Expert Systems with Applications 37 (2010) 5443–5453

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Analysis of the efficacy of a Two-Stage methodology for ant colony optimization: Case of study with TSP and QAP

Amilkar Puris [a,*], Rafael Bello [a], Francisco Herrera [b]

[a] Department of Computer Science, University of Las Villas, Cuba
[b] Department of Computer Science and Artificial Intelligence, University of Granada, Spain

ARTICLE INFO

Keywords:
Ant colony optimization
Traveling salesman problem
Assignation quadratic problem
Two-Stage strategy

ABSTRACT

Ant Colony Optimization (ACO) is a bioinspired metaheuristic based on ants foraging used to solve different classes of problems. In this paper, we show how, using a Two-Stage approach the quality of the solutions of ACO is improved. The Two-Stage approach can be applied to different ACO. The performance of this new approach is studied in the Traveling Salesman Problem and Quadratic Assignment Problem. The experimental results show that the obtained solutions are improved both problems using the Two-Stage approach. Several statistical procedures are applied to show the effect of this new approach.

1. Introduction

Ant Colony Optimization (ACO) is a metaheuristic (Dorigo, Caro, & Gambardella, 1999; Dorigo & Stützle, 2004) inspired by the shortest path searching behavior of various ant species. Since the initial work of Dorigo, Maniezzo and Colorni on the first ACO algorithm, the ant system (Dorigo, Colorni, & Maniezzo, 1996), several researchers have developed different ACO algorithms that performed properly when solving combinatorial problems such as the traveling salesman problem (TSP), the quadratic assignment problem (QAP), the sequential ordering problem, production scheduling, timetabling, project scheduling, vehicle routing, telecommunication routing, investment planning and staff scheduling, among others (Cordón, Herrera, & Stützle, 2002b; Dorigo & Stützle, 2003).

The ACO metaheuristics is a completely constructive model, the final solution is the resultant of a set of components, which are incorporated in every movement of the ants. This characteristic makes it necessary to evaluate the quality of every component before incorporating it to the solution, so that for big-size problems the model is rather inefficient regarding the run time. To suppress this drawback, a new approach of ACO was studied in Puris, Bello, Martinez, and Nowe (2007a); Puris, Bello, Trujillo, Nowe, and Martinez (2007b), named Two-Stage Ant Colony Optimization (TS-ACO). The underlying idea is to split the heuristic search performed by ants into two stages. In the first stage, preliminary results are found, these results are used by ants to improve the searching in the second stage.

In this paper the performance of an alternative of TS-ACO is discussed, in which the second stage only used the pheromone trail obtained in the first stage, called Pheromone Two-Stage ACO (PTS-ACO). The performance of this alternative is tested using as benchmark two well known problems, TSP (Gutin & Abraham, 2002) and QAP (Cela, 1998), where the run time is fixed to measure the quality solution for the both models (ACO and PTS-ACO). The Two-Stage approach is applied to different ACO algorithms, such as Ant System (AS), Ant Colony Optimization (ACS), and Max–Min Ant System (MMAS). The experimental results are shown and validated using statistical procedures.

In order to do that, the paper is organized as follows. In the next Section the ACO metaheuristic is described. Then, the two-stage ACO approach is presented in Section 3. In Section 4, the experimental framework is introduced, presenting the benchmarks and the statistical methodology. The study of experiments is showed in Section 5; first, the best classic ACO algorithm for each problem is chosen, later, the two-stage alternative is applied to each ACO algorithm selected and last a comparison between both models is presented. Concluding remarks are pointed out in Section 6. Finally, three appendix are included; statistical procedures used in order to compare the obtained results in this paper are explained in Appendix A, the results of the ACO algorithms for the benchmarks are presented in Appendix B and the results of the two-stage alternative applied to the best ACO algorithms for the benchmarks are presented in Appendix B.

2. Ant colony optimization

ACO (Dorigo & Stützle, 2004) algorithms are inspired from the foraging of real ant colonies to solve combinatorial optimization

* Corresponding author.
E-mail addresses: ayudier@uclv.edu.cu (A. Puris), rbellop@uclv.edu.cu (R. Bello), herrera@decsai.ugr.es (F. Herrera).

0957-4174/$ - see front matter © 2010 Published by Elsevier Ltd.
doi:10.1016/j.eswa.2010.02.069

problems. The algorithms are based on a colony of artificial ants, that is, simple computational agents that work cooperatively and that are communicated through artificial pheromone trails.

ACO algorithms are essentially constructive algorithms: in each algorithm iteration, every ant builds a solution to the problem by traveling on a construction graph. Each edge of the graph, representing the possible steps the ant can make, has associated two kinds of information that guide the ant movement:

- *Heuristic information*, which measures the heuristic preference of moving from node i to node j, of traveling the edge a_{ij}. It is denoted by η_{ij}. This information is not modified by the ants during the algorithm running.
- (Artificial) *pheromone trail information*, which measures the "learned desirability" of the movement and mimics the real pheromone that natural ants deposit. This information is modified during the algorithm's execution depending on the solutions found by the ants. It is denoted by τ_{ij}.

Finally, a local search could be applied (optional) to the solutions found by the ants. The effect of the local search is positive due to we find a positive synergy between ants, based exploration and local search exploration (Dorigo et al., 1996; Dorigo & Gambardella, 1997b).

Several ACO algorithms have been proposed which are included within the ACO metaheuristic, such as the Ant System (Dorigo et al., 1996), the Ant Colony System (Dorigo & Gambardella, 1997b), the Max–Min Ant System (Stützle & Hoos, 2000), the Rank-Based Ant System (Bullnheimer, Hartl, & Strauss, 1999), the Best–Worst Ant System (Cordón, de Viana, & Herrera, 2002a) and multi-objective ACO model (García-Martínez, Cordón, & Herrera, 2007), among others.

The next subsections will be briefly introduce the Ant System, Ant Colony System and Max–Min Ant System.

2.1. Ant system

The AS (Dorigo et al., 1996) was the first ACO algorithm. AS is characterized by the fact that the pheromone update is triggered once all ants have completed their solutions and it is done as follows. First, all pheromone trails are reduced by a constant factor, implementing in this way the pheromone evaporation. Second, every ant of the colony deposits an amount of pheromone in its path which is a function of the quality of its solution. Solutions in AS are built as follows. At each construction step, an ant k in AS chooses to go to a next node with a probability that is computed as:

$$p_{ij}^k = \frac{(\tau_{ij})^\alpha \cdot (\eta_{ij})^\beta}{\sum_{l \in N_i^k} (\tau_{ij})^\alpha \cdot (\eta_{ij})^\beta} \tag{1}$$

where η_{ij} is a priori available heuristic information for the transition from state i to j, α and β are two parameters which determine the relative influence of pheromone trail and heuristic information, and N_i^k is the feasible neighborhood of ant k, that is, the set of states which ant k has not yet visited. Parameters α and β have the following influence on the algorithm behavior. If $\alpha = 0$, the selection probabilities are proportional to $(\eta_{ij})^\beta$ and the closest states will more likely be selected, in this case AS corresponds to a classical stochastic greedy algorithm (with multiple starting points since ants are initially randomly distributed on the states). If $\beta = 0$, only pheromone amplification is at work; this will lead to the rapid emergence of a stagnation situation with the corresponding generation of tours which, in general, are strongly suboptimal (Dorigo et al., 1999). Search stagnation is defined in Dorigo, Bonabeau, and Theraulaz (2000) as the situation where all the ants follow the same path and construct the same solution.

