An Empirical Analysis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-criteria TSP[⋆]

C. García-Martínez, O. Cordón, F. Herrera

Dept. of Computer Science and Artificial Intelligence
University of Granada. 18071 - Granada. Spain
gcarlos@fedro.ugr.es, {ocordon, herrera}@decsai.ugr.es

Abstract. The difficulty to solve multiple objective combinatorial optimization problems with traditional techniques has urged researchers to look for alternative, better performing approaches for them. Recently, several algorithms have been proposed which are based on the Ant Colony Optimization metaheuristic. In this contribution, the existing algorithms of this kind are reviewed and experimentally tested in several instances of the bi-objective traveling salesman problem, comparing their performance with that of two well-known multi-objective genetic algorithms.

1 Introduction

Multi-criteria optimization problems are characterized by the fact that several objectives have to be simultaneously optimized, thus making especially difficult their solving. The existence of many multi-objective problems in the real world, their intrinsic complexity and the advantages of metaheuristic procedures to deal with them has strongly developed this research area in the last few years [11].

Ant colony optimization (ACO) is a metaheuristic 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, several researchers have developed different ACO algorithms that performed succesfully when solving many different combinatorial problems [6].

Recently, some researchers have designed ACO algorithms to deal with multi-objective problems (MOACO algorithms) [1, 2, 4, 5, 7, 8, 10]. The most of them are specific proposals to solve a concrete multicriteria problem such as scheduling, vehicle routing, or portfolio selection, among others. The aim of the current contribution is to analyze the application of these proposals to the same benchmark problem, the multi-objective traveling salesman problem (TSP), and to compare their performance to two second generation, elitist multi-objective genetic algorithms (MOGAs) that currently represent the state-of-the-art of multi-objective evolutionary optimization: SPEA2 and NSGA-II [3].

⋆ This work was partially supported by the Spanish Ministerio de Ciencia y Tecnología under project TIC2003-00877 (including FEDER fundings) and under Network HEUR TIC2002-10866-E.

This paper is structured as follows. In Section 2, some basics of multi-objective optimization are reviewed. In Section 3, the existing MOACO algorithms are introduced, reporting their key characteristics. In Section 4, the performance of most of these algorithms is analyzed by applying them to the bi-objective TSP. Finally, some concluding remarks and proposals for future work are showed in Section 5.

2 Multi-objective Optimization

Multi-criteria optimization problems are characterized by the fact that several objectives have to by simultaneously optimized. Hence, there is not usually a single best solution solving the problem, but a set of solutions that are superior to the remainder when all the objectives are considered, the Pareto set. These solutions are known as Pareto-optimal or non-dominated solutions, while the remainder are known as dominated solutions. All of the former are equally acceptable as regards the satisfaction of all the objectives.

This way, the formal definition of the dominance concept is as follows. Let us consider, without loss of generality, a multiobjective minimization problem with m parameters (decision variables) and K objectives:

$$Min\ f(x) = (f_1(x), f_2(x), \ldots, f_K(x)),\ \ with\ x = (x_1, x_2, \ldots, x_m) \in X\ \ .$$

A decision vector $a \in X$ dominates another $b \in X$ ($a \succ b$) if, and only if:

$$\forall i \in 1, 2, \ldots, K \mid f_i(a) \leq f_i(b) \quad \wedge \quad \exists j \in 1, 2, , \ldots, K \mid f_j(a) < f_j(b)\ \ .$$

3 Multiple Objective Ant Colony Optimization Algorithms

In this section, the different existing proposals for Pareto-based MOACO algorithms that aim at obtaining set of non-dominated solutions for the problem being solved are reviewed.

