Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 180 (2007) 116–148
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP ☆

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

Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain

Received 30 August 2004; accepted 6 March 2006
Available online 5 June 2006

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 a proposal of a taxonomy for them is presented. In addition, an empirical analysis is developed by analyzing their performance on several instances of the bi-criteria traveling salesman problem in comparison with two well-known multi-objective genetic algorithms.

Keywords: Traveling salesman; Ant colony optimization; Multiple objective optimization; Multiple objective evolutionary algorithms

1. Introduction

Multi-criteria optimization problems are characterized by the fact that several objectives have to be simultaneously optimized, thus making especially difficult the problem solving [6]. The use of metaheuristics for these problems has been subject to a growing interest in the last decade. 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 [26,33].

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 [20], several researchers have developed different ACO algorithms that performed properly when solving combinatorial problems such as the traveling salesman problem, the quadratic assignment problem, the sequential ordering problem, production scheduling, timetabling, project scheduling, vehicle routing, telecommunication routing, investment planning, staff scheduling, among others [21,11].

☆ This work was partially supported by the Spanish Ministerio de Ciencia y Tecnología under project TIC2003-00877 (including FEDER fundings), under Network HEUR TIC2002-10866-E, and under a scholarship from the Education and Universities Spanish Government Secretariat given to the author C. García-Martínez.

* Corresponding author. Tel.: +34 58 244019; fax: +34 58 243317.
E-mail addresses: cgarcia@decsai.ugr.es (C. García-Martínez), ocordon@decsai.ugr.es (O. Cordón), herrera@decsai.ugr.es (F. Herrera).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2006.03.041

Recently, some researchers have designed ACO algorithms to deal with multi-objective problems (MOACO algorithms) [1,3,5,16,17,25,28,30,32,38,39,41,46]. The most of them are specific proposals to solve a concrete multi-criteria problem such as scheduling, vehicle routing, or portfolio selection, among others.

The aim of the current contribution is to review and classify the existing MOACO algorithms by proposing a taxonomy for them and developing a systematic experimental study comparing them when tackling a concrete benchmark problem, the multi-objective traveling salesman problem (MOTSP). An empirical study based on this taxonomy will allow us to analyze whether the fact of belonging to a specific MOACO family involves a good or bad performance or, instead, the characteristics of the specific MOACO algorithm itself determine the quality of the Pareto fronts generated.

To do our study, six instances of the bi-criteria TSP are chosen and all the former MOACO algorithms are applied to solve them. Besides, two well-established multi-objective genetic algorithms (MOGAs) are considered as baselines to check the global performance of MOACO approaches, NSGA-II [15] and SPEA2 [48]. A detailed study is then developed to analyze the performance of each algorithm by considering several classical multi-objective quality metrics.

This paper is structured as follows. In Section 2, some preliminaries about ACO and multi-objective optimization are reviewed. In Section 3, the existing MOACO algorithms are introduced, reporting their key characteristics. In Section 4, a taxonomy for them is presented. The requirements of the experimentation (adaptation of the algorithms considered to the bi-objective TSP, MOTSP instances used, metrics of performance considered and parameter settings) are commented on in Section 5. The experimental results of the MOACO algorithms and MOGAs considered are analyzed in Section 6. In Section 7, some concluding remarks and proposals for future works are shown. Finally, the paper is complemented with an introduction to evolutionary multi-objective optimization in Appendix A.

2. Preliminaries

2.1. Ant colony optimization

Ants are social insects that live in colonies and that, thanks to their collaborative interaction, are capable of showing complex behaviors and to perform difficult tasks from an ant's local perspective. A very interesting aspect of the behavior of several ant species is their ability to find shortest paths between the ants' nest and the food sources. This fact is specially noticeable having in mind that, in many ant species, ants are almost blind, which avoids the exploitation of visual clues.

While walking between their nest and food sources, some ant species deposit a chemical called pheromone (an odorous substance). If no pheromone trails are available, ants move essentially at random, but in the presence of pheromone they have a tendency to follow the trail. In practice, choices between different paths occur when several paths intersect. Then, ants choose the path to follow by a probabilistic decision biased by the amount of pheromone: the stronger the pheromone trail, the higher its desirability. Because ants in turn deposit pheromone on the path they are following, this behavior results in a self-reinforcing process leading to the formation of paths marked by high pheromone concentration. This behavior also allows ants to identify shortest paths between their nest and the food source.[1]

The latter procedure is complemented in the natural environment by the fact that pheromone evaporates after some time. This way, less promising paths progressively loose pheromone because of being visited by less and less ants.

ACO algorithms take inspiration from the behavior of real ant colonies to solve combinatorial optimization problems. They are based on a colony of artificial ants, that is, simple computational agents that work cooperatively and communicate through artificial pheromone trails [22].

ACO algorithms are essentially construction algorithms: in each algorithm iteration, every ant constructs 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, i.e., of traveling the edge a_{ij}. It is denoted by η_{ij}.

[1] Note that ants only communicate indirectly, through modifications of the physical environment they perceive. This form of communication is called *artificial stigmergy* in [18].

This information is not modified by the ants during the algorithm run.

- *(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 run depending on the solutions found by the ants. It is denoted by τ_{ij}.

Several ACO algorithms have been proposed which are included within the ACO metaheuristic [11,18,21], such as the Ant System [20], the ant colony system [19], the Max–Min ant system [45], the rank-based ant system [4], and the best–worst ant system [8–10]. The two former algorithms are briefly reviewed as follows.

2.1.1. Ant system

Ant system (AS) [20], developed by Dorigo, Maniezzo and Colorni in 1991, 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. Initially, AS did not use any centralized daemon actions (actions not performed by the ants but by an external agent, which have not got any natural counterpart, but are just additional procedures to improve the metaheuristic performance), but it is very straightforward to, for example, add a local search procedure to refine the solutions generated by the ants.

Solutions in AS are constructed as follows. At each construction step, an ant h in AS chooses to go to a next node with a probability that is computed as

$$p_{ij}^{h} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum_{u\in\mathcal{N}_i^h}[\tau_{iu}]^{\alpha}\cdot[\eta_{iu}]^{\beta}}, & \text{if } j\in\mathcal{N}_h(i),\\ 0, & \text{otherwise,}\end{cases}$$

where $\mathcal{N}_h(i)$ is the feasible neighborhood of ant h when located at node i, and $\alpha,\beta\in\mathbb{R}$ are two parameters that weight the relative importance of the pheromone trail and the heuristic information. Each ant h stores the sequence it has followed so far and this memory L_h is exploited to determine $\mathcal{N}_h(i)$ in each construction step.

As said, the pheromone deposit is made once all ants have finished to construct their solutions. First, the pheromone trail associated to every edge is evaporated by reducing all pheromones by a constant factor:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij},$$

where $\rho\in(0,1]$ is the evaporation rate. Next, each ant retraces the path it has followed (stored in its local memory L_h) and deposits an amount of pheromone $\Delta\tau_{ij}^{h}$ on each traversed connection (on-line *a posteriori* update or global update):

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{h}, \quad \forall a_{ij}\in S_h,$$

where $\Delta\tau_{ij}^{h} = f(C(S_h))$, i.e., the amount of pheromone released is function of the quality $C(S_h)$ of the solution S_h of ant h. In the TSP, $f(x)$ is usually equal to x^{-1}.

Before concluding this section, it is important to notice that the creators of AS also proposed a typically better performing, extended version of this algorithm called *elitist AS* [20]. In elitist AS, once the ants have released pheromone on the connections associated to their generated solutions, the daemon performs an additional pheromone deposit on the edges belonging to the best solution found until that moment in the search process (this solution is called global-best solution in the following). The amount of pheromone deposited, which depends on the quality of that global best solution, is weighted by the number of elitist ants considered, e, as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + e\cdot f(C(S_{\text{global-best}})), \quad \forall a_{ij}\in S_{\text{global-best}}.$$

2.1.2. Ant colony system

Ant colony system (ACS) [19] is one of the first successors of AS. It introduces three major modifications into AS:

1. ACS uses a different transition rule, which is called *pseudo-random proportional rule*: Let h be an ant located at a node i, $q_0\in[0,1]$ be a parameter, and q a random value in $[0,1]$. The next node j to be visited is randomly chosen according to the following probability distribution:
   - If $q\leqslant q_0$:

$$p_{ij}^{h} = \begin{cases} 1, & \text{if } j = \arg\max_{u\in\mathcal{N}_h(i)}\{\tau_{iu}^{\alpha}\cdot\eta_{iu}^{\beta}\},\\ 0, & \text{otherwise,}\end{cases}$$

- else ($q > q_0$):

$$p_{ij}^h = \begin{cases} \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{u \in \mathcal{N}_i^h} [\tau_{iu}]^\alpha \cdot [\eta_{iu}]^\beta}, & \text{if } j \in \mathcal{N}_h(i), \\ 0, & \text{otherwise.} \end{cases}$$

As can be seen, the rule has a double aim: when $q \leqslant q_0$, it exploits the available knowledge, choosing the best option with respect to the heuristic information and the pheromone trail. However, if $q > q_0$, it applies a controlled exploration, as done in AS. In summary, the rule establishes a trade-off between the exploration of new connections and the exploitation of the information available at that moment.

2. Only the daemon (and not the individual ants) triggers the global pheromone update, i.e., an offline pheromone trail update is done. To do so, ACS only considers a single ant, the one who generated the global best solution, $S_{\text{global-best}}$.
The pheromone update is done by first evaporating the pheromone trails on all the connections used by the global-best ant (it is important to notice that *in ACS, pheromone evaporation is only applied to the connections of the solution that is also used to deposit pheromone* (see p. 77 in [22])) as follows:

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij}, \quad \forall a_{ij} \in S_{\text{global-best}}.$$

Next, the daemon deposits pheromone by the rule:

$$\tau_{ij} \leftarrow \tau_{ij} + \rho \cdot f(C(S_{\text{global-best}})), \quad \forall a_{ij} \in S_{\text{global-best}}.$$

Additionally, the daemon can apply a local search algorithm to improve the ants' solutions before updating the pheromone trails.

3. Ants apply an *online step-by-step pheromone trail update* (local update) that encourages the generation of different solutions to those yet found. Each time an ant travels an edge a_{ij}, it applies the rule:

$$\tau_{ij} \leftarrow (1-\varphi) \cdot \tau_{ij} + \varphi \cdot \tau_0,$$

where $\varphi \in (0,1]$ is a second pheromone decay parameter. As can be seen, the *online step-by-step update rule* includes both, pheromone evaporation and deposit. Because the amount of pheromone deposited is very small (in fact, τ_0 is the initial pheromone trail value which is chosen in such a way that, in practice, it corresponds to a lower pheromone trail limit. That is by the choice of the ACS pheromone update rules, no pheromone trail value can fall below τ_0), the application of this rule makes the pheromone trail on the connections traversed by an ant decrease.[2] Hence, this results in an additional exploration technique of ACS by making the connections traversed by an ant less attractive to the following ants and helps to avoid that every ant follows the same path.

2.2. Multi-objective optimization

Multi-criteria optimization problems are characterized by the fact that several objectives have to be simultaneously optimized. Hence, there is not usually a single best solution solving the problem, i.e., being better than the remainder with respect to every objective, as in single-objective optimization. Instead, in a typical multi-objective optimization problem, there is a set of solutions that are superior to the remainder when all the objectives are considered, the Pareto set. These solutions are known as non-dominated solutions [6], while the remainder are known as dominated solutions. Since none of the Pareto set solutions is absolutely better than the other non-dominated solutions, all of them 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 a multi-objective minimization problem with n parameters (decision variables) and K objectives:

$$\text{Min} \quad f(x) = (f_1(x), f_2(x), \ldots, f_K(x)), \quad \text{with}$$
$$x = (x_1, x_2, \ldots, x_n) \in X.$$

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

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

[2] ACS is actually based on Ant-Q, an earlier algorithm proposed by Gambardella and Dorigo [24]. The only difference between ACS and Ant-Q is in the definition of the term τ_0 in the online step-by-step update rule, which in Ant-Q is the discounted evaluation of the next state, set to $\gamma \cdot \max_{j \in \mathcal{N}_h(i)}\{\tau_{ij}\}$. However, experimental results suggested that ACS results in the same level of performance and, because of its greater simplicity, it was preferred.

Any vector that is not dominated by any other is said to be Pareto-optimal or non-dominated.