The solution construction ends after each ant has completed a tour, that is, after each ant has constructed a sequence of length n. Next, the pheromone trails are updated. In AS this is done by first reducing the pheromone trails by a constant factor (this is pheromone evaporation) and then allowing each ant to deposit pheromone on the arcs that belong to its tour:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k \quad \forall (i,j) \tag{2}$$

where $0 < \rho \leqslant 1$ is the pheromone trail evaporation rate and m is the number of ants. The parameter ρ is used to avoid unlimited accumulation of the pheromone trails and enables the algorithm to "forge" previously done bad decisions. On arcs which are not chosen by the ants, the associated pheromone strength will decrease exponentially with the number of iterations. $\Delta\tau_{ij}^k$ is the amount of pheromone ant k deposits on the arcs; it is defined as:

$$\Delta\tau_{ij}^k = \begin{cases} 1/L^k(t) & \text{if} \quad \text{arc}(i,j) \text{ is used by ant } k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $L^k(t)$ is the length of the kth ant's tour. By Eq. (3), the shorter the ant's tour is, the more pheromone is received by arcs belonging to the tour. In general, arcs which are used by many ants and which are contained in shorter tours will receive more pheromone and therefore are also more likely to be chosen in future iterations of the algorithm.

2.2. Ant colony system

Other algorithm in ACO metaheuristic is the ACS (Dorigo & Gambardella, 1997b), it improves over AS by increasing the importance of exploitation of information collected by previous ants with respect to exploration of the search space. This is achieved via three mechanisms:

1. A strong elitist strategy is used to update pheromone trails:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \tau_{ij}^{best} \tag{4}$$

   the *best* ant is the global-best ant, that is, the ant that made the best tour from the start of the trial.
2. The ants choose the next node to move to using a so-called *pseudo-random proportional* rule: with probability q_0 they move to the node j for which the product between pheromone trail and heuristic information is maximum, that is:

$$j = \arg \max_{l \in N_i^k} \{(\tau_{ij})^\alpha \cdot (\eta_i j)^\beta\} \quad \text{if } q \leqslant q_0 \tag{5}$$

   while with probability $1 - q_0$ they operate a biased exploration in which the probability is the same as in AS (see Eq. 1). The value q_0 is a parameter: when it is set to a value close to 1, as it is the case in most ACS applications, exploitation is favored over exploration. Obviously, when q_0 the probabilistic decision rule becomes the same as in AS.
3. Differs form previous ACO algorithms also because ants update the pheromone trails while building solutions (like it was done in ant-quantity and in ant-density). In practice ACS ants "eat" some of the pheromone trail on the edges they visit.

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \tau_{ij}(0) \tag{6}$$

This has the effect of decreasing the probability that a same path is used by all the ants (i.e., it favors exploration, counterbalancing this

way the other two above-mentioned modifications that strongly favor exploitation of the collected knowledge about the problem).

2.3. Min–min ant system

The MMAS (Pitakaso, Almeder, Doerner, & Hartl, 2007; Stützle & Hoos, 2000), has been specifically developed to obtain a stronger exploitation of the solutions and to improve of premature stagnation of the search. To achieve this, the MMAS presents the three key aspects:

(i) To exploit the best solutions found during the execution of the algorithm after each iteration only one single ant adds pheromone. This ant may be the one which found the best solution in the current iteration (*iteration-best ant*) or the one which found the best solution from the beginning of the trial (*global-best ant*).
(ii) To avoid stagnation of the search, the range of possible pheromone trails on each solution component is limited to an interval $[\tau_{min}; \tau_{max}]$.
(iii) Additionally, is initialized the pheromone trails to τ_{max}, achieving in this way a higher exploration of solutions at the beginning of the algorithm.

In the MMAS the ants move from the state i to j, by using the same probabilistic rule that the AS algorithm.

After all ants have completed the tour construction, the pheromone trails are updated, in MMAS only a single ant is used to update the pheromone trails after each iteration. Consequently, also like in ACS, MMAS uses Eq. (4), where only the best ant (*the global-best* or *the iteration-best*) is allowed to add pheromone after each algorithm iteration. Computational results have shown that best results are obtained when pheromone updates are performed using the *iteration-best* solution with increasing frequency during the algorithm execution.

Independently of the choice between *the iteration-best* and *the global-best* ant for the pheromone trail update, search stagnation may occur. This can happen if at each choice point, the pheromone trail is significantly higher for one choice than for all the others. In this situation, due to the probabilistic choice governed by Eq. (1), the ant will prefer this solution component over all alternatives and further reinforcement will be given to the solution component in the pheromone trail update. In such situation the ants construct the same solution over and over again and the exploration of the search space stops.

For this reason, MMAS imposes explicit limits τ_{min} and τ_{max} on the minimum and maximum pheromone trails such that for all pheromone trails $\tau_{ij}(t)$. $\tau_{min} \leqslant \tau_{ij}(t) \leqslant \tau_{max}$. After each iteration must be ensured that the pheromone trail respects the limits. If we have $\tau_{ij}(t) > \tau_{max}$, we set $\tau_{ij}(t) = \tau_{max}$; analogously, if $\tau_{ij}(t) < \tau_{min}$, we set $\tau_{ij}(t) = \tau_{min}$. Also, note that by enforcing $\tau_{min} > 0$ and if $\tau_{ij} <$ for all solution components, the probability of choosing a specific solution component is never 0.

But still appropriate values for the pheromone trail limits must be chosen. In Stützle and Hoos (2000) different forms of determining these values are proposed, as well as other important elements for a better behavior of the algorithm; for instance, forms to determine the initial values of pheromone, also an additional mechanism, called *pheromone trail smoothing*, may be useful to increase MMAS performance.

3. Two-stage ant colony optimization

The behavior of ants in an ACO algorithm can be informally summarized as follows. A colony of ants concurrently and asynchronously are moved through adjacent states of the problem by building paths on a graph (Dorigo & Stützle, 2004). They walk following pheromone trails and heuristic information from a stochastic local decision policy. The ants incrementally build solutions that solve an optimization problem. Once the ant has built a solution, or while the solution is being built, the ant evaluates the (partial) solution and deposits pheromone trails on the components or connections it used. This pheromone information will lead the search of the next ants. This exploration strategy causes that the ACO algorithms develop an expensive exploration of the space search defined for a discrete problem.

TS-ACO is a proposal in order to obtain good exploration of the search space. The first studies about TS-ACO were presented in Puris et al. (2007a); Puris et al. (2007b). This strategy is bases in "Divide and Conquer" methodology . The exploration carried out by the ants is divided into two stages. The first stage is responsible to obtain partial results, which are used to exploration in the second stage. The split process affects to the number of ants (m) per stage, number units for solution (n) and stop condition per stage.

A ratio r is introduced in order to calculate the portion of the overall search to be performed in each stage. For instance, if $r = 0.3$, it means that the first stage will comprise 30% of the overall search and during the second stage, the 70% shall be carried out. The components are divided the following way:

1. (*Ants*) The total number of ants (m) defined by ACO algorithms is splits into two sets (m_1 *and* m_2). The m_1 set of ants develops the exploration in the first stage and m_2 for the second stage. These sets are calculated as:

$$m_1 = r \cdot m \tag{7}$$
$$m_2 = m - m_1 \tag{8}$$

2. (*Solutions*) The number of total components of the graph (n) is splits into two parts (n_1 *and* n_2), n_1 represents the size of the partial solution found to ants in the first stage and n_2 represents the size of the solution found in the second stage. These values are calculated as:

$$n_1 = r \cdot n \tag{9}$$
$$n_2 = n - n_1 \tag{10}$$
$$n_2 = n \tag{11}$$

In the second stage, the solutions can be built the two way:
(a) Adding to each one of the bests partial solution found the first stage, n_2 components Eq. (10).
(b) Explorer all graph starting form one initial node and adding n_2 components Eq. (11)

3. (*Stop condition*) The stop condition defined for ACO algorithms (usually run time, called sc) is splits into two values (sc_1 *and* sc_2), the first (sc_1) represents the stop condition of the ACO algorithm for the first stage and the another (sc_2) is the stop condition for the second. These values are calculated as:

$$sc_1 = r \cdot sc \tag{12}$$
$$sc_2 = sc - sc_1 \tag{13}$$

As output of the first stage, the bests partial solutions and the pheromone trails are obtained (partial results). Depending on the input of the second stage, different alternatives are presented. In previous works (Puris et al., 2007a; Puris et al., 2007b) both partial results are used for the ants to guide the search in second stage.