3.1 Multiple Objective Ant-Q Algorithm

Multiple objective Ant-Q algorithm (MOAQ) is an MOACO algorithm that was proposed by Mariano and Morales in [10] to be applied to the design of water distribution irrigation networks. It was based on a distributed reinforcement learning algorithm called Ant-Q, a variant of the classical Ant Colony System (ACS). In Ant-Q, several autonomous agents learn an optimal policy $\pi : S \rightarrow A$, that outputs an appropriate action $a \in A$, given the current state $s \in S$, where A is the set of all possible actions in a state and S is the set of states. The available information to the agent is the sequence of immediate rewards $r(s_i, a_i)$ for all the possible actions and states $i = 0, 1, 2, \ldots, Q(s, a)$, and a domain dependent heuristic value indicating how good is to select a particular action (a) being in the actual state (s), $HE(s,a)$. Each agent uses the following transition rule to select the action a according with the actual state s:

$$a = \begin{cases} arg \max\limits_{a \in A}(HE^{\alpha}(s,a) \cdot Q^{\beta}(s,a)), & \text{if } q > q_0 \\ P, & \text{otherwise} \end{cases} ,$$

where α and β are parameters that weight the relative importance of pheromone trail and heuristic information, q is a random value in [0,1], and q_0 ($0 \leq q_0 \leq 1$) is calculated in every iteration as follows:

$$q_0 = \frac{q_0 \cdot \lambda}{q_{max}} ,$$

where $q_{max} \in [0,1]$, and P is a random action selected according to the following probability distribution:

$$p(a) = \frac{HE^{\alpha}(s,a) \cdot Q^{\beta}(s,a)}{\sum_{b \in A} HE^{\alpha}(s,b) \cdot Q^{\beta}(s,b)} ,$$

with s being the current state.

The basic idea behind MOAQ is to perform an optimization algorithm with a family of agents (ants) for each objective. Each family k tries to optimize an objective considering the solutions found for the other objectives and its corresponding function HE^k. This way, all the agents from the different families act in the same environment proposing actions and expecting a reward value r which depends on how their actions helped to find trade-off solutions between the rest of the agents. The delayed reinforcement is computed as follows: $Q(s,a) = (1-\rho) \cdot Q(s,a) + \rho \cdot [r(s,a) + \gamma \cdot Q(s',a')]$, where ρ is the learning step, s' and a' are the state and action in the next algorithm step, and γ is a discount factor.

Finally, MOAQ presents other three distinguishing characteristics. First, the j-th ant from the i-th family uses the solution found by the j-th ant of family i-1 while constructing its solution. Second, when a solution found is not feasible, the algorithm applies a punishment to its components on the Q values. And third, along the process, non-dominated solutions are stored in a external set, as usually done in elitist (second generation) MOGAs.

3.2 Ant Algorithm for Bi-criterion Optimization Problems

The so-called BicriterionAnt algorithm was designed by Iredi et al. in [8] to specifically solve a bi-criteria vehicle routing problem. To do so, it uses two different pheromone trail matrices, τ and τ', one for each of the criteria considered.

In every generation, each of m ants in the colony generates a solution to the problem. During its construction trip, the ant selects the next node j to be visited by means of the following probability distribution:

$$p(j) = \begin{cases} \frac{\tau_{ij}^{\lambda\alpha} \cdot \tau_{ij}'^{(1-\lambda)\alpha} \cdot \eta_{ij}^{\lambda\alpha} \cdot \eta_{ij}'^{(1-\lambda)\alpha}}{\sum_{h \in \Omega} \tau_{ih}^{\lambda\alpha} \cdot \tau_{ih}'^{(1-\lambda)\alpha} \cdot \eta_{ih}^{\lambda\alpha} \cdot \eta_{ij}'^{(1-\lambda)\alpha}}, & \text{if } j \in \Omega \\ 0, & \text{otherwise} \end{cases} ,$$

where α and β are the usual weighting parameters, η_{ij} and η'_{ij} are the heuristic values associated to edge (i, j) according to the first and the second objective, respectively, Ω is the current feasible neighborhood of the ant, and λ is computed for each ant t, $t \in \{1, \ldots, m\}$, as follows:

$$\lambda_t = \frac{t-1}{m-1} \ .$$

Once all the ants have generated their solutions, the pheromone trails are evaporated by applying the usual rule on every edge (i, j): $\tau_{ij} = (1 - \rho) \cdot \tau_{ij}$, with $\rho \in [0, 1]$ being the pheromone evaporation rate.

Then, every ant that generated a solution in the non-dominated front at the current iteration is allowed to update both pheromone matrices, τ and τ', by laying down an amount equal to $\frac{1}{l}$, with l being the number of ants currently updating the pheromone trails. The non-dominated solutions generated along the algorithm run are kept in an external set, as it happened in MOAQ.