As said, a common difficulty with multi-objective optimization is the appearance of an *objective conflict* [31], i.e., none of the feasible solutions allows simultaneous optimal solutions for all objectives. Mathematically, the best way of acting is to keep with those solutions with the least objective conflict. These solutions can be viewed as points in the search space that are optimally placed from the individual optimum of each objective. However, such solutions may not satisfy a decision-maker because he may want a solution that satisfies some associated priorities of the objectives. To find such points, all classical methods scalarize the objective vector reducing it to a scalar optimization problem. Actually, in this case a compromise solution is found subjected to specified constraints.

Four are the most commonly classical methods to solve the multi-optimality [7]: objective weighting, distance functions, Min–Max formulation and Lexicographic approach. All of them are based on reducing the multi-objective optimization problem to a single objective one: In the former one, the multiple objective functions are combined into one overall objective function, as follows:

$$F = \sum_{i=1}^{K} w_i \cdot f_i(x), \qquad \sum_{i=1}^{K} w_i = 1, \quad 0 \leqslant w_i \leqslant 1.$$

In distance functions, the single-objective F function to be optimized is calculated using a *demand-level* vector, $\tilde{f}$, which has to be specified by a decision-maker:

$$F = \left[\sum_{i=1}^{K} |f_i(x) - \tilde{f}_i|^r \right]^{1/r}, \quad 1 \leqslant r < \infty.$$

Min–Max formulation attempts to minimize the relative deviations of the single-objective functions from the corresponding individual optimum. Hence, it tries to minimize:

$$f(x) = \max[Z_i(x)], \quad i = 1, 2, \ldots, K$$

where $Z_i(x)$ is calculated for a non-negative target optimal value $\tilde{f}_i > 0$:

$$Z_i(x) = (f_i - \tilde{f}_i)/\tilde{f}_i.$$

The last method, Lexicographic approach [7], is based on an order of importance of the objectives. Then, it attempts to optimize the ith objective keeping the best found solution for the $(i-1)$th objective.

All the classical techniques used to solve multi-objective problem have serious drawbacks [7], that are reviewed as follows:

- Since objectives are combined to form a single objective function, a single Pareto-optimal solution can be simultaneously obtained. In real-world problems, the decision-maker often requires different alternatives for the decision making, but these techniques can not offer them.
- Even if different weights are adopted during a single run in order to obtain several solutions, the algorithm will not be able to generate concave portions of the Pareto front [12].
- Furthermore, before forming the single objective from the objectives set, the decision-maker must usually have a thorough knowledge of the priority of each objective, which is a difficult problem in itself.
- If some objectives are noisy or have a discontinuous variable space, these methods may not appropriately work.
- If the objective functions are not deterministic, the definition of a weight vector or a demand level may become even more difficult.
- They have much sensitivity and dependency toward weights or demand levels.

In order to solve these problems, some advanced multi-objective optimization techniques have been proposed in the last few years. They are mainly based on well-established metaheuristics such as simulated annealing, evolutionary algorithms, and ACO algorithms, as seen in this paper [26].

3. Multiple objective ant colony optimization algorithms

In this section, the different existing proposals for MOACO algorithms are reviewed by reporting their main characteristics, as well as the original problem they were designed to tackle.

3.1. Multiple objective Ant-Q algorithm

Multiple objective Ant-Q algorithm (MOAQ) is a MOACO algorithm that was proposed by Mariano and Morales in [37,38] to be applied to the design of water distribution irrigation networks. It was based on a distributed reinforcement learning algorithm called Ant-Q [24], a variant of the classical ACS [19] (see Section 2.1.2). In Ant-Q, several

autonomous agents learn an optimal policy $\pi: S \to A$, 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 to the actual state s:

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

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 $(0 \leqslant q_0 \leqslant 1)$ is calculated in every iteration as follows:

$$q_0 = (q_0 \cdot \lambda)/q_{\max},$$

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. On the other hand, when a solution found is not feasible, the algorithm applies a punishment to its components on the Q values.

Finally, MOAQ presents a distinguishing characteristic. While the jth ant from the ith family $(i > 1)$ is constructing its solution, it takes into account the solution found by the jth ant of family $i-1$. Hence, the objectives are processed in a certain order of importance similarly to the lexicographic ordering. This issue is related to the problem to which the algorithm was applied, the design of water distribution irrigation networks. In this problem some objectives needed to be determined before optimizing others. So, it needed to be tackled in a specific order. At first sight, this could seem that the algorithm would return only a single solution, which is the best for the first objective, and try to optimize the remainder. However, MOAQ keeps an external set of non-dominated solutions that is returned when the run finishes. The authors explain this characteristic in [38] by indicating that the algorithm tries to find a set of compromise solutions for all the objectives involved according to an order of importance, which, in some cases, can be arbitrary.

3.2. Ant algorithm for bi-criterion optimization problems

In [32], Iredi et al. introduced some general techniques in order to solve multi or bi-criteria problems by ACO algorithms. They tested these techniques on a bi-criteria vehicle routing problem. In this section, we describe one of the algorithms proposed in [32], which will be called BicriterionAnt. It uses a pheromone trail matrix for every objective, e.g., when there are two objectives, there will be two pheromone matrices, τ and τ'.

In every generation, each of the 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:

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

where α and β are the usual weighting parameters, η_{ij} and η'_{ij} are the heuristic values associated to edge a_{ij} 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 h, $h \in \{1, \ldots, m\}$, in order to force the ants to search in different regions of the Pareto front, as follows:

$$\lambda_h = (h-1)/(m-1).$$

Once all the ants have generated their solutions, the pheromone trails are evaporated by applying the usual rule on every edge a_{ij}:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij}$$

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 $1/l$, 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 [32], Iredi et al. proposed another MOACO algorithm. It was designed from BicriterionAnt (see Section 3.2) with a distinguishing remark: it takes as an user parameter the number of colonies N_C, which is independent from the number of objectives. Then, each colony has a pheromone trail matrix associated for every objective, e.g., if there are two objectives to be optimized, then there will be two pheromone trail matrices in every colony.

In order to update the pheromone matrices, the authors consider two different methods:

- Method 1—*Update by origin*: an ant only updates the pheromone trail matrices in its own colony. The algorithm using method 1 will be called UnsortBicriterion.
- Method 2—*Update by region*: the sequence of solutions along the non-dominated front is split into N_C parts of equal size. Ants that have found solutions in the ith part update the pheromone trails in colony $i, i \in [1, N_C]$. 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.

Finally, another difference is the way to compute the value of the transition parameter λ for ant h. 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 $[(i-1)/(N_C+1),(i+1)/(N_C+1)]$.

3.4. Pareto ant colony optimization

Pareto ant colony optimization (P-ACO), proposed by Doerner et al. in [17], 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 it 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:

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

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

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

Every time an ant travels an edge a_{ij}, 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 in the tackled problem:

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

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 for vehicle routing problem with time windows

As its name suggests, the multiple ant colony system for vehicle routing problem with time windows (MACS-VRPTW) algorithm introduced by Gambardella et al. in [25] was thought to solve this specific kind of vehicle routing problems. As P-ACO, it also starts from the classical ACS.

MACS-VRPTW is based on setting up a preference to minimize one objective (the number of tours) over the other (the travel time). This solution is defined as the first of a lexicographic order on the values of the objectives. It defines two different colonies, ACS-VEI and ACS-TIME, whose activities are coordinated by the global MACS-VRPTW algorithm in order that both objectives are simultaneously optimized. The former colony tries to diminish the number of vehicles used while the latter optimizes the feasible solutions obtained by the former. Each one uses an independent pheromone trail matrix for its specific objective, and they collaborate by sharing the best solution found by their cooperative action. The global algorithm kills and runs again the two colonies each time a new best solution containing less vehicles than the previous one is obtained.

3.6. Multiple ant colony system

Multiple ant colony system (MACS) [1] was proposed as a variation of the MACS-VRPTW algorithm reviewed in the previous section. So, it is also based on 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:

$$j = \begin{cases} \arg\max_{j\in\Omega}(\tau_{ij}\cdot[\eta_{ij}^0]^{\lambda\beta}\cdot[\eta_{ij}^1]^{(1-\lambda)\beta}), & \text{if } q \leqslant q_0, \\ \hat{i}, & \text{otherwise,} \end{cases}$$

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

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

Every time an ant crosses the edge a_{ij}, 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, $\hat{f}^0$ and $\hat{f}^1$, and applying the following expression:

$$\tau_0 = 1/(\hat{f}^0\cdot\hat{f}^1).$$

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 h 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 equation 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 S of the current Pareto optimal set P by applying the following rule on its composing edges a_{ij}:

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

Recently, Pinto et al. have slightly modified MACS in order to solve a multi-objective multi-cast routing problem [41].

3.7. 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 [5]. 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 till the cycle ends before updating the pheromone trails.

The original algorithm takes the classical AS [20] 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:

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

In this equation, η_{ij} is the heuristic information for edge a_{ij}, 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 are 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 evaporation rate is considered for each pheromone trail matrix). Then, the pheromone trails of these edges are updated. Every ant h lays down the following amount of pheromone in each edge a_{ij} used by each ant and for every objective k:

$$\Delta\tau_{ij}^k = Q/f^k(S_h),$$

where Q is a constant related to the amount of pheromone laid by the ants, f^k is the objective function k, and S_h 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.

3.8. COMPETants

Initially, Doerner et al. introduced COMPETants to deal with bi-objective transportation problems [16]. The algorithm, based on the rank-based AS [4], 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 an adaptation of the AS transition rule to the case of multiple heuristic and pheromone trail values):

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

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:

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

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 a_{ij} 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:

$$\Delta\tau_{ij}^{\lambda} = 1 - (\lambda - 1)/\Gamma,$$

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:

$$\hat{f}^1/(\hat{f}^0 + \hat{f}^1)$$

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:

$$f(\text{best})/(4 \cdot f'(\text{best}') + f(\text{best})),$$

where f is the objective function associated to 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'.

3.9. Multiple objective ant colony optimization metaheuristic

In [28], Gravel et al. proposed a multi-objective ACO algorithm for a real-world scheduling problem related to the aluminum production industry, that was called Multiple Objective ACO metaheuristic (MOACOM) by its creators. MOACOM is based on an AS algorithm that deals with the multiple objectives in a lexicographic order, a priori established by the decision maker. At the end of a cycle, only the first solution according to the lexicographic order considered is taken into account. For this reason, the aim of MOACOM is not to find a good non-dominated solution set. Hence, the algorithm deals with a single heuristic information and pheromone trail matrices. The authors provide several rules to construct the heuristic matrix η, where each value is the result of the aggregation of information associated to every objective.

Considering the latter two matrices, the AS transition rule is applied to build the ants solutions. At the end of a cycle, the pheromone trail intensity is updated based on the evaluation of the primary objective in isolation. Every ant h lays down the following amount of pheromone on the edges used:

$$\Delta\tau_{ij} = Q/f^0(S_h),$$

where f^0 is the primary objective function, S_h is the solution found by the ant, and Q is a constant.

3.10. Ant colony optimization approach to multiple objectives

As the previous algorithm, the ACO approach to multiple objectives (ACOAMO), proposed in [39] to solve a multi-objective JIT sequencing problem, is based on dealing with a single best solution and not with a set of non-dominated ones in each iteration. This solution is the first of a lexicographic order defined on the objectives. ACOAMO splits the solutions in components and randomly distributes these components over the search space. The heuristic information is the inverse of the distances between the components, and the ants are guided by the following probability distribution:

$$p(j) = \begin{cases} \frac{\tau_{ij}\cdot\eta_{ij}}{\sum_{u\in\Omega}\tau_{iu}\cdot\eta_{iu}}, & \text{if } j \in \Omega, \\ 0, & \text{otherwise} \end{cases}$$

with τ being the pheromone trail matrix, η referring to the heuristic information, and Ω being the current feasible neighborhood of the ant.