Now, the two-stage alternative presented in this paper, PTS-ACO, is describe. The partial results obtained as output of the first stage, only the pheromone trails are used in the second stage in or-

Step 1 Defined input parameters: ratio (r), number of ants (m), number of elements of the solution (n) and run time (sc)
Step 2 First stage
  2.1 Calculate the parameters for this stage (see Eqs. 7, 9 and 12)
  2.2 Apply ACO algorithm with the parameter values for this stage
  2.3 Output (the pheromone trails)
Step 3 Second stage
  3.1 Calculate the parameters for the second stage
  3.2 Calculate the parameters for this stage (see Eqs. 8, 11 and 13)
  3.3 Apply the ACO algorithm with the parameter values for this stage (Input: the pheromone trails obtained in the previous stage)
  3.4 Output (the best solution)
Step 4 End process

Fig. 1. PTS-ACO algorithm.

der to perform a good search process in this stage. This process is described below:

[*First stage:*] The number of ants assigned for the first stage (m_1) starts the search for a component of the graph (same ACO algorithms). Then, a partial solution of size (n_1) is found by each ant. In this process, the pheromone trails are updated depending of the ACO algorithm used. The search process stops when the run time for this stage (sc_1) is over.
[*Second stage:*] Each ant assigned for this stage (m_2) starts in the same way than the previous stage (in a component of the graph). In this stage, each ant finds a solution with n_2 components (Eq. 11) The pheromone trails used to initialize the search in this stage have been obtained obtained in the previous stage. Also, pheromone trails are updated in this stage depending of the ACO algorithm used. The search stops when the run time for this stage (sc_2) is over.

In the ACO algorithms, the cooperation among ants is the basis of its good behavior. This cooperation is obtained by modified pheromone trails; high pheromone values are associated with good path. In PTS-ACO the cooperation occurs into each stage; because at both stage, an ACO algorithm is used. Furthermore, another cooperation between stages is introduced, because the pheromone trails obtained in the first stage are used in the second stage. This pheromone trails contain the information of the search in the first stage, where high values are associated with the best partial solutions found. Algorithm Fig. 1 shows the associated algorithm.

4. Experimental framework

In order to develop the comparative study between ACO and PTS-ACO strategies, in this Section we present two combinatorial problems, as experimental frame. Next, we present the statistical methodology used to validate the obtained results.

4.1. Traveling salesman problem

The TSP (Gutin & Abraham, 2002) can be represented by a complete graph G = (N, A) with N being the set of nodes, also called cities, and A being the set of arcs fully connecting the nodes. Each arc $(i,j) \in A$ is assigned a value d_{ij} which represents the distance between cities i and j. The TSP is the problem of finding a shortest closed tour visiting each of the $n = |N|$ nodes of G exactly once. For symmetric TSPs, the distances between the cities are independent of the direction of traversing the arcs, that is, $d_{ij} = d_{ji}$ for every pair of nodes. In the asymmetric TSP at least for one pair of nodes (i, j) we have $d_{ij} \neq d_{ji}$. All the TSP instances used in the empirical studies presented in this paper are taken from the TSPLIB (Reinelt, 1991) benchmark library.[1] These instances have been used in many other studies and partly stem from practical applications of the TSP. Structure of the symmetric TSP instance is shown in Fig. 2, where the first row represents the number of cities in this instance. Elements of the matrix represent the distance between cities i and j (d_{ij}).

```
9
0 23 54 23 56 13 76 34 36
23 0 34 55 63 78 22 44 90
54 34 0 76 45 75 23 53 66
23 55 76 0 86 56 23 53 48
56 63 45 86 0 82 52 75 39
13 78 75 56 82 0 74 34 67
76 22 23 23 52 74 0 39 47
34 44 53 53 75 34 39 0 63
36 90 66 48 39 67 47 63 0
```

Fig. 2. TSP instance for 9 cities.

The description of the TSP instance for the experimental analysis is shown in Table 1, where the column 1 represents the instance name. The next column, the number of cities that present the instance, and the last column, represent the maximum run time for ACO and PTS-ACO algorithms, on each instance. This time is represented on second.

4.2. Quadratic assignment problem

The QAP (Cela, 1998) is the problem of assigning a set of facilities to a set of locations with given distances between the locations and given flows between the facilities. The goal is to place the facilities on locations in such a way that the sum of the products between flows and distances is minimal. Given n facilities and n locations, two $n \times n$ matrixes $A[a_{ij}]$ and $B[b_{rs}]$, where a_{ij} is the distance between locations i and j, and b_{rs} is the flow between facilities r and s, the QAP is the problem to minimize:

$$f(\phi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\phi(i)\phi(j)} \tag{14}$$

where ϕ is an arbitrary permutation of the set of integers 1,...,n (corresponding to an assignment of facilities to locations), and $\phi(i)$ gives the location of facility i in ϕ. Intuitively, $a_{ij}b_{\phi(i)\phi(j)}$ represents the cost contribution of simultaneously assigning facility i to location $\phi(i)$ and facility j to location $\phi(j)$. The QAP is an NP-hard optimization problem and it is considered to be one of the hardest optimization problems. To date, instances of size $n \leqslant 20$ can generally not be solved to optimality and heuristic algorithms must be applied, which are able to find very high quality solutions in a relatively short computation time. The instances on which we will test the algorithms proposed in this paper are taken from the QAPLIB (Burkard, Karisch, & Rendl, 1991) benchmark library[2].

For the QAP it is known that there are several different types of instances and that the particular instance type has a considerable influence on the performance of heuristic methods. According to, the instances of QAPLIB which we use in this paper can be classified into the following four classes.

(i) *Unstructured, randomly generated instances*. Instances with the distance and flow matrix entries generated randomly according to a uniform distribution. These instances are among the hardest to solve exactly. Nevertheless, most iterative search methods find solutions within 1–2% from the best known solutions relatively fast.
(ii) *Grid-based distance matrix*. In this class of instances the distance matrix stems from an $n_1 \times n_2$ grid and the distances

[1] Accessible at http://www.iwr.uniheidelberg.de/iwr/comopt/software/TSPLIB95
[2] Accessible at http://serv1.imm.dtu.dk/sk/qaplib/

Table 1
Summary description for TSP instances.

Instance name	Instance size	Time
berlin52.tsp	52	0.65
st70.tsp	70	2
rd100.tsp	100	4
ch150.tsp	150	11
kroA200.tsp	200	26
tsp225.tsp	225	37
a280.tsp	280	68
lin318.tsp	318	98
pcb442.tsp	442	241
rat575.tsp	575	516
rat783.tsp	783	1273

are defined as the Manhattan distance between grid points. These instances have multiple global optima (at least 4 if $n_1 \neq n_2$ and at least 8 if $n_1 = n_2$) due to the definition of the distance matrixes.

(iii) *Real-life instances*. Instances from this class are instances from practical applications of the QAP. Real-life instances have in common that the flow matrixes have many zero entries and the remaining entries are clearly not uniformly distributed.