3.3 Multi Colony for Bi-criterion Optimization Problems

In the same contribution [8], Iredi et al. proposed another MOACO algorithm, BicriterionMC, very similar to the previous BicriterionAnt. The main difference between them is that each ant only updates one pheromone matrix in the new proposal. The authors introduce a general definition for the algorithm with p pheromone trail matrices and then make $p = 2$ to solve bi-criterion problems. To do so, they consider two different methods for the pheromone trail update:

– Method 1 - *Update by origin*: an ant only updates the pheromone trails in its own colony. This method enforces both colonies to search in different regions of the non-dominated front. The algorithm using method 1 is called UnsortBicriterion.
– Method 2 - *Update by region*: the sequence of solutions along the non-dominated front is split into p parts of equal size. Ants that have found solutions in the i-th part update the pheromone trails in colony i, $i \in [1, p]$. The aim is to explicitly guide the ant colonies to search in different regions of the Pareto front, each of them in one region. The algorithm using method 2 is called BicriterionMC.

The other difference is the way to compute the value of the transition parameter λ for the ant t. The authors describe three different rules and propose to use the third of them which gives better results. This rule overlaps the λ-interval of colony i in a 50% with the λ-interval of colony $i - 1$ and colony $i + 1$. Formally, colony i has ants with λ-values in $[\frac{i-1}{p+1}, \frac{i+1}{p+1}]$, with p being equal to the number of colonies.

3.4 Pareto Ant Colony Optimization

Pareto Ant Colony Optimization (P-ACO), proposed by Doerner et al. in [5], was originally applied to solve the multi-objective portfolio selection problem. It considers the classical ACS as the underlying ACO algorithm but the global pheromone update is performed by using two different ants, the best and the second-best solutions generated in the current iteration for each objective k. In P-ACO, several pheromone matrices τ^k are considered, one for each objective k. At every algorithm iteration, each ant computes a set of weights $p = (p_1, \ldots, p_k)$, and uses them to combine the pheromone trail and heuristic information. When an ant has to select the next node to be visited, it uses the ACS transition rule considering the k pheromone matrices:

$$j = \begin{cases} arg \max\limits_{j \in \Omega}([\sum_{k=1}^{K} p_k \cdot \tau_{ij}^k]^\alpha \cdot \eta_{ij}^\beta, & \text{if } q \leq q_0 \\ \hat{i}, & \text{otherwise} \end{cases},$$

where K is the number of objectives, η_{ij} is an aggregated value of attractiveness of edge (i, j) used as heuristic information, and $\hat{i}$ is a node selected according to the probability distribution given by:

$$p(j) = \begin{cases} \frac{\left[\sum_{k=1}^{K} p_k \cdot \tau_{ij}^k\right]^\alpha \cdot \eta_{ij}^\beta}{\sum_{h \in \Omega} \left[\sum_{k=1}^{K} p_k \cdot \tau_{ih}^k\right]^\alpha \cdot \eta_{ih}^\beta}, & \text{if } j \in \Omega \\ 0, & \text{otherwise} \end{cases}.$$

Every time an ant travels an edge (i, j), it performs the local pheromone update in each pheromone trail matrix, i.e., for each objective k, as follows: $\tau_{ij}^k = (1 - \rho) \cdot \tau_{ij}^k + \rho \cdot \tau_0$, with ρ being the pheromone evaporation rate, and τ_0 being the initial pheromone value.

The global pheromone trail information is updated once each ant of the population has constructed its solution. The rule applied for each objective k is as follows: $\tau_{ij}^k = (1 - \rho) \cdot \tau_{ij}^k + \rho \cdot \Delta\tau_{ij}^k$, where $\Delta\tau_{ij}^k$ has the following values:

$$\Delta\tau_{ij}^k = \begin{cases} 15, & \text{if edge } (i, j) \in \text{best and second-best solutions} \\ 10, & \text{if edge } (i, j) \in \text{best solution} \\ 5, & \text{if edge } (i, j) \in \text{second-best solution} \\ 0, & \text{otherwise} \end{cases}.$$

Along the process, the non-dominated solutions found are stored in an external set, as in the previous MOACO algorithms.