When an ant goes through the edge a_{ij}, it applies the ACS local update rule:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho\cdot\tau_0.$$

Finally, every ant that completes a solution computes its cost as the Euclidean sum of the distances of every edge crossed, C. If this value is lesser than the smallest value of C found so far, then the global update is performed to every edge in this solution according to the following expression:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho\cdot C^{-1}.$$

3.11. SACO

SACO is a very specific MOACO algorithm proposed in [46] by T'kindt et al. for 2-machine bi-criteria flowshop scheduling problems. Its name comes from the fact that it puts into effect a search methodology mimicking that applied by simulated annealing (developing a stronger diversification at the beginning of the trek, and a stronger intensification at later stages). Again, it is based on only dealing with a single best solution, that having the best cost for one of the objectives, and not with a set of non-dominated solutions. At every cycle, each ant builds a feasible solution using the pheromone trail information, which can be used in two different ways: (a) *intensification mode*, where the ant chooses the most suitable job according to the highest value of τ_{ij}, and (b) *diversification mode*, where the ant chooses the next node by a random-proportional rule. The probability of selecting the intensification mode is regulated by p_0, a parameter computed at every iteration it as follows:

$$p_0 = \log(it)/\log(N)$$

with N being the total number of iterations.

At the end of the cycle, the evaporation is performed on every edge and the best solution is kept and used to update the pheromone trails of the edges composing it

$$\tau_{ij} = \tau_{ij} + 1/f^0(S)$$

with S being the best solution found and f^0 being one of the objectives.

3.12. Multiple objective ant colony optimization approach to assembly line balancing problems

The MOACO approach to assembly line balancing problem (MOACO-ALBP), proposed in [3] to

solve Assembly Line Balancing Problems, is based on dealing with a single best solution and not with a set of non-dominated ones in each iteration. This solution is the first of a lexicographic order defined on the objectives.

MOACO-ALBP performs like an AS with a pheromone matrix and a heuristic one. The pheromone matrix is updated according to a function of all the objective criterions. The used formulas are specific to the Assembly Line Balancing Problem. In addition, it keeps the best obtained solution across the iterations.

3.13. Multi-criteria population-based ACO

In [30], Guntsch and Middendorf proposed an adaptation of the Population-based ACO (PACO) [29] for multiple objective optimization (MO-PACO). PACO differs from the standard ACO heuristic in that it employs a population $P = \{\pi_1, \ldots, \pi_k\}$ of k good solutions, from which the pheromone information τ_{ij} is derived as follows: Each element of the pheromone matrix has an initial value τ_{init}. Whenever a solution enters the population P, a positive update is performed by adding a pheromone amount Δ. In addition, if a solution is removed from the population, its influence is explicitly removed from the pheromone matrix by performing a *negative* update, i.e., using $-\Delta$. As a result, the pheromone matrix is perfectly defined by the members in the population P:

$$\tau_{ij} = \tau_{init} + \Delta \cdot |\{\pi \in P | (i,j) \in \pi\}|.$$

Notice that, since the construction of the solutions depends, among others, on the current distribution of individuals in P, PACO seems similar to the Estimation of Distribution Algorithms, where new solutions depends only on the distribution of the individual of the current population.

MO-PACO is the multiple objective version of PACO. It uses an external population, with the non-dominated solutions it finds (Q), which the individuals of P are chosen from. Every m solutions, MO-PACO creates a new P with a randomly chosen individual and the $(k-1)$ nearest ones to it. Then, MO-PACO calculates several pheromone matrix, one for each optimization criterion, which, together with a heuristic matrix for each optimization criterion, are used for the transition rule. Every new solution is tested against the individuals in Q in order to keep Q updated with the non-dominated solutions found so far.

In addition, MO-PACO uses an *average-rank-weight method* that modifies the probability distribution used in the transition rule, which assigns a higher importance to one of the optimizing criterions according to the quality of the solutions in P, regard to those in Q, with respect to this criterion (see [30]).

4. A proposal of a taxonomy for the different MOACO algorithms

MOACO algorithms can be classified according to different criteria. One of them could be whether the algorithm returns a set of non-dominated solutions, i.e., if it looks for a set of Pareto solutions during its run, or it just gives a single solution as output. According to this, MOAQ, BicriterionAnt, UnsortBicriterion, BicriterionMC, P-ACO, MACS, our adaptation of MONACO, COMPETants and MO-PACO belong to the former group, that could be called *Pareto-based MOACO algorithms*, while the remainder (MACS-VRPTW, MOACOM, ACOAMO, SACO, and MOACO-ALBP) compose the second.

Another, more interesting criterion could be the fact whether the algorithms are based on the operation of only one or several ant colonies. However, we should notice that this criterion is problematic because there are different ways in which several ant colonies can be used, such as the use of several pheromone trails, several heuristic functions or, even, several independent processes as done in MACS-VRPTW.

We have decided to classify the algorithms according to two different criteria when the ant has to choose the next node to be visited:

- The use of only one or several pheromone trails.
- The use of only one or several heuristic functions.

In this way, we get the taxonomy shown in Table 1:

SACO has not been considered for the previous classification as it does not use any heuristic information at all. As seen in Section 3.11, in SACO, the ants are only guided by the pheromone trails.

An empirical study based on this taxonomy will allow us to analyze which family can generate better Pareto fronts. This way, it would be for example possible to check if the use of more than a single pheromone trail matrix, which gives a higher diversification capability to the algorithm, is more important in order to obtain a better performance than other factors such as either the use of several or only

Table 1
A taxonomy for MOACO algorithms

	A single heuristic matrix	Several heuristic matrices
A single pheromone trail matrix	MOACOM	MOAQ
	MOACO-ALBP	
	ACOAMO	MACS
Several pheromone trail matrices	P-ACO	BicriterionAnt
		UnsortBicriterion
	MONACO	BicriterionMC
		COMPETants
		MACS-VRPTW
		MO-PACO

one heuristic matrix, or the specific operation mode of each MOACO algorithm.

This way, we would be able to corroborate previous results as regards this issue such as those obtained in [32] by Iredi et al. They concluded that the performance of the BicriterionMC algorithm when using a pheromone trail matrix for each objective is better than when using a single pheromone trail matrix for all of them.

5. On the experimental study

To perform the experimental comparison, we will only consider those MOACO algorithms introduced in Section 3 whose aim is to obtain a set of non-dominated solutions for the problem being solved (Pareto-based MOACO algorithms). We have not considered the MO-PACO algorithm due to it does not follows the classical ACO heuristic, which does not consider any population of solutions. This way, eight algorithms will be used: MOAQ, BicriterionAnt, UnsortBicriterion, BicriterionMC, P-ACO, MACS, our adaptation of MONACO, and COMPETants.

The reason why the remainder are not implemented and tested in the developed study is because they are so specific to be applied to the problem tackled, the bi-objective TSP, as they are usually based on a lexicographical order, thus requiring to specify an order of importance between the two problem objectives (which is not the case). Although MOAQ also needed an order of importance, the authors remarked that it could be arbitrary, and thus this aspect is not taken into account and the algorithm is considered in our experimental setup.

On the other hand, we should notice that two of the most known, Pareto-based second generation MOGAs, NSGA-II (see Appendix A.1) and SPEA2 (see Appendix A.2) (which represent the state-of-the-art in multi-objective evolutionary optimization, see [7]), will be run on the same problem instances and their results used as *baselines* for the MOACO algorithms performance. Hence, ten different algorithms will be run on the six bi-criteria TSP instances selected.

It is important to indicate that, in order to make fair comparisons, every algorithm will store the non-dominated solutions found along its run in an external unbounded archive (there is no limit for the number of non-dominated solutions collected). However, this archive will not be used by the algorithms in order to modify their behavior. For instance, the number of ants in MOACOs or the size of P' in SPEA2 (see Appendix A.2) is constant through the run.

In addition, as seen in Section 3, the eight MOACO algorithms implemented are characterized by tackling very diverse applications. Hence, several changes have to be done in order to adapt them to solve the multi-objective TSP. Leaving apart the fact that all of them have to manage the same problem representation, the usual permutation, the different changes performed are reported in the following sections.

Finally, it is interesting to notice that no local optimizer has been added to the studied MOACO algorithms in the current contribution. The reason is that this study aims at comparing the tradeoff between diversification and intensification offered by the original algorithms themselves, i.e., by the good combination of the selection and replacement mechanism and crossover and mutation operators, characteristics of MOGAs, and the good management of problem specific and learned information, characteristic of MOACO algorithms. Of course, local search is usually considered in the latter kind of algorithms in order to optimize the quality of the final results in the TSP, but this causes a completely different behavior where it would be difficult to determine the real influence of the original algorithm search process with respect to the one of the local optimizer.

5.1. Adaptation of MOACOs to multi-objective TSP

5.1.1. MOAQ for the multi-objective TSP

In our MOAQ-TSP, the Q values corresponding to the system rewards (see Section 3.1) will be implemented as a matrix. The set of appropriate actions

$a \in A$ will be the set of possible nodes to be visited in the current step of the ant's way, and S, the set of states, will be the set of nodes of the graph.

There will be as many colonies as the number of objectives (graphs). Ants belonging to different colonies will use different heuristic functions HE^k. However, every ant will use the same pheromone matrix τ, initialized using the values of its corresponding reward function Q. The value of HE^k for edge a_{ij} will be

$$HE^k(i,j) = 1/d_{ij}^k$$

with d_{ij}^k being the cost associated to the edge a_{ij} according to the kth graph.

The reward given to the Q values of the edge a_{ij} will be

$$r(i,j) = \sum_{s \in \mathrm{NDPI}} \frac{K}{\sum_{k=1}^{K} f^k(S)},$$

where K is the number of objectives, NDPI is the set of non-dominated solutions found in the current iteration, and $f^k(S)$ is the cost of solution S according to the kth objective function.

The punishment applied to components of unfeasible solutions will not be considered as every permutation is a feasible solution in the multi-objective TSP. Finally, the jth ant of family i will build a solution trying to optimize the cost according to the ith objective function without using the solution of the jth ant of family $i-1$, due to the fact that the objectives can be optimized without taking account any specific ordering.

5.1.2. BicriterionAnt for the multi-objective TSP

BicriterionAnt uses two pheromone matrices, τ and τ'. Every ant in the non-dominated front of the current generation updates both matrices and the amount deposited is $1/l$, with l being the number of ants allowed to perform the update. In the bi-criteria TSP, τ_{ij} and τ'_{ij} are, respectively, the pheromone trail on edge a_{ij} on each matrix, and if we use the previous rule to update both pheromone matrices, both of them will have the same values. Due to this, we have changed the pheromone update rule. The amount deposited on τ by the ant h will be the inverse of the cost of its solution according to the first objective function, $f^1(S_h)$, and the amount on τ' will be the inverse of the cost according to the second objective function, $f^2(S_h)$. Usually, the cost associated to the first graph will be different to the cost associated to the second, so the amount of pheromone on edge a_{ij} will be probably different in τ and in τ'

$$\tau_{ij} = \tau_{ij} + 1/f^1(S_h); \quad \tau'_{ij} = \tau'_{ij} + 1/f^2(S_h). \tag{1}$$

5.1.3. BicriterionMC for the multi-objective TSP

We will use ten colonies in BicriterionMC-MOTSP with two pheromone matrices in every colony, as Iredi et al. did in [32] for a problem with two criteria. Besides, the pheromone trail update rule considered will be that shown in Eq. (1) in the previous section. When updating the first matrix τ, the algorithm will apply the left-hand expression. On the other hand, the right-hand expression will be used when updating the second matrix τ'.

5.1.4. P-ACO for the multi-objective TSP

In P-ACO, η_{ij} is an aggregated value of the attractiveness of edge a_{ij} which depends on the specific problem instance. For P-ACO-MOTSP, η_{ij} will be the inverse of the average of the costs of the edges for every graph

$$\eta_{ij} = \frac{K}{\sum_{k=1}^{K} d_{ij}^k} \tag{2}$$

with d_{ij}^k being the cost of edge a_{ij} according to the kth graph and K being the number of graphs.

P-ACO uses a specific rule to update the pheromone trail values. In the multi-objective TSP, we have considered that the ant which is updating a pheromone trail matrix, lays down an amount equal to the inverse of the cost of its solution according to the graph associated to the matrix

$$\Delta\tau_{ij}^k = \begin{cases} 1/f^k(\text{best}) + 1/f^k, & \text{if edge } a_{ij} \in \text{ best,} \\ \quad (\text{second-best}) & \text{and second-best solutions,} \\ 1/f^k(\text{best}), & \text{if edge } a_{ij} \in \text{ best solution,} \\ 1/f^k(\text{second-best}), & \text{if edge} \\ & a_{ij} \in \text{ second-best solution,} \\ 0, & \text{otherwise.} \end{cases}$$

5.1.5. MONACO for the multi-objective TSP

The only specification to the previously updated algorithm shown in Section 3.7 is that η_{ij} will be computed as in P-ACO-MOTSP, using Eq. (2).