(iv) *Real-life like instances*. Since the real-life instances in QAPLIB are of a rather small size, a particular type of randomly generated problems has been proposed in. These instances are generated in such a way that the matrix entries resemble the distributions found for real-life problems.

Fig. 3 represents an instance of the QAP and has the following structure; the first row determines the number of rows and columns of locations and facilities matrixes. These matrices are symmetrical and appear consecutively in Fig. 3. A detailed analysis of the search space generated by this problem has been presented in, which used local search algorithms with a fixed number of iterations to determine the characteristics of that space.

Table 2 shows the QAP instances used to the experiments. The first column represents the instance name's, the columns second and third shown the facilities and localizations quantity of each instance, and in the next column the maximum time for each instance is showed. This time is represented in seconds and was calculated depending on the instance size. For instance sizes lower than 50, the exploration time is defined as the instance's size times 10 s while those instances having a size greater than or equal to 50 are assigned a exploration time computed in the similar way to the previous case but multiplied by 20.

4.3. Statistical methodology for comparison

When a new optimization algorithm proposal is developed, it is necessary to compare it with previous approaches. Statistical analysis needs to be carried out in order to find significant differences among the results obtained by the studied methods. In García, Fernández, Luengo, and Herrera (2009a); García, Molina, Lozano, and Herrera (2009b); Luengo, García, and Herrera (2009) are presented the use of some non-parametric tests for comparing the results in Computational Intelligent.

```
10
 0 71 11 29 82 82  6 71  8 77    0 21 95 82 56 41  6 25 10  4
21  0 63  6 44 40 75 79  0 89   71  0 74 30 89 76 76 40 93 56
95 63  0 35  9  1 85 84 12  0   11 74  0  1 50  4 36 27 85  2
82  6 35  0 26 91 11 35 82 26   29 30  1  0  1 15 11 35 11 20
56 44  9 26  0 69 56 86 45 91   82 89 50  1  0 21 61 80 58 21
41 40  1 91 69  0 59 18 76 39   82 76  4 15 21  0 76 72 44 85
 6 75 85 11 56 59  0 18 57 36   6 76 36 11 61 76  0 94 90 51
25 79 84 35 86 18 18  0 61 36   71 40 27 35 80 72 94  0  3 48
10  0 12 82 45 76 57 61  0 21   8 93 85 11 58 44 90  3  0 29
 4 89  0 26 91 39 36 36 21  0   77 56  2 20 21 85 51 48 29  0
```

Fig. 3. Structure of the tai10a instance.

Table 2
Summary description for QAP instances.

Instance name	Number facilities	Number localizations	Time
Tai20a	20	20	20
Tai25a	25	25	25
Tai30a	30	30	30
Tai35a	35	35	35
Tai40a	40	40	40
Tai50a	50	50	100
Tai60a	60	60	120
Tai80a	80	80	160
Sko42	42	42	42
Sko49	49	49	49
Sko56	56	56	112
Sko64	64	64	128
Sko72	72	72	144
Sko81	81	81	162
Sko90	90	90	180
Tai20b	20	20	20
Tai25b	25	25	25
Tai30b	30	30	30
Tai35b	35	35	35
Tai40b	40	40	40
Tai50b	50	50	100
Tai60b	60	60	120
Tai80b	80	80	160
Kra30a	30	30	30
Kra30b	30	30	30
Ste36a	36	36	36
Ste36b	36	36	36

For pair wise comparison we will use the Wilcoxon Signed-Ranks Test (Demšar, 2006; Sheskin, 2006; Wilcoxon, 1945), and for multiple comparison we will employ different approaches, including the Friedman test (Friedman, 1937), the Iman and Davenport test (Iman & Davenport, 1980) and the Holm method (Holm, 1979). We will use in all cases $\alpha = 0.1$ as level of confidence. A wider description of these tests is presented in the Appendix A. This experimental frame will be use for study comparison of the results presents in Section 5.

5. Experimental study

In this Section we present the experimental study developed for this paper. In Section 5.1 contains the parameters configuration used for all algorithms. In Section 5.2, the study to select the best ACO algorithm among AS, ACS and MMAS for both problems is shown. In Section 5.3 two-stage strategy is applied for each ACO algorithm selected in previous subsection with different r values are studied in. Finally, Section 5.4 the comparative study between exploration strategies are shown.

5.1. Parameters configuration

The parameter setting used for AS, ACS and MMAS algorithms is shown in Table 3. The first column represents the algorithms name, the m column determines the number of ants used for each algorithm, $\tau(0)$ represents the initial value of pheromone trail, for the MMAS this value is the maximum value calculated for the first iteration (see Eq. (15)). The constant of evaporation is showed in column ρ. The columns α and β represent the transition ruler parameters (see Eq. (1)) and the q_0 column is used for the ACS algorithm (see Eq. (5)). The column local search represents at what moment the 2-opt (Dorigo & Gambardella, 1997a; Gambardella, Taillard, & Agazzi, 1999) algorithm is executed; the "best of cycle" represents that only the best ant in each iteration executes the lo-

Table 3
Parameters setting for experimental studies.

Algorithm	m	$\tau(0)$	ρ	α	β	q_0	Local search (2-opt)	Stop condition
Traveling Salesman Problem								
AS	10	0.2	0.1	2	3	–	best of cycle	time
ACS	10	0.2	0.1	1	5	0.67	best of cycle	time
MMAS	10	τ_{max}	0.1	2	3	–	best of cycle	time
Quadratic Assignment Problem								
AS	10	0.2	0.8	1	–	–	all ants	time
ACS	10	0.2	0.8	1	–	0.6	all ants	time
MMAS	10	τ_{max}	0.8	1	–	–	all ants	time

cal search and "all ants" represents that all ants execute the local search. The last column represents the stop condition, it which is defined by run time (see Tables 1 and 2).

The maximum and minimum value of pheromone for the MMAS algorithm are calculate according to:

$$\tau_{max} = (1 - /(1-\rho)) \cdot (1 - f(\phi_{gb})) \quad (15)$$

where $f(\phi_{gb})$ is the quality of the best solution, so far whereas the minimum values of the pheromone are computed as in expression (16)

$$\tau_{min} = \tau_{max}/10 \cdot n \quad (16)$$

where n is the size of the problem instance.

After presenting the parameters configuration, we will carry out the study of each model.

5.2. Experimental analysis for ACO algorithms

The purpose of this Subsection is to study the performance of the selected algorithms (AS, ACS, MMAS) in each problem (TSP, QAP), in order to select the best on for apply the two-stage strategy.

Two ACO algorithms applied to TSP have been studied recently in Wu, Zhao, Ren, and Quan (2009a, 2009b). Each ant is initially put on a randomly chosen city and has a memory which stores the partial solution it has constructed so far (initially the memory contains only the start city). Starting from its initial city, an ant iteratively moves from city to city. When being at a city i, an ant k chooses to go to a still unvisited city j with a probability given by Eq. (1) or (5), in this case $\eta_{ij} = 1/d_{ij}$. The solution construction ends after each ant has completed a tour. Next, the local search 2-opt [23] is applied, only for the best local solution. Then, the pheromone trails are updated, depending of ACO algorithm used (see Eq. (2) or (4),

The ACO application to the TSP can be extended to the QAP in a straightforward way (Gambardella et al., 1999,Tseng & Liang, 2006). The main difference lies in the definition of the solution components which for the QAP are given by the assignments of facilities to locations. Hence, the pheromone trails $\tau_{ij}(t)$ in the QAP application correspond to the desirability of assigning a facility i to a location j. For the solution construction, it can be convenient to use a pre ordering of the facilities (or, equivalently, the locations) and assign facilities in the given order. The decision points are related to the assignments: at each decision point an ant probabilistically decides on which location the next facility should be put. In ACO for the QAP, these decisions are done according to Eq. (1) or (5) using a QAP specific heuristic information. In this case the feasible neighborhood N_i^k of ant k comprises those locations which are still free. The single construction steps are repeated until a complete assignment is obtained. The pheromone update is done as in the TSP application.