3.5 Multiple Ant Colony System

Multiple Ant Colony System (MACS) [1] was proposed as a variation of the MACS-VRPTW introduced in [7]. So, it is also based in ACS but, contrary to its predecessor, MACS uses a single pheromone matrix, τ, and several heuristic information functions, η_k, initially two, η^0 and η^1. In this way, an ant moves from node i to node j by applying the following rule:

$$j = \begin{cases} arg \max\limits_{j \in \Omega} \left(\tau_{ij} \cdot [\eta^0_{ij}]^{\lambda\beta} \cdot [\eta^1_{ij}]^{(1-\lambda)\beta} \right), & \text{if } q \leq q_0 \\ \hat{i}, & \text{otherwise} \end{cases} ,$$

where β weights the relative importance of the objectives with respect to the pheromone trail, λ is computed for each ant t as $\lambda = \frac{t}{m}$, with m being the number of ants, and $\hat{i}$ is a node selected according to the following probability distribution:

$$p(j) = \begin{cases} \frac{\tau_{ij} \cdot [\eta^0_{ij}]^{\lambda\beta} \cdot [\eta^1_{ij}]^{(1-\lambda)\beta}}{\sum_{h \in \Omega} \tau_{ih} \cdot [\eta^0_{ih}]^{\lambda\beta} \cdot [\eta^1_{ih}]^{(1-\lambda)\beta}}, & \text{if } j \in \Omega \\ 0, & \text{otherwise} \end{cases} .$$

Every time an ant crosses the edge (i, j), it performs the local pheromone update as follows: $\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0$.

Initially, τ_0 is calculated from a set of heuristic solutions by taking their average costs in each of the two objective functions, f^0 and f^1, and applying the following expression:

$$\tau_0 = \frac{1}{\hat{f}^0 \cdot \hat{f}^1} .$$

However, the value of τ_0 is not fixed during the algorithm run, as usual in ACS, but it undergoes adaptation. Every time an ant t builds a complete solution, it is compared to the Pareto set P generated till now to check if the former is a non-dominated solution. At the end of each iteration, τ'_0 is calculated by applying the previous formula with the average values of each objective function taken from the solutions currently included in the Pareto set.

Then, if $\tau'_0 > \tau_0$, the current initial pheromone value, the pheromone trails are reinitialized to the new value $\tau_0 \leftarrow \tau'_0$.

Otherwise, the global update is performed with each solution x of the current Pareto optimal set P by applying the following rule on its composing edges (i, j):

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \frac{\rho}{f^0(x) \cdot f^1(x)} .$$

3.6 Multi-Objective Network ACO

Multi-objective Network ACO (MONACO) is quite different to the remaining algorithms reviewed in this section as it was designed to be applied to a dynamic problem, the optimization of the message traffic in a network [2]. Hence, the policy of the network changes according to the algorithm's steps, and it does not wait till the algorithm ends up. In the following, we present an adaptation of the original algorithm, developed by ourselves, to use MONACO in static problems. It requires several modifications such as the fact that the ants have to wait the cycle ends before updating the pheromone trails. The original algorithm takes the classical AS as a base but uses several pheromone trail matrices, τ^k. Each ant, which is defined as a message, uses the multi-pheromone trail and a single heuristic information to choose the next node to visit, according to the following probability distribution:

$$p(j) = \begin{cases} \frac{\eta_{ij}^{\beta} \cdot \prod_{k=1}^{K} (\tau_{ij}^{k})^{\alpha_k}}{\sum_{h \in \Omega} \eta_{ih}^{\beta} \cdot \prod_{k=1}^{K} (\tau_{ih}^{k})^{\alpha_k}}, & \text{if } j \in \Omega \\ 0, & \text{otherwise} \end{cases} .$$