5.2. TSP instances

The TSP is a very common combinatorial optimization problem that can be described as: given a

finite number of "cities" along with the cost of traveling between each pair of them, find the cheapest way of visiting all the cities once and returning to the starting point.

More formally, it can be represented by a complete weighted graph, $G = (N, A)$, with N being the set of cities and A the set of edges fully connecting the nodes N. Each edge is assigned a value d_{ij}, which is the length of edge $a_{ij} \in A$. The TSP is the problem of finding a minimal length Hamiltonian circuit of the graph, where a Hamiltonian circuit is a closed tour visiting exactly once each of the $n = |N|$ cities of G.

In the K-objective TSP, K different cost factors are defined between each pair of towns. In practical applications, the cost factors may for example correspond to cost, length, travel time or tourist attractiveness. We could say that an instance of the K-objective TSP has several graphs associated, each of them having a different cost d_{ij}^k for the same edge a_{ij}.

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 multiobjective 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 [34] for more information on them. In this study, we will use six bi-criteria TSP instances: Kroab50, Krocd50, Kroab100, Kroad100, Krobc100, and Krocd100.

5.3. Metrics of performance considered

In this contribution, we are interested on the two kind of metrics shown in Appendix A.3:

- those which measure the quality of a non-dominated solution set returned by an algorithm and,
- those which compare the performance of two different multi-objective algorithms [47].

As regards the former group, we should notice that the Pareto-optimal sets $\overline{X}$ and $\overline{Y}$, needed to compute the values of metrics M_1 and M_1^*, are not known, so we will construct a pseudo-optimal Pareto front combining all the solutions found by all the algorithms and removing the dominated ones in order to use the M_1^* metric in our study. On the other hand, we will use the M_2^* and M_3^* metrics, with $\sigma = 10{,}000$ and 20,000, depending on the size of the problem, 50 or 100, respectively (these values are close to the 10% of the Euclidean distance between the two outer solutions in the Pareto fronts obtained).

Besides, the C metric will be taken as the metric of the second group.

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

We should remark that, as the main goal of this work was to study the performance of the MOACO algorithms to generate proper Pareto fronts for the bi-criteria TSP, we have not considered the comparison of the solutions in the genotypic space. To do so, a metric for the decision space would be needed. This could be an interesting extension for future works.

5.4. Parameter settings

Each of the considered multi-objective algorithms (both MOACO and MOGAs) has been run ten times for each of the six bi-criteria TSP instances. The generic parameter values considered are so usual when applying MOACO algorithms and MOGAs to the TSP problem (see Table 2). The values of the specific parameters, such as λ in MOAQ or the number of ants for the BicriterionMC algorithm, have been obtained from the papers where the algorithms using them were defined.

To choose the initial pheromone trail value τ_0, we have considered the method used by each algorithm to update the pheromone trails. Most of them add an amount of pheromone equal to the inverse of the cost of the selected solutions, or the sum of them. Thus, the initial pheromone trail will be computed with the following rule:

$$\tau_0 = \frac{K}{\sum_{k=1}^{K} f^i(G_i)},$$

where G_i is a solution for the ith graph given by a greedy algorithm, K is the number of graphs/objectives in the current problem, and $f^i(G_i)$ is the cost of the solution G_i according to the ith graph.

However, two MOACO algorithms use a special rule in order to update the pheromone trails. Thus, they will consider a different value for τ_0: τ_0 will be 0.1 for COMPETants, as Doerner et al. did in [16], and MACS will apply the following rule to compute it:

Table 2
Parameter values considered

Parameter	Value
Number of runs for each algorithm	10
Maximum run time	300 seconds (50 nodes instances) 900 seconds (100 nodes instances)
Number of ants	20 (100 for BicriterionMC and UnsortBicriterion)
α	1
β	2
ρ	0.2
q_0	0.98 (0.99 for MOAQ)
γ (MOAQ)	0.4
λ (MOAQ)	0.9
N_C (BicriterionMC and UnsortBicriterion)	10
MOGA population size	100 (NSGA-II) 80 plus 20 in the elite population (SPEA2)
Crossover probability	0.8
Mutation probability	0.1
Computer specifications	Intel Celeron™ 1200 MHz with 256 MB RAM
Operating system	Linux Red Hat 9.0

$$\tau_0 = \frac{1}{\prod_{i=1}^{K}\left[\frac{\sum_{j=1}^{K} f^j(G_i)}{K}\right]}$$

as mentioned in Section 3.6.

6. Results and analysis

This section is devoted to report and analyze the different results obtained in the experimentation developed. In order to develop this analysis, we will consider the following aspects:

- The number of iterations and evaluations performed by the algorithms. These results will allow us to identify which algorithms are *faster* and which are *slower*.
- The graphical representation of the Pareto sets returned. These graphics will help us to understand the values for the following metrics.
- Values for the M_1^* metric. They will evaluate the distance between the Pareto sets returned and the pseudo-optimal Pareto front.
- Values for the M_2^* metric. M_2^* values will evaluate the distribution of the solutions in the Pareto sets returned.
- Values for the M_3^* metric. M_3^* values will evaluate the extent of the Pareto sets obtained.
- Values for the C metric. C values will compare the algorithms between them by calculating the dominance degree of their respective Pareto sets.
- Global analysis. Finally, we will draw some conclusions according to the previous results.

All the experimental data will be reported in the form of box-plots, where the minimum and maximum values are the lowest and highest lines, the upper and lower ends of the box are the upper and lower quartiles and a thick line within the box shows the median.

6.1. Analysis of the results

6.1.1. Statistics of the runs

Figs. 1 and 2 show the statistics of the runs considered for each algorithm in each of the six bi-criteria TSP instances. The former two graphics are associated to the two small instances, Kroab50

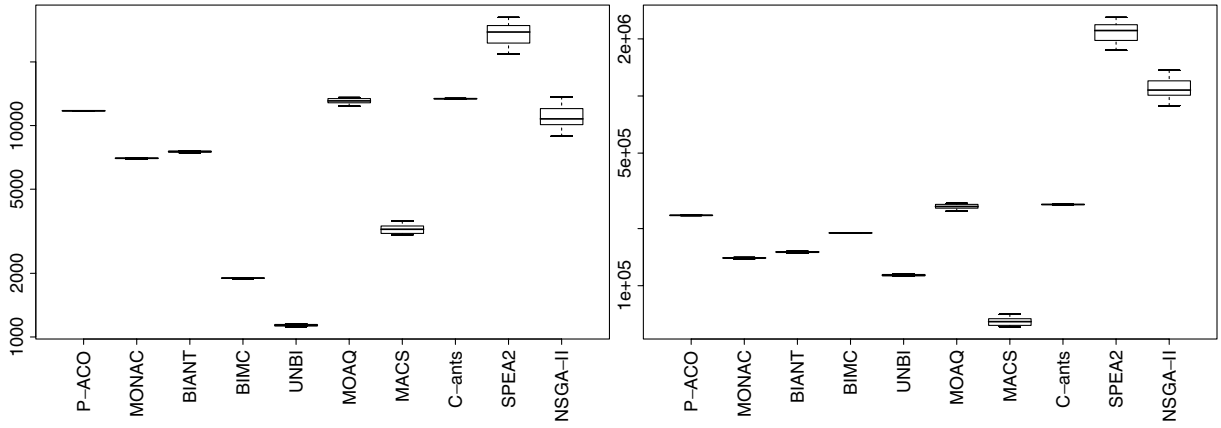

Fig. 1. Number of iterations (left) and evaluations (right) performed by the algorithms in the Kroab50 and Krocd50 runs (logarithmic scale).

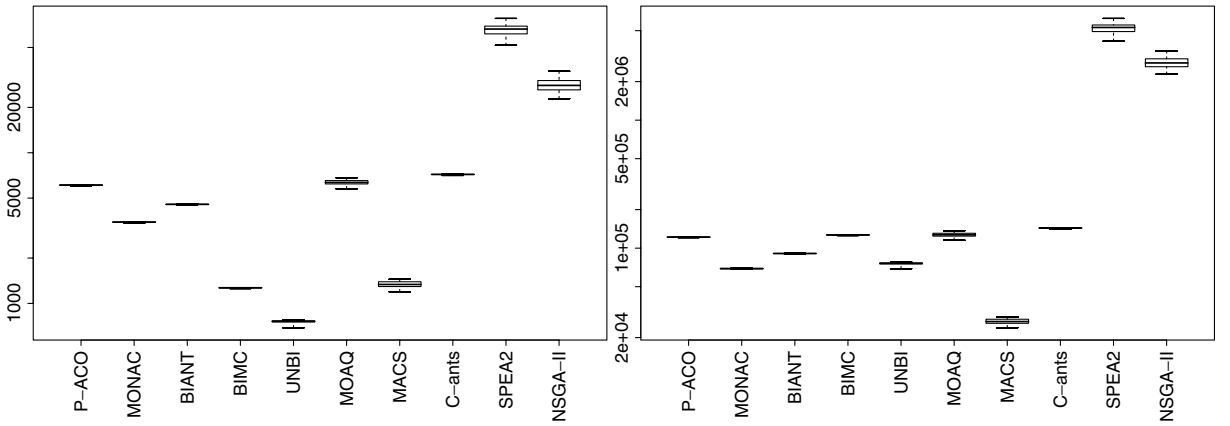

Fig. 2. Number of iterations (left) and evaluations (right) performed by the algorithms in the Kroab100, Kroad100, Krobc100 and Krocd100 runs (logarithmic scale).

and Krocd50, while the latter two correspond to the four large instances, Kroab100, Kroad100, Krobc100 and Krocd100. We have used the following abbreviations:

- *MONAC* for MONACO,
- *BIANT* for BicriterionAnt,
- *BIMC* for BicriterionMC,
- *UNBI* for UnsortBicriterion, and
- *C-ants* for COMPETants.

In view of these graphics, we can see that both MOGAs, NSGA-II and SPEA2, can usually perform much more iterations and evaluations than the MOACO algorithms considered in the same fixed run time. Hence, this shows how MOACO algorithms are somehow *slower* than MOGAs in the current problem. We can also notice that BicriterionMC and UnsortBicriterion perform less iterations whereas MACS is the algorithm which performs less evaluations than any other. The former is due to the fact that BicriterionMC and UnsortBicriterion used 100 ants while the other MOACO algorithms only handled 20 ants.

On the other hand, COMPETants, MOAQ and P-ACO are the *quickest* of the MOACO algorithms implemented, since they perform more evaluations and iterations than the remainder.

6.1.2. Visual analysis of the Pareto fronts generated

In order to graphically represent the different algorithms performance, we will proceed in the same way that Zitzler et al. did in [47]. All the Pareto sets generated by each algorithm in the 10 runs performed will be fused into a single Pareto front by removing the dominated solutions from the joined set.

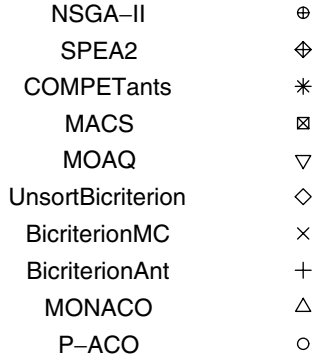

Fig. 3. Legend of Pareto front graphics (Figs. 4–9).

Let us call $\overline{P_i^j}$ the non-dominated solution set returned by algorithm i in the jth run, P_i the union of the solution sets returned by the ten runs of algorithm i ($\overline{P_i^1} \cup \overline{P_i^2} \cup \cdots \cup \overline{P_i^{10}}$), and finally $\overline{P_i}$ the set of all non-dominated solutions in P_i. We use the legend in Fig. 3. The graphics in Figs. 4 and 5, show these sets $\overline{P_i}$ for the Kroab50 and Krocd50 instances, respectively. In the same way, Figs. 6–9 relate to the sets $\overline{P_i}$ when the algorithms are applied to Kroab100, Kroad100, Krobc100 and Krocd100 instances. These graphics offers a visual information, not measurable but sometimes more useful than numeric valuations.