The complete tables of results obtained for ACO algorithms are shown in Appendix B. In order to compare these results we have used a multiple comparison test to select the best algorithm for each problem. In Table 4, the results of applying Friedman and Iman–Davenport tests are shown in order to check if there are differences in the results for each problem. We employ the X^2-distribution with 4 degrees of freedom for Friedman in both experiments, the F-distribution for Iman–Davenport with 2 and 22 degrees of freedom for a number of instances ($N_{ds} = 11$) in TSP and for QAP experiments the F-distribution with 2 and 52 degrees of freedom for $N_{ds} = 27$. We emphasize in boldface the highest value between the two values that are being compared, and as the smallest in all cases corresponds to the value given by the statistic, it informs us of the rejection of the null-hypothesis of equality of means, telling us of the existence of significant differences among the observed results in all instances.

Table 4
Results of the Friedman and Iman–Davenport tests for comparing performance of the ACO algorithms in two problems studied.

Method	Test value	Distribution value	p-Value
TSP			
Friedman	**16.666**	7.779	2.40E−4
Iman–Davenport	**25.000**	2.561	1.54E−4
QAP			
Friedman	**27.796**	7.779	9.20E−7
Iman–Davenport	**27.580**	2.407	6.82E−9

Table 5 shows the rankings (computed using a Friedman test) of the 3 algorithms considered for each problem. In order to emphasize the results of the rankings showed in Table 5, we depicted them in Fig. 4.

Now, we apply a Holm test to compare the best ranking method (ACS for TSP and MMAS for QAP) with the remaining ACO methods. The result of the test are shown in Table 6, in which the algorithms are ordered with respect to the z value obtained. Thus, by using the normal distribution, we can obtain the corresponding p-value associated with each comparison and this can be compared with the associated α/i (see Fig. 5).

Analyzing the results presented in Appendix B and the statistical study shown in this Subsection is concluded that:

1. For TSP, the best results are obtained by ACS algorithm. Table 6 shows that significant differences were detected by a Holm test between the control algorithm and other algorithms.

Table 5
Mean ranks of Friedman test for all problems.

Algorithm	Mean ranks	
	TSP	*QAP*
AS	2.8333	1.8703
ACS	*1.1666*	2.7592
MMAS	1.8000	*1.3703*

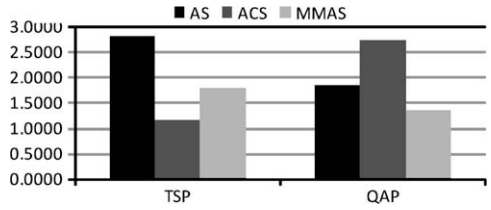

Fig. 4. Mean ranks for TSP and QAP problems by AS, ACS and MMAS algorithms.

Table 6
Statistical result by Holm test for TSP and QAP with $\alpha = 0.1$ (The control algorithm for TSP is ACS and MMAS for QAP).

Algorithm	z	p-Value	α/i	Hypothesis
TSP				
AS	4.0824	4.45E−5	0.05	Rejected for ACS
MMAS	2.0412	0.0412	0.1	Rejected for ACS
QAP				
ACS	5.1031	3.34E-7	0.05	Rejected for MMAS
AS	1.8372	0.0661	0.1	Rejected for MMAS

2. For QAP, the best results are obtained by MMAS algorithm. Table 6 shows that significant differences were detected in favor of the MMAS with respect to the remainder algorithms.

5.3. Experimental analysis for PTS-ACO

In this Section, we apply the pheromone two-stage alternative for each ACO algorithm selected previously. We describe now the two-stage alternative is applied for each ACO algorithm in the solutions of both problems.

In Appendix C, the experimental results for each r value and two problems are showed. Different r values were tested $r = \{0.2, 0.25, 0.3\}$ in order to measure the behavior of this parameter in the two-stage strategy. We will employ multiple comparison tests for the statistical study, using for this purpose Friedman, Iman–Davenport and Holm tests. As we did in the previous Section, we will compare the PTS-ACO for TSP and QAP.

I When using P-TS-ACS for TSP, the process begins calculating the dimension for each stage. We use the configuration of the ACS parameters shown in Table 3. For instance, using Dataset rd100.tsp, $r = 0.3$ and run time equal to 4 s; the process would be as follows:
   (a) In the first stage, the ACS algorithm looks for partial tour of 30 cities ($r \cdot 100$ (total number of cities)). Each ant finds a tour in different cycles. In this stage only 3 ants are selected ($r \cdot 10$ (total number of ants, see Table 3)). The pheromone is updated in this stage depending on the best global partial tours. This stage stops when the exploration time is over 1.2 s ($r \cdot 4$).
   (b) In the second stage, the ACS algorithm is executed by 7 ants ($10 - 3$). In this case, the ants will obtain the tours of all cities (100), that is, each ant finds a tour in different iterations. The pheromone trails used in this stage are obtained in the first stage. The ACS algorithm finishes when the exploration time is over 2.8 s ($4 - 1.2$).

II The second experimental analysis for QAP is done to MMAS algorithm using two-stage alternative. For instance, dataset tai20a, $r = 0.2$ and execute time equal to 4 s ; the process would be as follows:
   (a) In the first stage, the MMAS algorithm looks for partial assignations of 4 items ($r \cdot 20$ (total number of facilities or localization)). Each ant finds an assignation in different iterations. In this stage only 2 ants are selected ($r \cdot 10$ (total number of ants). The pheromone is updated in this stage depending on the best global partial assignations. This stage concludes when the exploration time is over 0.8 s ($r \cdot 4$ s).
   (b) In the second stage, the MMAS algorithm is executed by 8 ants ($10 - 2$), in this case, the ants will obtain the assignations of all items (20), that is, each ant finds an assignation in different iterations. The pheromone trails used in this stage are obtained in the first stage. The MMAS algorithm finishes when the exploration time is over 3.2 s ($4 - 0.8$).

Table 7
Description of the use algorithms.

P-TS-ACS(0.2)	Two-stage alternative applied ACS with $r = 0.2$
P-TS-ACS(0.25)	Two-stage alternative applied ACS with $r = 0.25$
P-TS-ACS(0.3)	Two-stage alternative applied ACS with $r = 0.3$
PTS-MMAS(0.2)	Two-stage alternative applied MMAS with $r = 0.2$
PTS-MMAS(0.25)	Two-stage alternative applied MMAS with $r = 0.25$
PTS-MMAS(0.3)	Two-stage alternative applied MMAS with $r = 0.3$

Table 7 shows the notation of the algorithms used for these study.

Analyzing the results presented in Appendix B and the statistical study shown in Tables 8–10 we are conclude that:

1. For TSP, significant difference among PTS-ACS with all r values have been detected for Friedman and Iman–Davenport tests (see Table 8). Then, for the PTS-ACS the best r value based on

Table 8
Results of the Friedman and Iman–Davenport tests for comparing performance of the PTS-ACO algorithms in two problems studied.