In this formula, η_{ij} is the heuristic information for the edge (i, j), both β and the different α_k's weight the importance of each pheromone matrix value and the heuristic information, K is the number of objective functions and Ω is the feasible neighborhood of the ant at this step. After each cycle, the pheromone trails associated to every edge visited by at least one ant in the current iteration is evaporated in the usual way: $\tau_{ij}^k = (1 - \rho_k) \cdot \tau_{ij}^k$, with ρ_k being the pheromone evaporation rate for objective k (notice that a different one is considered for each pheromone trail matrix). Then, the pheromone trails of these edges are updated. Every ant lays down the following amount of pheromone in each edge (i, j) used by each ant and for every objective k:

$$\Delta\tau_{ij}^k = \frac{Q}{f^k(x)} ,$$

where Q is a constant related to the amount of pheromone laid by the ants, f^k is the objective function k, and x is the solution built by the ant. As said, the aim of the original algorithm is not to find a good set of non-dominated solutions but to make the network work efficiently. To apply it to static optimization problems, we have to store the non-dominated solutions generated in each run in an external set.

3.7 COMPETants

Initially, Doerner et al. introduced COMPETants to deal with bi-objective transportation problems [4]. The algorithm, based on the rankAS, used two ant colonies, each with its own pheromone trail matrix, τ^0 and τ^1, and its heuristic information, η^0 and η^1. An interesting point is that the number of ants in each population is not fixed but undergoes adaptation. When every ant has built its solution, the colony which has constructed better solutions gets more ants for the next iteration. The ants walk through the edges using the following probability distribution to select the next node to be visited (notice that it is adaptation of the AS transition rule to the case of multiple heuristic and pheromone trail values):

$$p(j) = \begin{cases} \frac{\tau_{ij}^{k^{\alpha}} \cdot \eta_{ij}^{k^{\beta}}}{\sum_{h \in \Omega} \tau_{ih}^{k^{\alpha}} \cdot \eta_{ih}^{k^{\beta}}}, & \text{if } j \in \Omega \\ 0, & \text{otherwise} \end{cases} .$$

Each ant uses the τ and η values of its colony k. Besides, some ants in every population, called spies, use another rule combining the information of both pheromone trails:

$$p(j) = \begin{cases} \frac{[0.5\cdot\tau_{ij} + 0.5\cdot\tau'_{ij}]^{\alpha}\cdot\eta_{ij}^{k\,\beta}}{\sum_{h\in\Omega}[0.5\cdot\tau_{ih}+0.5\cdot\tau'_{ih}]^{\alpha}\cdot\eta_{ih}^{k\,\beta}}, & \text{if } j \in \Omega \\ 0, & \text{otherwise} \end{cases} .$$

In this new rule, τ is the ant's pheromone trail and τ' the pheromone trail of the other colony. η^k is the ant's heuristic information.

When every ant has built its solution, the pheromone trails of each edge (i, j) are evaporated: $\tau_{ij}^k = (1 - \rho)\cdot\tau_{ij}^k$.

Then, the Γ best ants of each population deposit pheromone on the edges visited using its own pheromone trail matrix and the following rule:

$$\Delta\tau_{ij}^{\lambda} = 1 - \frac{\lambda - 1}{\Gamma} \quad ,$$

where λ is the position of the ant in the sorted list of the Γ best ants.

Before the end of the cycle, every ant is assigned to the first colony with the following probability:

$$\frac{\hat{f}^1}{\hat{f}^0 + \hat{f}^1} \quad ,$$

with $\hat{f}^1$ being the average of the costs of the solutions in the second colony (that associated to the second objective function), and $\hat{f}^0$ being the average of the costs of the first colony (that corresponding to the first objective function). The remaining ants will be assigned to the second colony.

Finally, the number of spies in both colonies is randomly calculated with the following probability:

$$\frac{f(best)}{4\cdot f'(best') + f(best)} \quad ,$$

where f is the objective function of the current colony, f' is the objective function associated to the other colony, *best* is the best ant of the current colony according to f and *best'* is the best ant of the other according to f'.

4 Experimental Results

4.1 Experimental Setup

In our case, the K-objective symmetric TSP instances considered have been obtained from Jaszkiewicz's web page: `http://www-idss.cs.put.poznan.pl/~jaszkiewicz/`, where several instances usually tackled in evolutionary multi-objective optimization are collected. Each of these instances was constructed from $K = 2$ different single objective TSP instances having the same number of towns. The interested reader can refer to [9] for more information on them. In this first study, we will use four bi-criteria TSP instances: Kroab50, Krocd50, Krobc100, and Kroad100.