Looking at Figs. 4–9, we can draw some conclusions, which must be later trusted in view of the performance metrics values:

- Non-dominated solution sets returned by both MOGAs, NSGA-II and SPEA2, are poor in

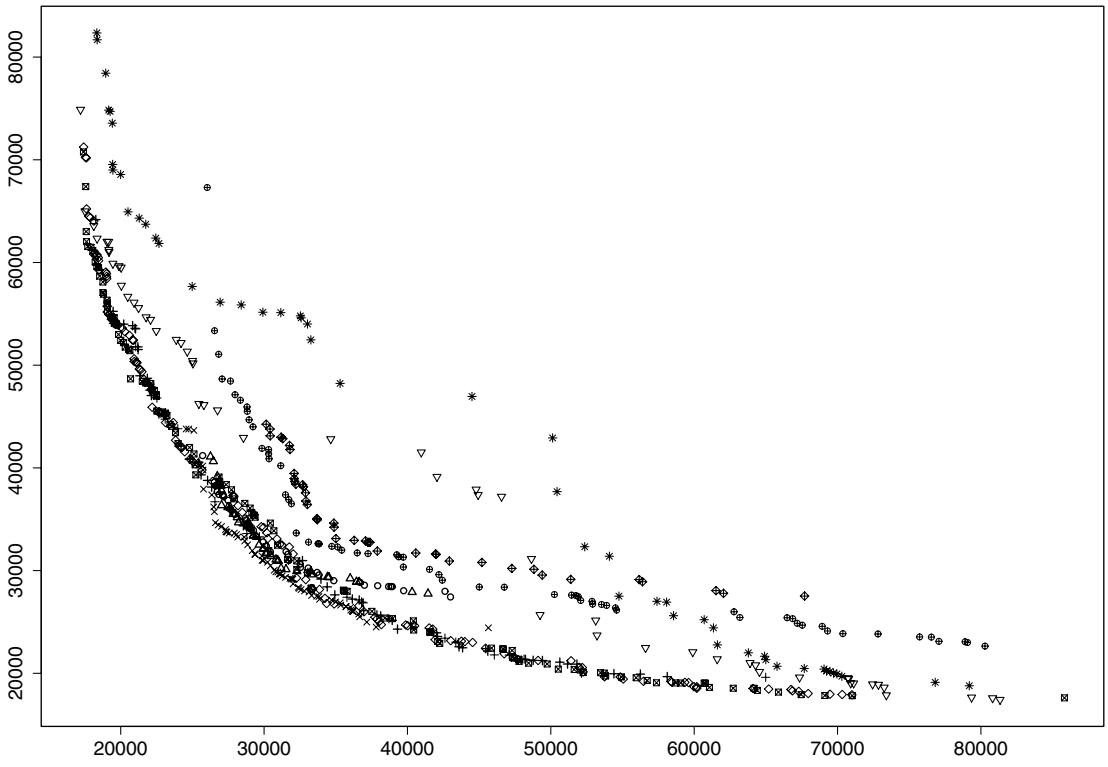

Fig. 4. Non-dominated solution sets returned by the ten runs of each algorithm when applied to the Kroab50 instance.

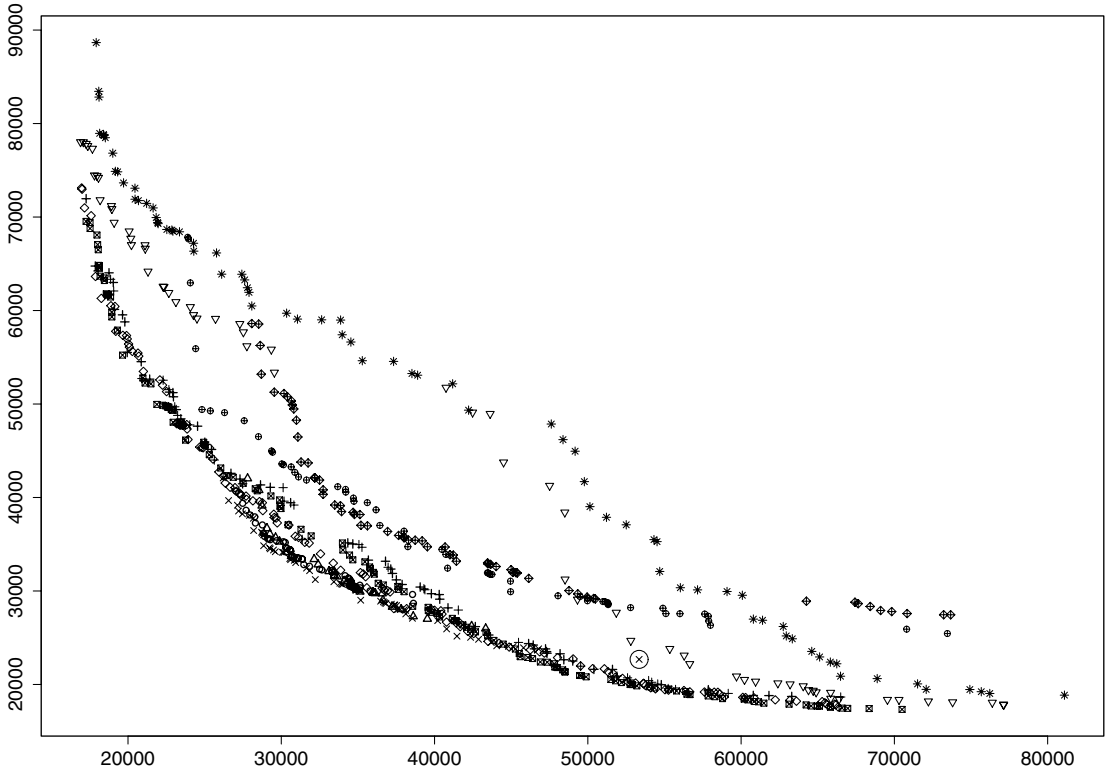

Fig. 5. Non-dominated solution sets returned by the ten runs of each algorithm when applied to the Krocd50 instance.

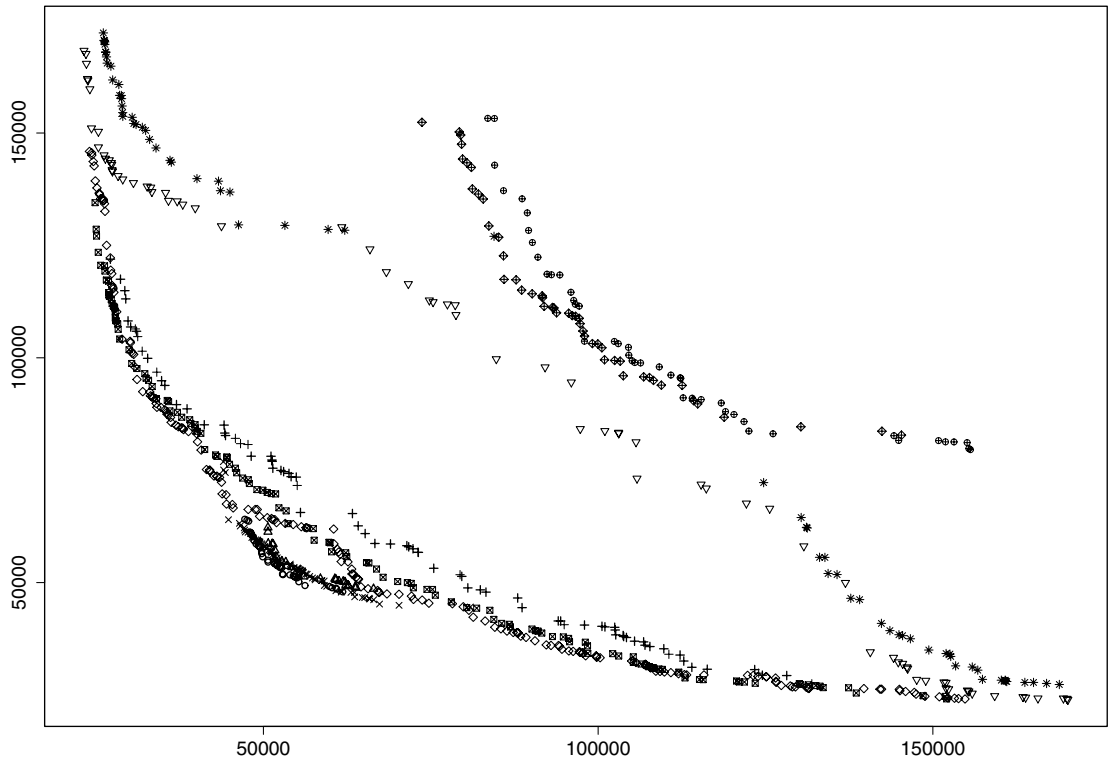

Fig. 6. Non-dominated solution sets returned by the ten runs of each algorithm when applied to the Kroab100 instance.

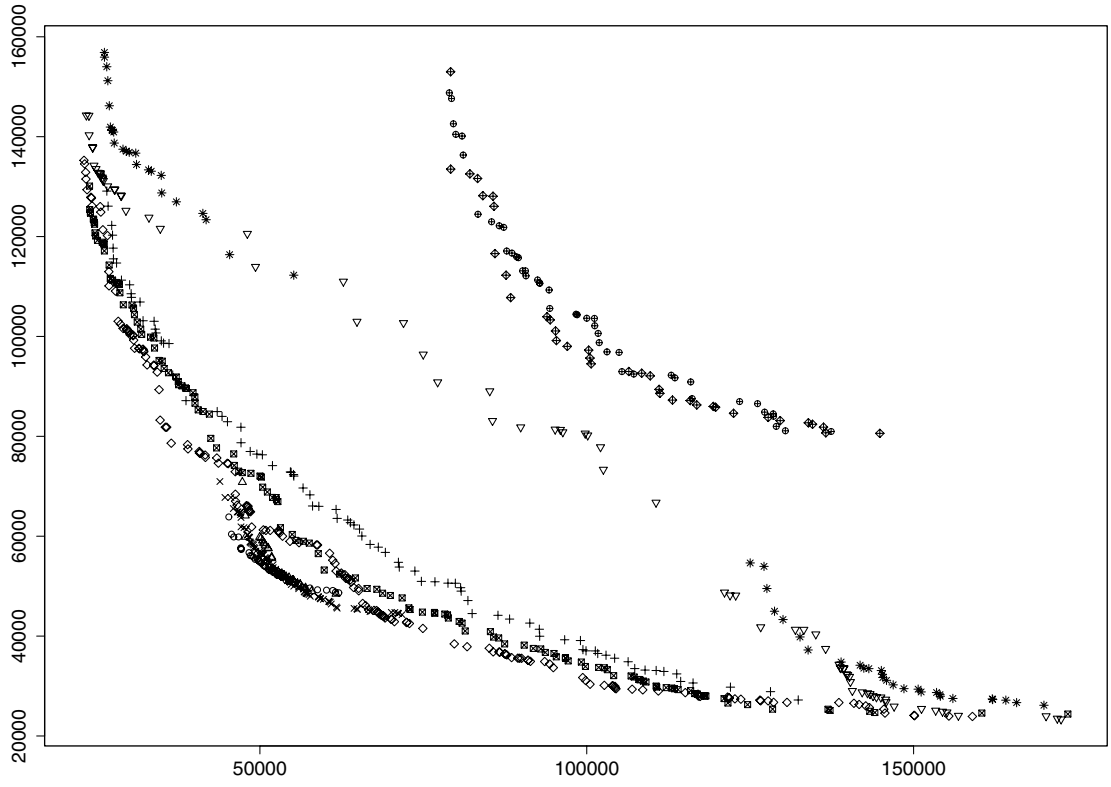

Fig. 7. Non-dominated solution sets returned by the ten runs of each algorithm when applied to the Kroad100 instance.