Method	Test value	Distribution value	p-Value
TSP			
Friedman	**5.3749**	7.7794	0.0680
Iman–Davenport	**3.1744**	2.5613	0.0614
QAP			
Friedman	**0.8888**	7.7794	0.8888
Iman–Davenport	**0.4351**	2.4076	0.6495

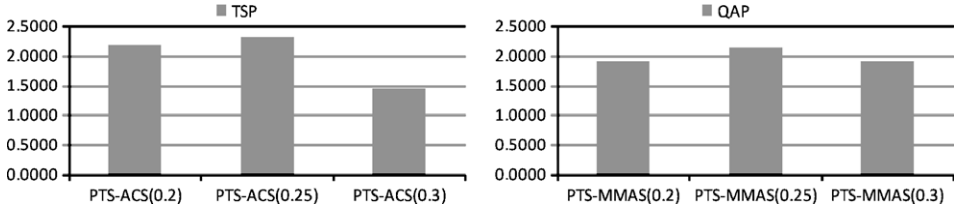

Fig. 5. Means ranks for TSP (left) and QAP (right).

Table 9
Mean ranks of Friedman test for all problems.

TSP		*QAP*	
Algorithm	Mean ranks	Algorithm	Mean ranks
P-TS-ACS(0.2)	2.2083	PTS-MMAS(0.2)	*1.9259*
P-TS-ACS(0.25)	2.3333	PTS-MMAS(0.25)	2.1481
P-TS-ACS(0.3)	*1.4583*	PTS-MMAS(0.3)	1.9260

Table 10
Statistical results by Holm test for TSP with $\alpha = 0.1$ (The control algorithm is PTS-ACS(0.3)).

Algorithm	z	p-Value	α/i	Hypothesis
TSP				
P-TS-ACS(0.25)	2.1433	0.0320	0.05	Rejected for PTS-ACS(0.3)
PTS-ACS(0.2)	1.8371	0.0661	0.1	Rejected for PTS-ACS(0.3)

rankings has been $r = 0.3$, this result is shown in Table 9. Finally Table 10 contains a Holm test, which shows that the PTS-ACS with $r = 0.3$ is better in performance than the remaining r values.

2. For QAP, the Friedman and Iman–Davenport tests do not detect significant differences among the r values. For these results, we select the algorithm that achieves the higher ranking. In this case, we chose PTS-MMAS with $r = 0.2$.

5.4. Comparison of algorithms

In the final part of our study, we will analyze the behavior of our two-stage alternative opposite ACO strategy in each problem. The Wilcoxon test is used to complete this study.

The main conclusion after this study is that the two-stage alternative for ACO algorithms is better than the strategy based on one stage. For TSP, the Wilcoxon (see Table 11) test found significant differences in favor to PTS-ACS with $r = 0.3$. The same results are obteined for QAP, where the null hypothesis is rejected in favor to PTS-MMAS with $r = 0.2$. In both cases, the it p-value is lower than significance value fixed to these experiments ($\alpha = 0.1$)

6. The concluding remarks

We have proposed a new two-stage alternative PTS-ACO for exploration with ACO algorithms. In this alternative, the pheromone trails obtained in the first stage are used in order to guide the exploration into the second stage of the search process.

We have compared the two-stage alternative with ACO model based in on one stage. For this comparison, we have selected the best classic ACO algorithm for each problem according to our results: ACS for TSP and MMAS for QAP.

The experimental results have shown that the use of the two-stage alternative to the ACO algorithms obtains better results than the original ACO model,existing a major cooperation between ACO agents with the new alternative.

Appendix A. On the use of non-parametric tests based on ranking

In this paper, we have made use of statistical techniques for the analysis the used methods, since they are a necessity in order to provide a correct empirical study (Demšar, 2006; García & Herrera, 2009). Specifically, we have employed non-parametric tests, due to the fact that the initial conditions that guarantee the reliability of the parametric tests may not be satisfied, making the statistical analysis to lose credibility (Demšar, 2006). In this Appendix, we describe the procedures for performing pairwise a multiple comparisons. Specifically, we have employed the Wilcoxon signed-rank test as non-parametric statistical procedure for performing pairwise comparisons between two algorithms. For multiple comparisons we have used the Friedman and Iman–Davenport tests to detect statistical differences and the Holm post-hoc test in order to find what algorithms partners' average results are dissimilar. Next, we will describe both approaches.

A.1. Wilcoxon signed-ranks test

This is the analogous of the paired t-test in non-parametric statistical procedures; therefore, it is a pairwise test that aims to detect significant differences between two sample means, that is, the behavior of two algorithms. Let d_i be the difference between the performance scores of the two algorithms on ith out of $N_d s$ data-sets. The differences are ranked according to their absolute values; average ranks are assigned in case of ties. Let R^+ be the sum of ranks for the data-sets on which the first algorithm outperformed the second, and R^- the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored:

$$R_+ = \sum_{d>0} rank(d_i) + \frac{1}{2}\sum_{d=0} rank(d_i) \tag{17}$$

$$R_- = \sum_{d<0} rank(d_i) + \frac{1}{2}\sum_{d=0} rank(d_i) \tag{18}$$

Let T be the smaller of the sums, $T = \min(R^+, R^-)$. If T is less than or equal to the value of the distribution of Wilcoxon for Nds degrees of freedom (Zar, 1998, Table B.12), the null-hypothesis of equality of means is rejected. Wilcoxon signed-ranks test is more sensible than the t-test. It assumes commensurability of differences, but only qualitatively: greater differences still count more, which is probably desired, but the absolute magnitudes are ignored. From the statistical point of view, the test is safer since it does not assume normal distributions. Also, the outliers (exceptionally good/bad performances on a few data-sets) have less effect on the Wilcoxon than on the t test. The Wilcoxon test assumes continuous differences d_i, therefore they should not be rounded to one or two decimals, since this would decrease the power of the test due to a high number of ties.

When the assumptions of the paired t-test are met, Wilcoxon signed-ranks test is less powerful than the paired t test. On the other hand, when the assumptions are violated, the Wilcoxon test can be even more powerful than the t test. This allows us to apply it

Table 11
Wilcoxon test to compare the PTS-ACO against ACO in two problems with $\alpha = 0.1$.

Algorithm	R^+	R^-	p-Value	Hipothesis
TSP				
PTS-ACS(0.3) vs ACS	64.00	14.00	0.05	Rejected for **PTS-ACS(0.3)**
QAP				
PTS-MMAS(0.2) vs MMAS	265.5	112.5	0.078	Rejected for **PTS-MMAS(0.2)**

over the means obtained by the algorithms in each data-set, without any assumptions about the sample of results obtained.

A.2. Friedman test and post-hoc tests

In order to perform a multiple comparison, it is necessary to check whether all the results obtained by the algorithms present any inequality. In the case of finding it, then we can know, by using a post-hoc test, what algorithms partners' average results are dissimilar. In the following, we describe the non-parametric tests used.