To perform the experimental comparison, besides the eight MOACO algorithms reviewed in Section 3, two of the most known, Pareto-based second generation MOGAs (which represent the state-of-the-art in multi-objective evolutionary optimization, see [3]) are considered as baselines for the MOACO algorithms performance. Hence, ten different algorithms will be run on the four bi-criteria TSP instances selected.

We have followed the same comparison methodology that Zitzler et al. in [12]. The comparison metric is thus based on comparing a pair of non-dominated sets by computing the fraction of each set that is covered by the other:

$$C(X', X'') = \frac{|\{a'' \in X''\,;\, \exists a' \in X' : a' \succ a''|}{|X''|} \;\;,$$

where $a' \succ a''$ indicates that the solution a' dominates the solution a''.

Hence, the value $C(X', X'') = 1$ means that all the solutions in X'' are dominated by or equal to solutions in X'. The opposite, $C(X', X'') = 0$, represents the situation where none of the solutions in X'' are covered by the set X'. Note that both $C(X', X'')$ and $C(X'', X')$ have to be considered, since $C(X', X'')$ is not necessarily equal to $1 - C(X'', X')$.

In addition, we will compare the number of iterations and evaluations needed by each algorithm to generate its Pareto front.

As seen in Section 3, the eight MOACO algorithms considered are characterized by tackling very diverse applications. Hence, several changes have been done in our implementations to adapt them to solve the multi-objective TSP, moreover than making them manage an appropriate problem representation, the usual permutation. These changes are not reported due to the lack of space.

Each of the considered multi-objective algorithms (both MOACO and MOGAs) has been run ten times for each of the four bi-criteria TSP instances during 300 seconds (small instances) and 900 seconds (large ones) in an Intel Celeron^{TM} 1200MHz computer with 256 MB of RAM. The generic parameter values considered are the usual ones when applying MOACO and MOGAs to the TSP problem: 20 ants, $(\alpha, \beta) = (1, 2)$, $\rho = 0.2$, $q_0 = 0.98$ (0.99 for MOAQ). To choose the initial pheromone trail value τ_0, we have employed the same method used by each algorithm to update the pheromone trails, obtaining it from the greedy solution of each of the K single-objective problems.

On the other hand, the values of the specific parameters, such as λ in MOAQ, have been obtained from the papers where the algorithms using them were defined ($\lambda = 0.9$, $\gamma = 0.4$). As regards the MOGAs, the population include 100 individuals (80 plus 20 in the elite population in SPEA2), and the crossover and mutation probabilities are respectively 0.8 and 0.1.

Fig. 1. Number of iterations and evaluations in the Kroab50 and Krocd50 runs

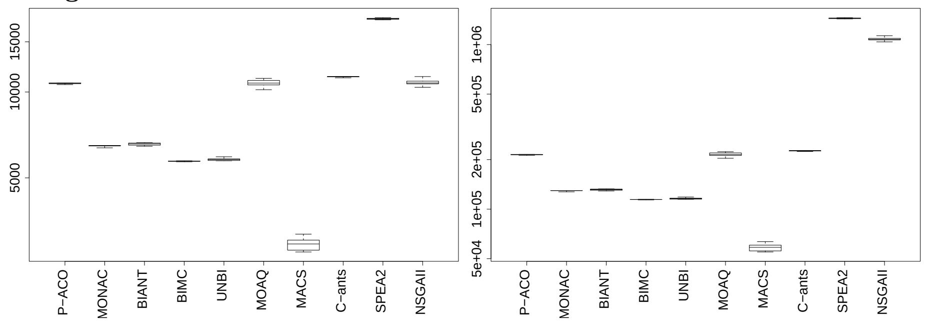
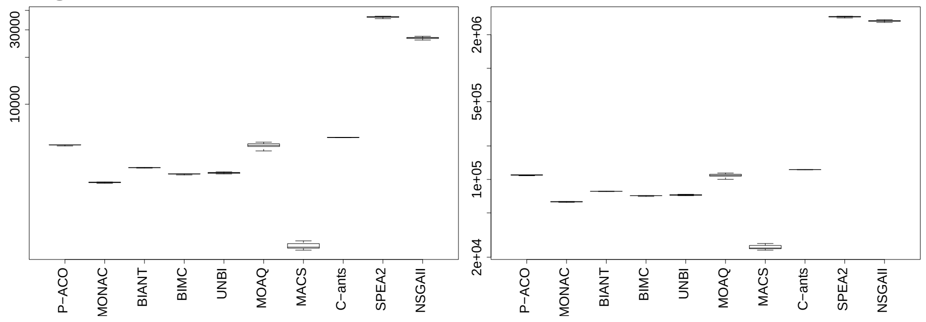