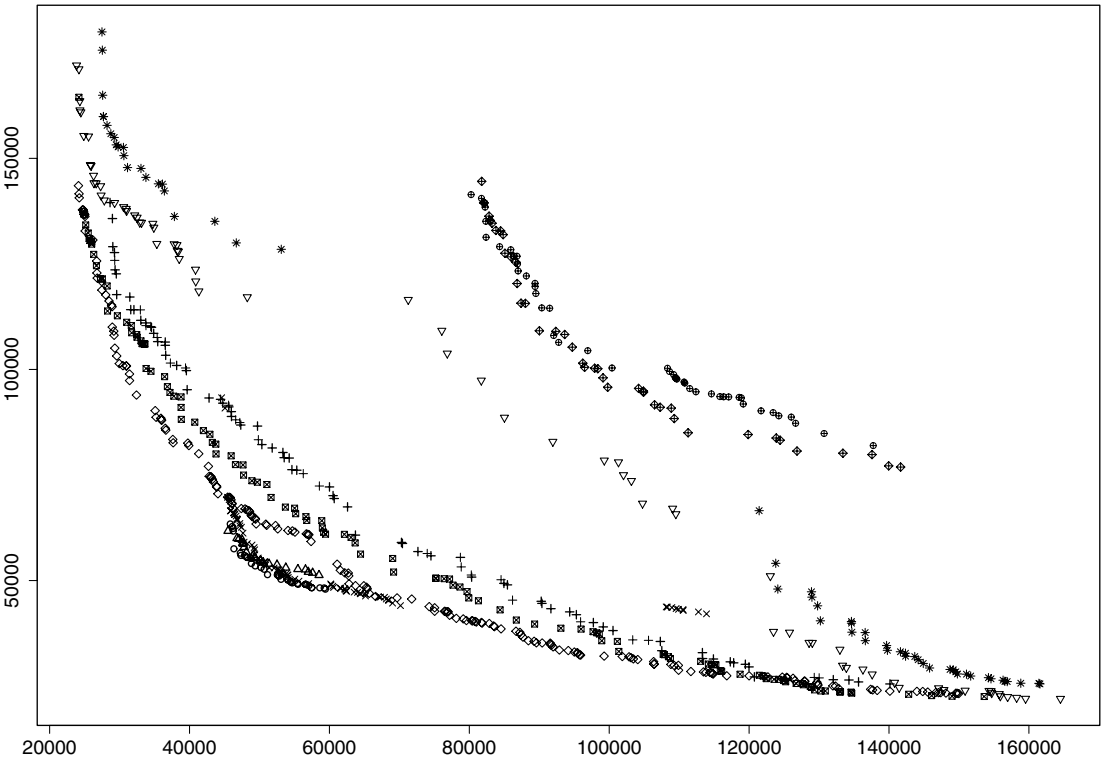

Fig. 8. Non-dominated solution sets returned by the ten runs of each algorithm when applied to the Krobc100 instance.

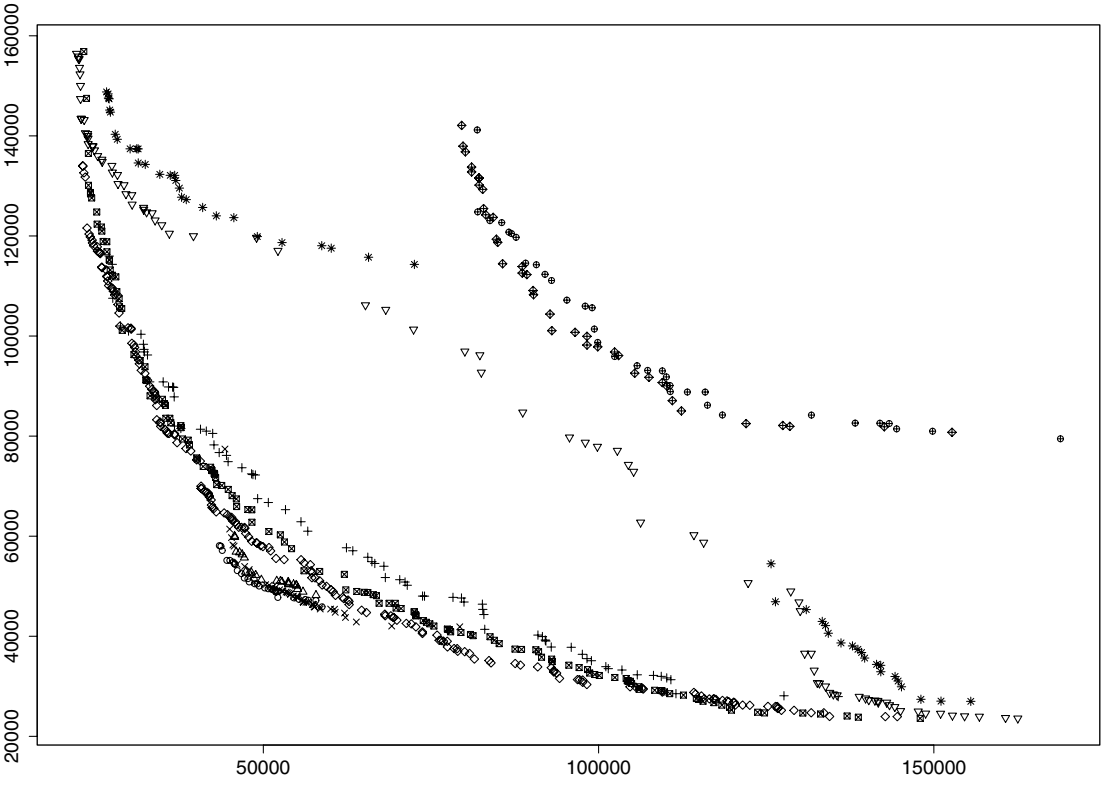

Fig. 9. Non-dominated solution sets returned by the ten runs of each algorithm when applied to the Krocd100 instance.

comparison with most of those returned by MOACO algorithms. These sets can only dominate some solutions returned by COMPETants and MOAQ when applied to the two small instances of size 50.

- COMPETants and MOAQ algorithms offer good solutions which optimize one of the two objectives. However, they are not able to cover the middle of the Pareto front in a proper way, i.e., to generate good quality solutions which establish a compromise between the different objectives. In fact, COMPETants is only able to generate solutions in the central part of the Pareto front in the two smaller problem instances considered. In the remaining four, the Pareto fronts generated converged to the two extents. This could be due to the fact that in both algorithms, the ants of a colony only consider the heuristic information associated to their own colony, and not the remainder.
- P-ACO, MONACO and BicriterionMC algorithms return very good solutions in the central part of the Pareto front but they are not able to generate any solution at all in the extents of the Pareto front. On the one hand, P-ACO and MONACO only use a single heuristic matrix, which is built by using the average of the cost of the edges for every graph. In addition, they both weight the learned information in a way that gives some automatic bias towards compromise solutions. On the other hand, BicriterionMC uses the *Update by region* method. It makes that the extreme colonies do not receive pheromone, and thus they loss it by evaporation. Finally, the ant belonging to the extreme colonies construct somehow random circuits (every τ_{ij} and τ'_{ij} are similar and close to zero), which is not sufficient in order to obtain new non-dominated solutions. In other words, it seems to loss diversity in its pheromone trails.
- Finally, BicriterionAnt and, specially, UnsortBicriterion and MACS achieve a good distribution over the Pareto front. Their solutions are sometimes dominated by those returned by P-ACO, MONACO, BicriterionMC, but only in the central part of the Pareto front.

6.1.3. Analysis of the M_1^* metric

The graphics in Figs. 10–12 show the values of the M_1^* metric obtained by each algorithm in each of the six bi-criteria TSP instances. We can see that P-ACO, MONACO, BicriterionAnt, BicriterionMC, Unsortbicriterion and MACS algorithms are those which get the lowest values on the two instances with 50 nodes (they produce non-dominated solution sets very close to the pseudo-optimal Pareto front), whereas MOAQ, COMPETants, SPEA2, and NSGA-II obtain Pareto sets far away from the pseudo-optimal one. On the other hand, the M_1^* values of the MOACO algorithms for the instances with 100 nodes behave in a similar way to the previous case. The algorithms that get the lowest values are P-ACO, MONACO, BicriterionMC, UnsortBicriterion and MACS. Both MOGAs obtain poor (high) values. This confirms what we said as regards P-ACO, MONACO, and BicriterionMC obtain high quality solutions. In addition, now we can say that the solutions obtained by BicriterionAnt, UnsortBicriterion, and MACS are also good, although they were dominated in the central region of the Pareto front.

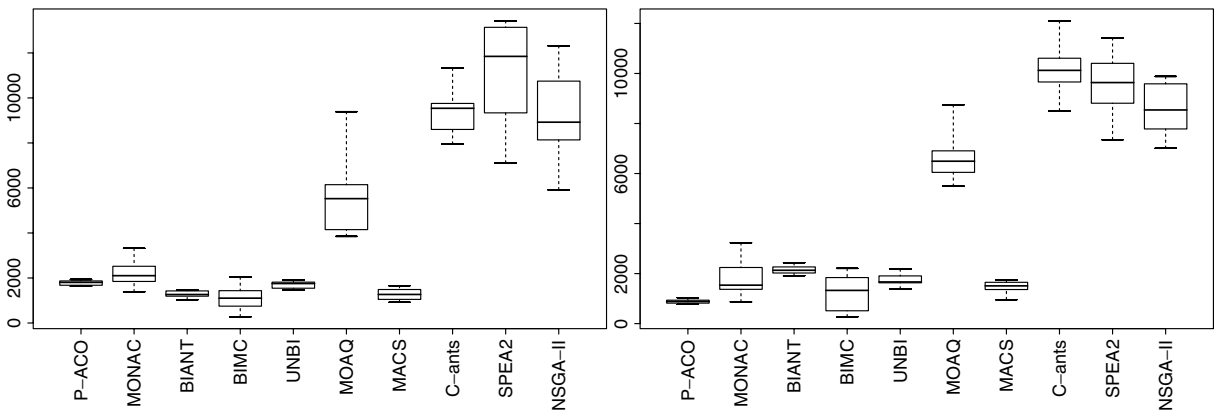

Fig. 10. M_1^* values for the algorithms in the instance Kroab50 and Krocd50.

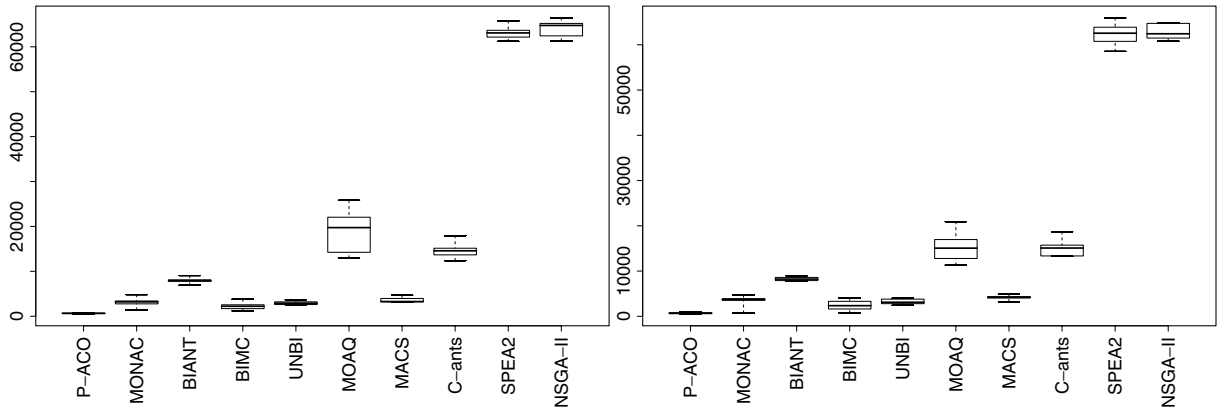

Fig. 11. M_1^* values for the algorithms in the instance Kroab100 and Kroad100.

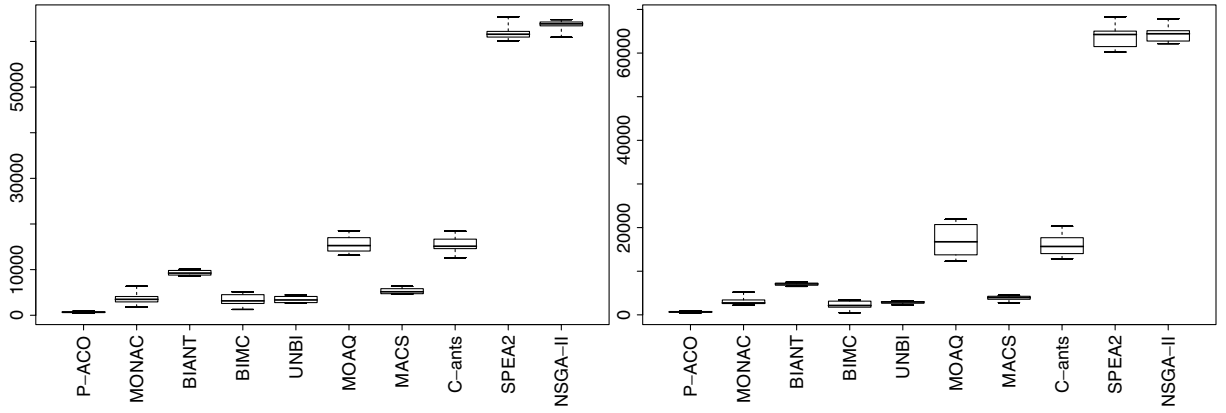

Fig. 12. M_1^* values for the algorithms in the instance Krobc100 and Krocd100.