- The first one is the Friedman test (Sheskin, 2006), which is a non-parametric test equivalent to the repeated measures ANOVA. Under the null-hypothesis, it states that all the algorithms are equivalent, so a rejection of this hypothesis implies the existence of differences among the performance of all the algorithms studied. After this, a post-hoc test could be used in order to find whether the control or proposed algorithm presents statistical differences with regards to the remaining methods in the comparison. Of the more solid of them is the Holm (1979) test, the which reject more hypothesis that other tests. The working mode of the Friedman test is described as follows: It ranks the algorithms for each data-set separately, the best performing algorithm getting the rank of 1, the second best rank 2, and so on. In case of ties average ranks are assigned. Let r_{ji} be the rank of the jth of k algorithms on the ith of N_{ds} data-sets. The Friedman test compares the average ranks of algorithms, $R_j = \frac{1}{N_{ds}}\sum_i r_i^j$ Under the null-hypothesis, which states that all the algorithms are equivalent and so their ranks R_j should be equal, the Friedman statistic:

$$X_F^2 = \frac{12N_{ds}}{k(k+1)}\left[\sum jR_j^2 - \frac{k(k+1)^2}{4}\right] \tag{19}$$

is distributed according to X_F^2 with $k-1$ degrees of freedom, when N_{ds} and k are big enough.
- The second one of them is Iman and Davenport test (Iman & Davenport, 1980), which is a non-parametric test, derived from the Friedman test, less conservative than the Friedman statistic:

$$F_F = \frac{(N_{ds}-1)X_F^2}{N_{ds}(K-1)-X_F^2} \tag{20}$$

which is distributed according to the *F-distribution* with $k-1$ and $(k-1)(N_{ds}-1)$ degrees of freedom. Statistical tables for critical values can be found at (Zar, 1998).
- Holm test (Holm, 1979): it is a multiple comparison procedure that can work with a control algorithm and compares it with the remaining methods. The test statistics for comparing the ith and jth method using this procedure is:

$$z = (R_i - R_j)\Big/\sqrt{\frac{k(k+1)}{6N_{ds}}} \tag{21}$$

The z value is used to find the corresponding probability from the table of normal distribution, which is then compared with an appropriate level of confidence a. Holm test adjusts the value for a in order to compensate for multiple comparison.

The Holm test is a step-up procedure that sequentially tests the hypotheses ordered by their significance. We will denote the ordered p values by p_1, p_2, so that $p_1 \leqslant p_2 \leqslant p_{k-1}$. The Holm test compares each p_i with $\alpha/(k-i)$, starting from the most significant p-value. If p_1 is below $\alpha/(k-1)$, the corresponding hypothesis is rejected and we allow to compare p_2 with $\alpha/(k-2)$. If the second hypothesis is rejected, the test proceeds with the third, and so on. As soon as a certain null hypothesis cannot be rejected, all the remain hypotheses are retained as well.

The post-hoc procedure described above allow us to know whether or not a hypothesis of comparison of means could be rejected at a specified level of significance a. However, it is very interesting to compute the p value associated to each comparison,

Table 12
Description of the used algorithms.

Symbolic	Description
BK	The best know solution
AS	Results for Ant System algorithm
ACS	Results for Ant Colony System algorithm
MMAS	Results for Max–Min Ant System algorithm
Time	The run time for each algorithm

Table 13
Experimental results for TSP instances by ACO algorithms.

Instance	BK	AS	MMAS	ACS	Time
gr24.tsp	1272	*1275.64*	1278.04	1277.52	0.15
berlin52.tsp	7542	7746.16	7689.92	7729.16	0.65
st70.tsp	675	714.96	*703.12*	726.68	2
rd100.tsp	7910	8717.36	8694.24	*8659.53*	4
ch150.tsp	6528	7219.84	*6867.68*	6908.84	11
kroA200.tsp	29368	40269.76	32858.68	*32643.76*	26
tsp225.tsp	3919	4124.64	4095.8	*4039.4*	37
a280.tsp	2579	3493.16	3020.28	*2973.72*	68
lin318.tsp	42029	53175.84	47739.36	*47494.96*	98
pcb442.tsp	50778	60781.08	*59118.32*	59264.12	241
rat575.tsp	6773	9479.68	7795.44	*7717.48*	516
rat783.tsp	8806	12684.6	10199.16	*10038.87*	1273

Table 14
Experimental results for the QAP instances by ACO algorithms.

Instance	AS	MMAS	ACS	Time
Random problems with uniformly distributed matrix entries (i)				
Tai20a	0.01155	*0.00462*	0.00522	20
Tai25a	0.01419	*0.01058*	0.01712	25
Tai30a	0.01682	*0.01618*	0.02068	30
Tai35a	*0.01835*	0.02388	0.02205	35
Tai40a	0.02619	*0.02515*	0.02880	40
Tai50a	0.03234	*0.03105*	0.03158	100
Tai60a	0.03325	*0.03089*	0.03369	120
Tai80a	*0.02556*	0.02594	0.03238	160
Random flows on grids (ii)				
Sko42	0.01024	*0.00664*	0.01082	42
Sko49	0.01144	*0.01031*	0.01303	49
Sko56	0.01377	*0.01098*	0.01362	112
Sko64	*0.01344*	0.01350	0.01382	128
Sko72	*0.01367*	0.01423	0.01583	144
Sko81	0.01207	*0.01181*	0.01337	162
Sko90	*0.04370*	0.04433	0.04461	180
Real-life instances (iii)				
Tai20b	9.05E−04	*0.0*	*0.0*	20
Tai25b	*0.0*	*0.0*	0.00140	25
Tai30b	*4.41E−04*	0.00018	0.00238	30
Tai35b	0.00493	*0.00285*	0.00518	35
Tai40b	0.00916	*0.00423*	0.00672	40
Tai50b	0.00780	*0.00456*	0.00985	100
Tai60b	0.00785	*0.00411*	0.00814	120
Tai80b	0.02801	*0.02289*	0.03151	160
Randomly generated real-life like instances (iv)				
Kra30a	0.01394	*0.00713*	0.02464	30
Kra30b	0.00316	*0.00208*	0.00756	30
Ste36a	0.01870	*0.01444*	0.02777	36
Ste36b	*0.02152*	0.02210	0.05204	36

Table 15
Description of the used algorithms.

Symbolic	Description
PTS-ACS	Alternative of the Two-Stage exploration apply to the ACS
PTS-MMAS	Alternative of the Two-Stage exploration apply to the MMAS
Time	The run time for each algorithm

Table 16
Experimental results for TSP instances by ACO algorithms.

Instance	BK	PTS-ACS			Time
		$r = 0.2$	$r = 0.25$	$r = 0.3$	
gr24.tsp	1272	1273.96	1274.32	*1273.68*	0.15
berlin52.tsp	7542	7763.56	7717.76	*7647.4*	0.65
st70.tsp	675	712.8	712.36	*709.2*	2
rd100.tsp	7910	8580.96	8733.24	*8545.92*	4
ch150.tsp	6528	8580.96	8733.24	*8545.92*	11
kroA200.tsp	29368	32973.04	33242.04	*32775.2*	26
tsp225.tsp	3919	4077.2	4046.16	*4032.44*	37
a280.tsp	2579	2956.56	2970.92	*2948.44*	68
lin318.tsp	42029	*47365.16*	47556.48	47507.44	98
pcb442.tsp	50778	58610.88	58585.23	*58190.92*	241
rat575.tsp	6773	*7034.43*	7074.32	7063.56	516
rat783.tsp	8806	10123.96	10197.96	*10033.16*	1273

Table 17
Experimental results for the QAP instances by PTS-MMAS.