Fig. 2. Number of iterations and evaluations in the Krobc100 and Kroad100 runs



4.2 Results Obtained and Analysis of Results

All the experimental data are reported in the form of box-plots, where the minimum, maximum and median values as well as the upper and lower quartiles (upper and lower ends of the box) are graphically represented. Figures 1 and 2 show the statistics of the ten runs of each algorithm in each bi-criteria TSP instances (in a logarithmic scale and grouped in a single box-plot for each pair of instances of the same size) . In view of these graphics, we can see that the MOGAs, NSGA-II and SPEA2, can perform much more iterations and evaluations than the MOACO algorithms considered in the same fixed run time. Hence, this shows how MOACO algorithms are "slower" than MOGAs in the current problem. Notice also that MACS is the algorithm which performs less iterations and evaluations while COMPETants is the quickest of the MOACO algorithms.

The graphics in Figure 3 are box-plots based on the C metric. Each rectangle contains four box-plots (from left to right, Kroab50, Krocd50, Krobc100, and Kroad100) representing the distribution of the C values for a certain ordered pair of algorithms. Each box refers to algorithm A associated with the corresponding row and algorithm B associated with the corresponding column and gives the fraction of B covered by A ($C(A, B)$). Consider, for instance, the top right box, which represents the fraction of solutions of SPEA2 covered by the nondominated sets produced by the P-ACO algorithm.

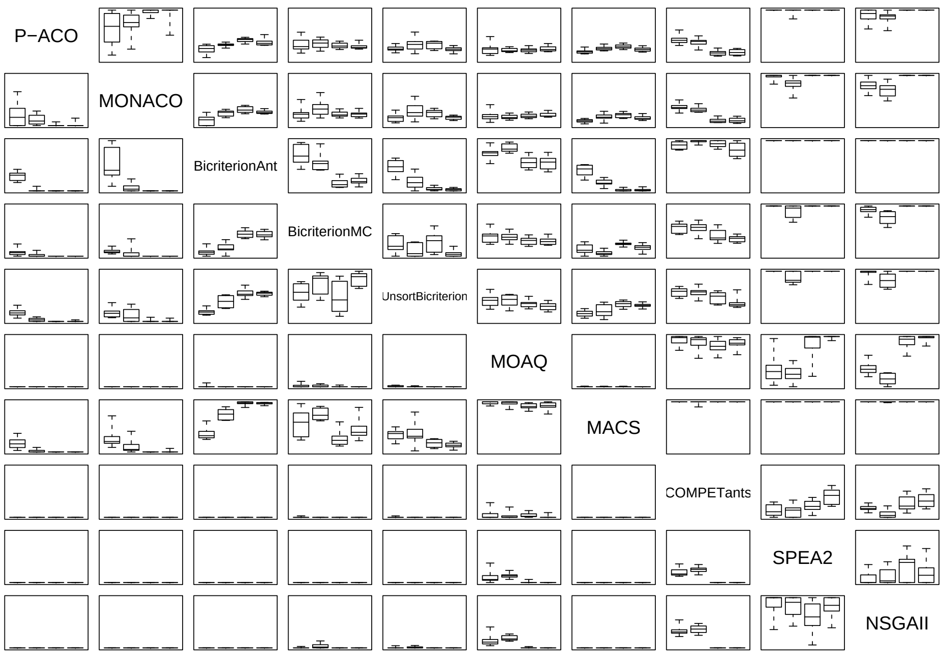


Fig. 3. Box-plots of the results obtained in the C metric

In view of the box-plots of metric C, we can draw the conclusion that the MOACO algorithms considered are very competitive against the MOGAs implemented. The former offer good sets of non-dominated solutions which almost always dominate the solutions returned by NSGA-II and SPEA2. In addition, the Pareto fronts derived by the MOGAs do not dominate any of the fronts given by MOACO algorithms.