6.1.4. Analysis of the M_2^* metric

The graphics in Figs. 13–15 show the values of the M_2^* metric obtained by each algorithm in each of the six bi-criteria TSP instances. We can see that UnsortBicriterion and MACS algorithms are those which get the highest values, whereas P-ACO, MONACO and BicriterionMC get the lowest ones. This confirms what we said as regards UnsortBicriterion and MACS cover almost the whole Pareto front whereas P-ACO, MONACO

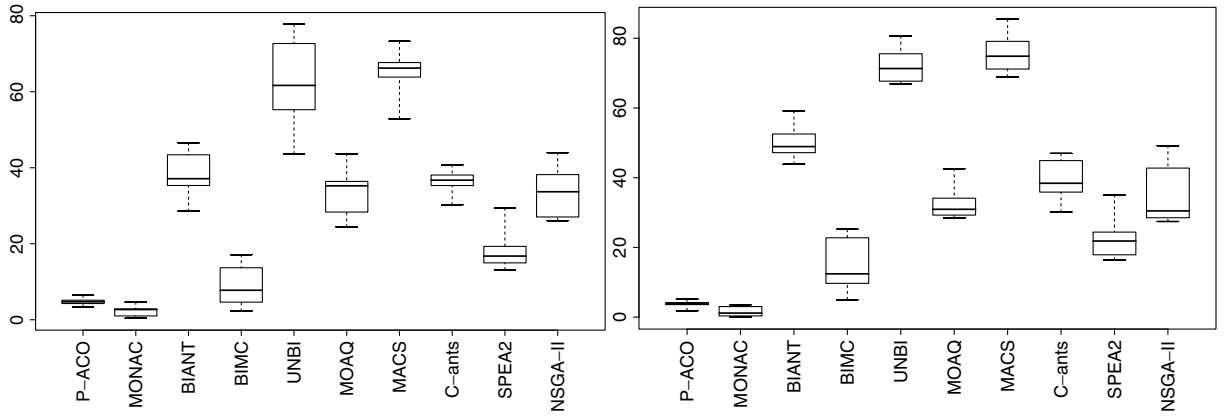

Fig. 13. M_2^* values for the algorithms in the instance Kroab50 and Krocd50.

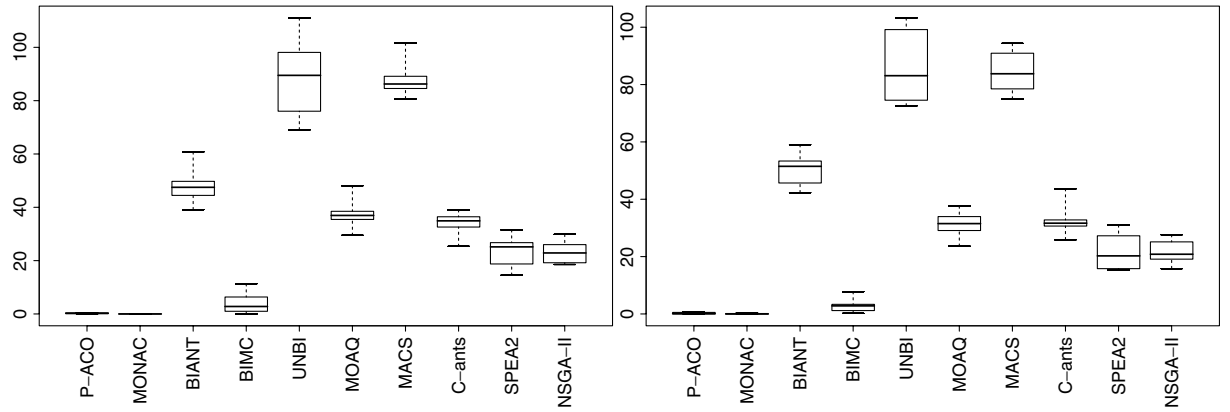

Fig. 14. M_2^* values for the algorithms in the instance Kroab100 and Kroad100.

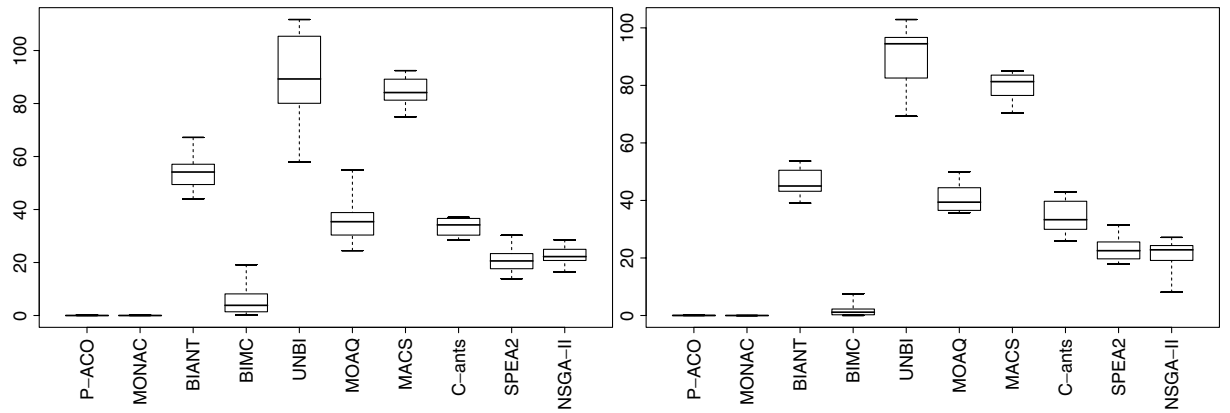

Fig. 15. M_2^* values for the algorithms in the instance Krobc100 and Krocd100.

and BicriterionMC only obtain solutions in the central part. In addition, the values obtained by the MOGAs, NSGA-II and SPEA2, are low, specially in the instances with 100 nodes. They are only a little bit higher than those obtained by MONACO, P-ACO and BicriterionMC.

6.1.5. Analysis of the M_3^* metric

Figs. 16–18 show the values of the M_3^* metric obtained by each algorithm in each of the six bi-criteria TSP instances. In view of these figures, the algorithms which get the higher values are COMPETants, MOAQ, MACS and UnsortBicriterion. On the opposite, P-ACO and MONACO achieve very low measures in the current problem. We should remark how, when analyzing Figs. 4–9, we recognized that non-dominated solution sets returned by P-ACO and MONACO were located in a small region of the objective space, whereas the Pareto fronts returned by COMPETants, MOAQ and MACS were properly spread off.

Finally, it is interesting to remark the special behavior of BicriterionMC. Notice that it obtains relatively higher M_3^* values in the Krocd50, Kroab100 and Kroad100 instances than P-ACO and MONACO, what indicates that, in these instances and in some runs, BicriterionMC returned some non-dominated solutions far away from the central part of the Pareto front. They may not be present in Figs. 5–7 because they could be dominated by other solution from other run. However, one of them can be identified in Fig. 5 near the point (54,000, 24,000) (highlighted with a circle).

6.1.6. Analysis of the C metric

The graphics in Fig. 19 are box-plots based on the C metric. Each rectangle contains six box-plots representing the distribution of the C values for a certain ordered pair of algorithms. From left to right, the leftmost box-plot refers to Kroab50, the second relates to Krocd50, the third to Krobc100, the fourth to Kroad100, the fifth to Kroab100

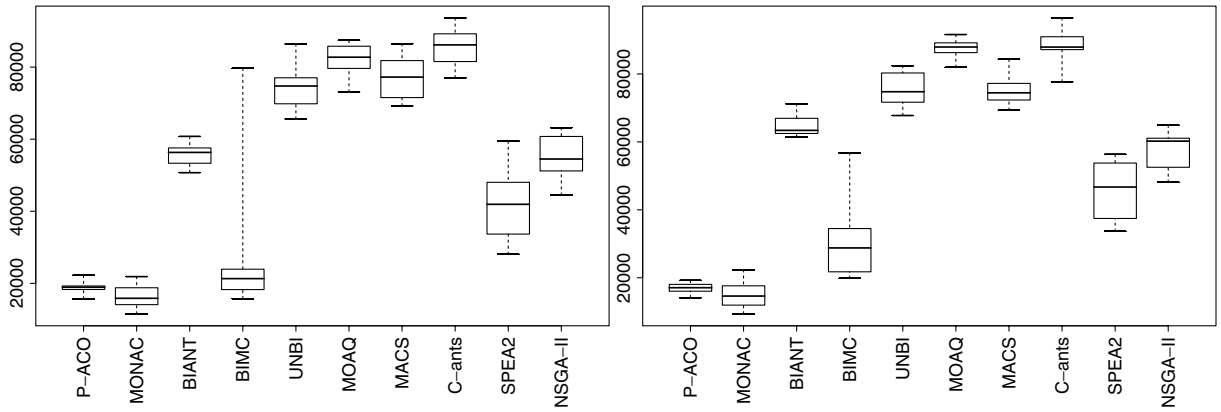

Fig. 16. M_3^* values for the algorithms in the instance Kroab50 and Krocd50.

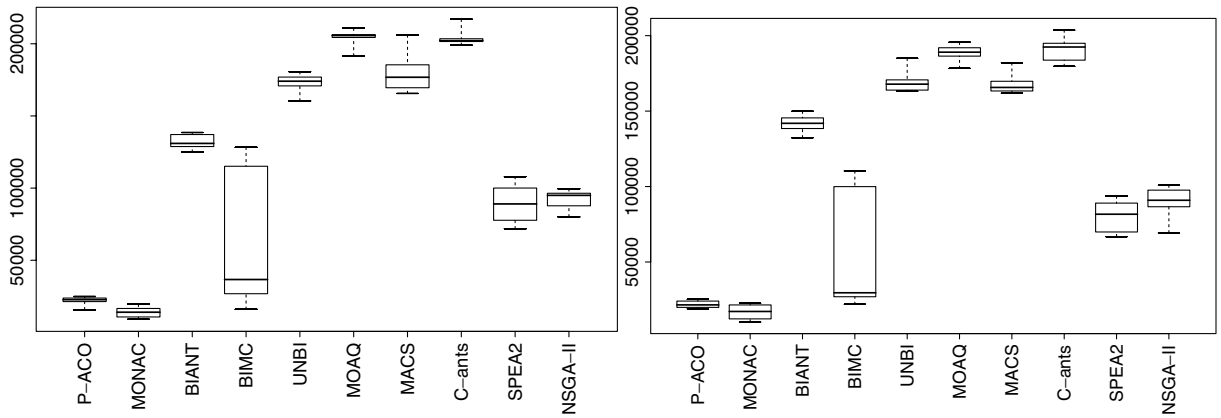

Fig. 17. M_3^* values for the algorithms in the instance Kroab100 and Kroad100.