Instance	PTS-ACS			Time
	$r = 0.2$	$r = 0.25$	$r = 0.3$	
Random problems with uniformly distributed matrix entries (i)				
Tai20a	*0.0037*	0.0041	0.0054	20
Tai25a	*0.0109*	0.0121	0.0121	25
Tai30a	*0.0151*	0.0160	0.0165	30
Tai35a	0.0237	*0.0222*	0.0225	35
Tai40a	0.0268	*0.0242*	0.0253	40
Tai50a	0.0295	0.0288	*0.0277*	100
Tai60a	0.0308	0.0321	*0.0294*	120
Tai80a	0.0264	*0.0208*	0.0261	160
Random flows on grids (ii)				
Sko42	*0.0064*	0.0077	0.0074	42
Sko49	0.0087	0.0091	*0.0080*	49
Sko56	0.0082	*0.0080*	0.0092	112
Sko64	*0.0096*	0.0099	0.0108	128
Sko72	0.0115	0.0123	*0.0114*	144
Sko81	0.0124	*0.0109*	0.0121	162
Sko90	0.0443	0.0447	*0.0442*	180
Real-life instances (iii)				
Tai20b	*0.0*	*0.0*	*0.0*	20
Tai25b	*0.0*	*0.0*	*0.0*	25
Tai30b	1.7E−4	*3.8E−5*	5.3E−4	30
Tai35b	*0.0024*	*0.0024*	0.0027	35
Tai40b	0.0200	*0.0064*	0.0075	40
Tai50b	0.0054	0.0055	*0.0038*	100
Tai60b	0.0039	0.0040	*0.0038*	120
Tai80b	*0.0198*	0.0268	0.0202	160
Randomly generated real-life like instances (iv)				
Kra30a	*0.0037*	0.0083	0.0106	30
Kra30b	0.0025	0.0023	*0.0017*	30
Ste36a	*0.0144*	0.0226	0.0186	36
Ste36b	*0.0205*	0.0301	0.0214	36

which represents the lowest level of significance of a hypothesis that results in a rejection. In this manner, we can know whether two algorithms are significantly different and we can also have a metric of how different they are. Next, we will describe the method used for computing these exact p-Values for each test procedure, which are called "adjusted *p-values*" (Wright, 1992).

- The adjusted *p-value* for the Holm procedure is computed by $p_{Holm} = (k - i)p_i$. Once computed all of them for all hypotheses, it is not possible to find an adjusted *p-value* for the hypothesis i lower than for the hypothesis j, $j < i$. In this case, the adjusted *p-value* for hypothesis i is set to the same value as the one associated to hypothesis j.

Appendix B. Results for the TSP and QAP by ACO algorithms

All results that have been showed in this Appendix were computed over 25 independent runs. In the TSP case these results represent the average and the average percentage excess of the best solution for QAP.The first Table 12 shows the used symbolic for each algorithm presented in this appendix.

Best results are printed (see Tables 13 and 14) in italic typeface. The number in the name gives the instance dimension, that is, the number of cities for TSP and of facilities and localizations for QAP respectively.

Appendix C. Results for the TSP and QAP by PTS-ACO

All results that have been showed in this Appendix were computed over 25 independent runs. In the TSP case these results represent the average (see Table 16) and the average percentage excess of the best solution for QAP (see Table 17). The Table 15 shows the used symbolic for each algorithm presented in this appendix.

Best results are printed in italic typeface. The number in the name gives the instance dimension, that is, the number of cities for TSP and of facilities and localizations for QAP respectively.

References

Bullnheimer, B., Hartl, R., & Strauss, C. (1999). A new rank based version of the ant system: A computational study. *Central European Journal for Operations Research and Economics, 7*(1), 25–38.

Burkard, R., Karisch, S., & Rendl, F. (1991). QAPLIB-quadratic assignment problem library. *European Journal of Operational Research, 55*(1), 115–119.

García-Martínez, C., Cordón, O. O., & Herrera, F. (2007). A taxonomy and an empirical anlysis of multiple objective ant colony optimization algorithms for bi-criteria TSP. *European Journal of Operational Research, 180*(1), 433–452.

Cela, E. (1998). *The quadratic assignment problem: Theory and algorithms*. Kluwer Academic.

Cordón, O., de Viana, I. F., & Herrera, F. (2002a). Analysis of the best–worst ant system and its variants on the QAP. In *Ant Algorithms. ANT2002. Lecture Notes in Computer Science* (Vol. 2463, pp. 228–234). Springer.

Cordón, O., Herrera, F., & Stützle, T. (2002b). A review on the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware and Soft Computing, 9*(2-3), 141–175.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data-sets. *Journal of Machine Learning Research, 7*, 1–30.

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

Dorigo, M., Caro, G. D., & Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life, 5*(2), 137–172.

Dorigo, M., Colorni, A., & Maniezzo, V. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems and Cybernetics, 26*(1), 29–41.

Dorigo, M., & Gambardella, L. (1997a). Ant colonies for the travelling salesman problem. *Biosystems, 43*(2), 73–81.

Dorigo, M., & Gambardella, L. M. (1997b). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*(1), 53–66.

Dorigo, M., & Stützle, T. (2003). The ant colony optimization metaheuristic: Algorithms, applications, and advances. In *Handbook of Metaheuristics* (pp. 250–285). Kluwer Academic.

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. MIT Press.

Friedman, M. (1937). Individual comparisons by ranking methods. *Journal of the American Statistical Association, 32*, 675–701.

Gambardella, L. M., Taillard, E., & Agazzi, G. (1999). Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society, 50*, 167–176.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2009a). A study of statistical techniques and performance measures for genetics-based machine learning:

Accuracy and interpretability. *Soft Computing and Applications*. doi:10.1007/s00500-008-0392-y.

García, S., & Herrera, F. (2009). An extension on statistical comparisons of classifiers over multiple data-sets for all pairwise comparisons. *Journal of Machine Learning Research, 9*, 2677–2694.

García, S., Molina, D., Lozano, M., & Herrera, F. (2009b). A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: A case study on the cec2005 special session on real parameter optimization. *Journal of Heuristics*. doi:10.1007/s10732-008-9080-4.

Gutin, G., & Abraham, P. (2002). *The traveling salesman problem and its variations*. Kluwer Academic.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics, 6*(2), 65–70.

Iman, R. L., & Davenport, J. M. (1980). Approximations of the critical region of the friedman statistic. *Communications in Statistics, 9*(Parte A Theory Methods), 571–595.

Luengo, J., García, S., & Herrera, F. (2009). A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions and non-parametric tests. *Expert Systems with Applications, 36*, 7798–7808.

Pitakaso, R., Almeder, C., Doerner, K. F., & Hartl, R. F. (2007). A MAX–MIN ant system for unconstrained multi-level lot-sizing problems. *Computers and Operations Research, 34*(9), 2533–2552.

Puris, A., Bello, R., Martinez, Y., & Nowe, A. (2007a). Two-stage ant colony optimization for solving the traveling salesman problem. In J. Mira & J. R. lvarez (Eds.). *IWINAC. Lecture Notes in Computer Science* (Vol. 4528, pp. 307–316). Springer.

Puris, A., Bello, R., Trujillo, Y., Nowe, A., & Martinez, Y. (2007b). Two-stage aco to solve the job shop scheduling problem. In L. Rueda, D. Mery, & J. Kittler (Eds.). *CIARP. Lecture Notes in Computer Science* (Vol. 4756, pp. 447–456). Springer.

Reinelt, G. (1991). TSPLIB - a traveling salesman problem library. *ORSA Jornal on Computing, 3*, 376–384.

Sheskin, D. (2006). *Handbook of parametric and nonparametric statistical procedures* (2th ed.). Chapman and Hall/CRC.

Stützle, T., & Hoos, H. H. (2000). MAX–MIN ant system. *Future Generation Computer System, 16*(9), 889–914.

Tseng, Lin-Yu, & Liang, Shyi-Ching (2006). A hybrid metaheuristic for the quadratic assignment problem. *Computational Optimization and Applications, 34*(1), 85–113.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics, 1*, 80–83.

Wright, S. (1992). Adjusted p-values for simultaneous inference. *Biometrics, 48*, 1005–1013.

Wu, Z., Zhao, N., Ren, G., & Quan, T. (2009a). New robust and efficient ant colony algorithms: Using new interpretation of local updating process. *Expert Systems with Applications, 36*(1), 481–488.

Wu, Z., Zhao, N., Ren, G., & Quan, T. (2009b). Population declining ant colony optimization algorithm and its applications. *Expert Systems with Applications, 36*(3), 6276–6281.

Zar, J. H. (1998). *Biostatistical analysis* (4th ed.). Upper Saddle River, New Jersy: Prentice Hall.