It is not easy to say which MOACO algorithm performs best, as all of them derive Pareto sets of similar quality. Maybe P-ACO could be considered as the algorithm with the best global performance as its Pareto fronts are few dominated by the remainder of the MOACO algorithms (see the box-plots in the first column), while they dominate the remainder to some degree (see those in the first row). However, it is easier to identify the algorithms which usually return Pareto sets of bad quality for the problem. This is the case of SPEA2, NSGA-II, COMPETants, and MOAQ. Besides, we should notice that all of these non-dominated solution sets are very well dominated by those got by MACS.

5 Concluding Remarks and Future Works

In the current contribution, we have developed a experimental study comparing the performance of the different existing proposals for Pareto-based MOACO algorithms when applied to several classical instances of the bi-criteria TSP.

From the results obtained, we have drawn the conclusion that the MOACO algorithms considered are more competitive in the current problem than two of the state-of-the-art MOGAs, SPEA-II and NSGA2.

Several ideas for future developments arise from this preliminary study: to analyze (i) the performance of the considered MOACO algorithms in other, larger instances of the multi-objective TSP; (ii) the influence of adding local search optimizers to the MOACO algorithms and to compare the performance of the resulting techniques against that of memetic algorithms such as Jaszkiewicz's MOGLS [9]; (iii) the performance of MOACO algorithms in other complex multi-objective combinatorial optimization problems.

References

1. B. Barán, M. Schaerer, A Multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows, Proc. Twenty first IASTED International Conference on Applied Informatics, Insbruck, Austria, February 10-13, 2003, pp. 97-102.
2. P. Cardoso, M. Jesús, A. Márquez, MONACO - Multi-Objective Network Optimisation Based on an ACO, Proc. X Encuentros de Geometría Computacional, Seville, Spain, June 16-17, 2003.
3. C.A. Coello, D.A. Van Veldhuizen, G.B. Lamant, Evolutionary Algorithms for Solving Multi-objective Problems, Kluwer, 2002.
4. K. Doerner, R.F. Hartl, M. Teimann, Are COMPETants More Competent for Problem Solving? - The Case of Full Truckload Transportation, Central European Journal of Operations Research (CEJOR), 11:2, 2003, pp. 115-141.
5. K. Doerner, W.J. Gutjahr, R.F. Hartl, C. Strauss, C. Stummer, Pareto Ant Colony Optimization: A Metaheuristic Approach to Multiobjective Portfolio Selection, Annals of Operations Research, 2004, to appear.
6. M. Dorigo, T. Stützle, The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances, In: F. Glover, G.A. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer, 2003.
7. L. Gambardella, E. Taillard, G. Agazzi, MACS-VRPTW: A Multiple ACS for Vehicle Routing Problems with Time Windows, In: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, 1999, pp. 73-76.
8. S. Iredi, D. Merkle, M. Middendorf, Bi-Criterion Optimization with Multi Colony Ant Algorithms, Proc. First International Conference on Evolutionary Multi-criterion Optimization (EMO'01), LNCS 1993, 2001, pp. 359-372.
9. A. Jaszkiewicz. Genetic Local Search for Multi-objective Combinatorial Optimization, European Journal of Operational Research, 137:1, 2002, pp. 50-71.
10. C.E. Mariano, E. Morales, A Multiple Objective Ant-Q Algorithm for the Design of Water Distribution Irrigation Networks, Technical Report HC-9904, Instituto Mexicano de Tecnología del Agua, June 1999.
11. E.L. Ulungu, J. Teghem, Multi-objective Combinatorial Optimization: A Survey, Journal of Multi-Criteria Decision Analysis, 3, 1994, pp. 83-104.
12. E. Zitzler, K. Deb, L. Thiele, Comparison of Multiobjective Evolutionary Algorithms: Empirical Results, Evolutionary Computation, 8:2, 2000, pp. 173-195.