniezzo, "An Investigation of Some Properties of an Ant Algorithm," *Proc. Parallel Problem Solving from Nature Conf. (PPSN 92),* Brussels, Elsevier, pp. 509-520, 1992,

[6] V.-D. Cung, T. Mautor, P. Michelon, and A. Tavares, "A Scatter Search Based Approach for the Quadratic Assignment Problem," *Proc. IEEE ICEC '97 Conf.,* Indianapolis, 1997.

[7] J.L. Denebourg, J.M. Pasteels, and J.C. Verhaeghe, "Probabilistic Behavior in Ants: A Strategy of Errors?" *J. Theoretical Biology,* vol. 105, pp. 259-271, 1983.

[8] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant System: Optimization by a Colony of Cooperating Agents," *IEEE Trans. Systems, Man, and Cybernetics,* Part B: Cybernetics, vol. 26, no. 1, pp. 29-41, 1996.

[9] A.E. Elshafei, "Hospital Layout as a Quadratic Assignment Problem," *Operations Research Quarterly,* vol. 28, pp. 167-179, 1977.

[10] P. Gilmore, "Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem," *J. SIAM,* vol. 10, pp. 305-313, 1962.

[11] T.C. Koopmans and M.J. Beckmann, "Assignment Problems and the Location of Economic Activities," *Econometrica,* vol. 25, pp. 53-76, 1957.

[12] J. Krarup and P.M. Pruzan, "Computer Aided Layout Design," *Math. Programming Study,* vol. 9, pp. 85-94, 1978.

[13] E. Lawler, "The Quadratic Assignment Problem," *Management Science,* vol. 9, pp. 586-599, 1963.

[14] Y. Li and P.M. Pardalos, "Parallel Algorithms for the Quadratic Assignment Problems," P.M. Pardalos, ed., *Advances in Optimization and Parallel Computing,* pp. 177-189, Elsevier, 1992.

[15] Y. Li, P.M. Pardalos, and M.G.C. Resende, "A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem," P.M. Pardalos and H. Wolkowicz, eds., *Quadratic Assignment and Related Problems,* DIMACS discrete mathematics and theoretical computer science series, vol. 16, pp. 237-261, 1994.

[16] V. Maniezzo, L. Muzio, A. Colorni, and M. Dorigo, "Il sistema formiche applicato al problema dell'assegnamento quadratico," Technical Report No. 94-058, Politecnico di Milano, Italy, 1994 (in Italian).

[17] V. Maniezzo, A. Colorni, and M. Dorigo, "Algodesk: An Experimental Comparison of Eight Evolutionary Heuristics Applied to the Quadratic Assignment Problem," *European J. Operational Research,* vol. 81, pp. 188-205, 1995.

[18] C.E. Nugent, T.E. Vollman, and J. Ruml, "An Experimental Comparison of Techniques for the Assignment of Techniques to Locations," *Operations Research,* vol. 16, pp. 150-173, 1968.

[19] *Quadratic Assignment and Related Problems,* P.M. Pardalos and H. Wolkowicz, eds., DIMACS series, Am. Math. Soc., vol. 16, pp. 555-565, 1994.

[20] S. Sahni and T. Gonzales, "P-Complete Approximation Problems," *J. ACM,* vol. 23, pp. 555-565, 1976.

[21] E. Taillard, "Robust Taboo Search for the Quadratic Assignment Problem," *Parallel Computing,* vol. 17, pp. 443-455, 1991.

Vittorio Maniezzo received the Laurea (masters of technology) degree in electronic engineering in 1986; and the PhD degree in automatic control and computer science engineering in 1993, both from Politecnico di Milano, Italy. He currently works at the Cesena site of the University of Bologna. His current research interests are in the fields of decision support systems (distributed architectures, MCDA, GIS integration) and of combinatorial optimization (evolutionary heuristic algorithms, LP-based techniques).

Alberto Colorni received the Laurea (masters of technology) degree in electronic engineering in 1970 from Politecnico di Milano, Italy. He has been a professor of operations research at the Politecnico di Milano since 1982, director of the research center on Decision Support Systems for Environment and Land Use at the MIP (Master Imprese-Politecnico) since 1991, and director of the METID (Innovative Methods and Technologies for Didactics) center since 1995. He currently works in the fields of combinatorial optimization (in particular, on heuristics from nature), DSS, and multicriteria methodologies, with applications to environmental impact assessment and innovative transportation systems.