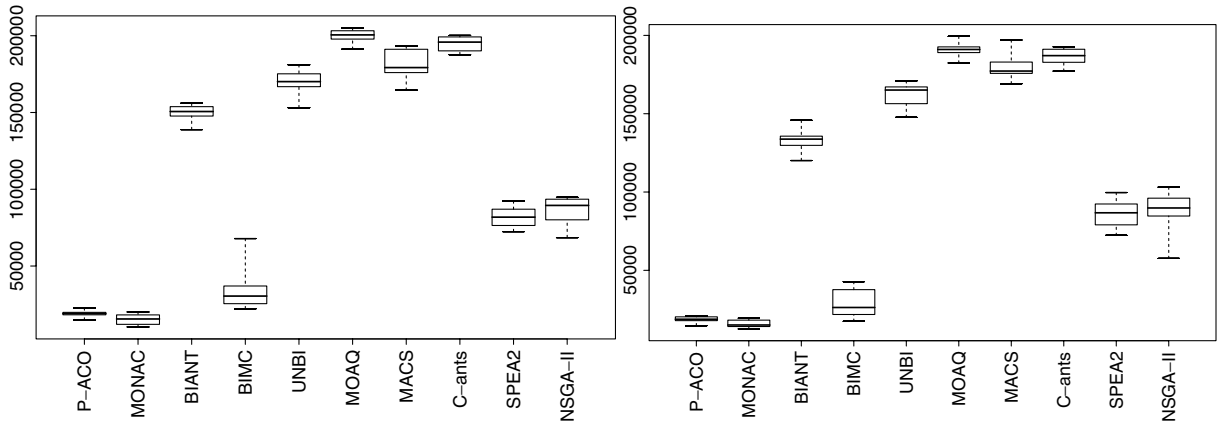

Fig. 18. M_3^* values for the algorithms in the instance Krobc100 and Krocd100.

and the rightmost to Krocd100. The scale is 0 at the bottom and 1 at the top per rectangle. Furthermore, 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,

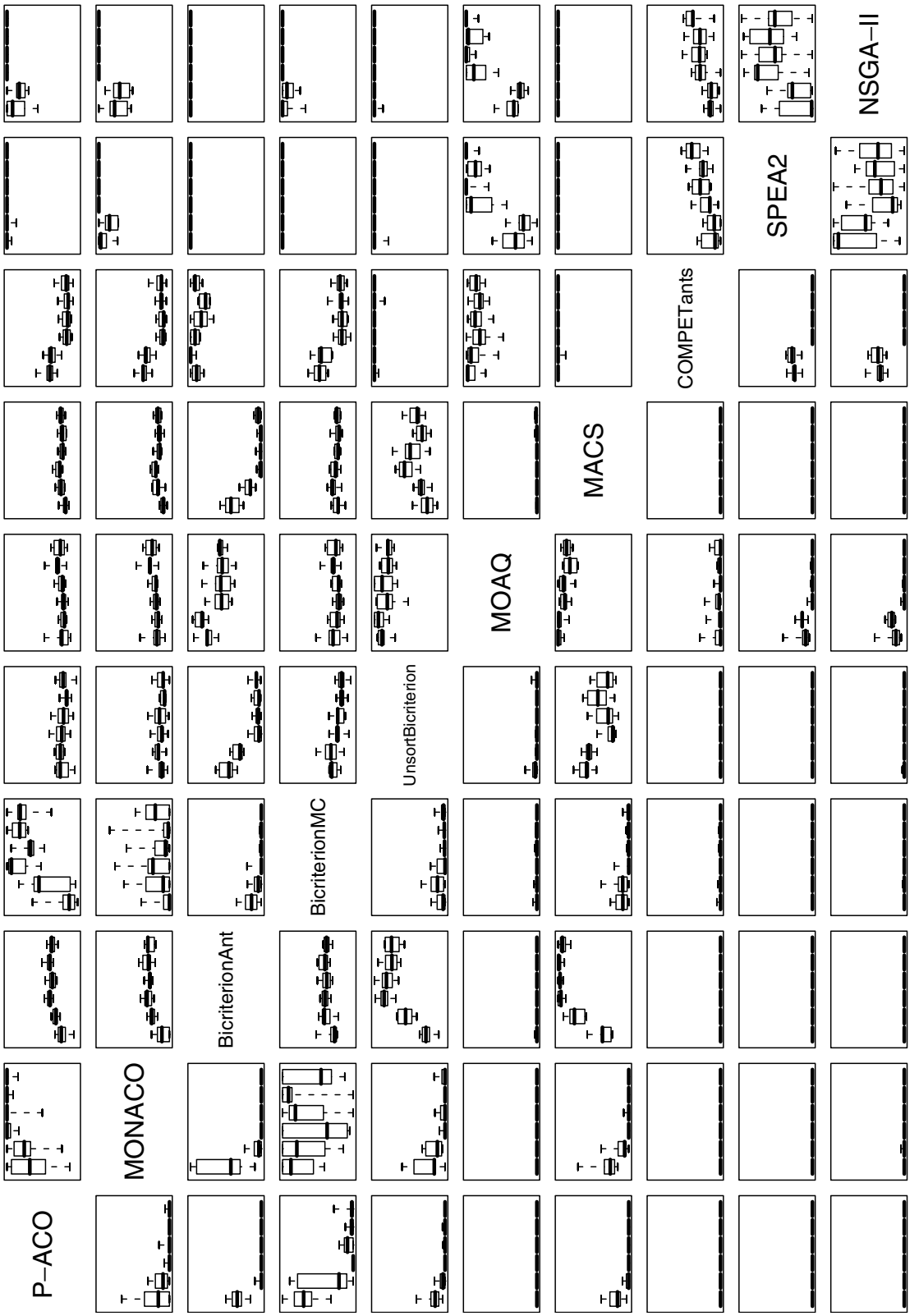

Fig. 19. Box-plots of the results obtained in the *C* metric.

for instance, the top right box, which represents the fraction of solutions of NSGA-II covered by the non-dominated sets produced by the P-ACO algorithm.

At the end of the analysis, we can draw the following conclusions:

- 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 NSGA-II and SPEA2 do not dominate the fronts given by MOACO algorithms.
- It is not easy to say which MOACO algorithm performs best, as many of them derive Pareto sets of similar quality, as regards the C metric values. These algorithms usually give non-dominated solution sets as output which do not dominate those generated from other algorithms of the same family, but they are not dominated either. However, we notice that the Pareto fronts returned by P-ACO, MONACO and BicriterionMC are practically not dominated by those obtained from the remaining algorithms. So we could say that P-ACO, MONACO and BicriterionMC are the algorithms with the best performance according to this metric, because their Pareto fronts are not dominated (see the box-plots in the first, second and fourth columns), although they actually do not dominate very well the remainder of the MOACO algorithms (see those in the first, second and fourth rows). Between them, P-ACO seems to perform better than the two others.
- However, it is easier to identify the algorithms which usually return Pareto sets of bad quality for the problem, when they are compared to those returned by the remaining algorithms considered. This is the case of SPEA2, NSGA-II, COMPETants, and MOAQ. Their Pareto fronts do not usually dominate those generated by the other algorithms while they are normally well-dominated.

6.2. Global analysis

This section is devoted to draw some general conclusions summarizing all the analysis developed. The conclusions obtained are shown as follows.

The MOACO algorithms considered are a good choice to solve the bi-objective TSP, performing better than the MOGAs implemented. Notice that, although NSGA-II is one of the state-of-the-art MOGAs for continuous optimization, it does not seem to be well-suited to the TSP since tours can not be well-represented for the application of crossover.

Besides, it is interesting to notice that the results obtained are robust with respect to the six instances considered, regardless the instance itself and its particular composition. This way, the MOACO algorithms get similar statistic values in the Pareto fronts derived regardless the concrete instance.

Finally, it is difficult to relate the performance of the MOACO algorithms, when tackling the TSP, according to the taxonomy introduced in Section 4. For example, MOAQ, based on a single pheromone trail matrix and two heuristic matrices, and COMPETants, based on the use of two pheromone trail and heuristic matrices, perform in the same way generating Pareto fronts of good quality in the extremes but badly cover the central parts. Opposite to what can be thought, the Pareto fronts generated by MOAQ dominate those got from COMPETants, although the former considers a single pheromone trail matrix and the latter two of them.

On the other hand, MACS and BicriterionAnt, which showed very good behavior, also belong to the two latter, different families (a single pheromone matrix and two heuristic matrices and two of both matrices, respectively). Again, the Pareto fronts of the former, which only considers a single pheromone matrix, outperforms those of the latter as regards the three metrics considered.

Nevertheless, we may say that the use of only one heuristic matrix, which could be obtained by averaging the different objectives when tackling the TSP, makes the MOACO algorithm obtain very good compromise solutions and forget the extreme regions of the Pareto front. This is the case of P-ACO and MONACO.

This way, it seems that the operation mode of the MOACO algorithm itself is more important for performance purposes than the number of pheromone trail matrices considered by it in the problem instances tackled.

However, there is a huge diversity in the behavior of the different MOACO algorithms analyzed. As seen in the Pareto fronts graphical representations collected in Section 6.1.2, and later corroborated

as regards the M_1^*, M_2^* and M_3^* metric values in Sections 6.1.3–6.1.5, we can find three different behaviors:

- Algorithms obtaining well-spread Pareto fronts in the objective space extents but with a very bad covering of the central part of the Pareto fronts, like MOAQ and COMPETants. This is clearly checked by the high values of both algorithms in metric M_3^* together with their low values in metric M_2^*, which show a good extent of the Pareto front obtained but a bad distribution of the non-dominated solutions. These two algorithms share the interesting property that most or all their ants only use one of the two heuristic matrices.
- Algorithms like P-ACO, MONACO and BicriterionMC with the opposite behavior, i.e., generating very good solutions in the central part of the Pareto front by obtaining fronts with a very low extent (with the corresponding lowest values in the M_1^*, M_2^* and M_3^* metrics). The two former algorithms only use one heuristic matrix which is built from the average of the two objectives, whereas BicriterionMC seems to loss diversity in its pheromone trail matrices and, finally,
- algorithms with a good generation of the non-dominated solution sets, with a proper trade-off between the central and extreme parts of the Pareto surface. This is the case of MACS, UnsortBicriterion and BicriterionAnt, as can be seen in view of their very low values in M_1^* metric (showing high quality in their solutions), highest values in metric M_2^* (showing the best distribution of solutions in the front) and their good values in the M_3^* one (showing a large extent covered).

Between these three algorithms, we can consider that MACS and UnsortBicriterion outperform BicriterionAnt as they obtain better values in M_2^*, M_3^* and C metrics than the latter. However, we can not easily state which algorithm is the best between MACS and UnsortBicriterion, because they both return very good and similar values for every metric of performance. In addition, with regard to the C metric, it does not seem that the solutions of one algorithm clearly dominate the ones of the other, as slight differences are only observed favouring each of them in different instances.

This way, the algorithms of the latter group seem to be the best choice for the current problem with the only drawback that P-ACO, MONACO and BicriterionMC generate better solutions than them in the central parts of the Pareto fronts (see Fig. 20). When comparing the latter group algorithms with those generating well-spread Pareto fronts such as MOAQ and COMPETants, it can be seen how this is clearly due the fact that the latter algorithms do not converge to the pseudo-optimal Pareto front. An example of this behavior is shown in Fig. 21.

The only case we found where the algorithms of a family behaves in the same way is that of P-ACO and MONACO, characterized by two pheromone matrices and a single heuristic matrix. Both algorithms achieve a good convergence to the central parts of the Pareto fronts, avoiding the generation of solutions in the extents of the front. However,

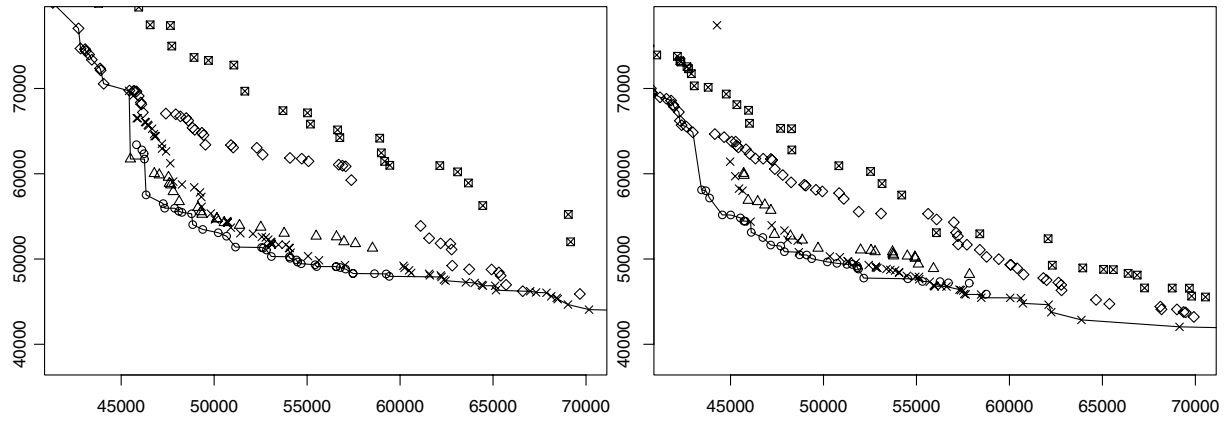

Fig. 20. Central parts of Pareto fronts of P-ACO, MONACO, BicriterionMC, MACS and UnsortBicriterion in the Krobc100 (left) and Krocd100 (right) instances. The line which joins the overall non-dominated solutions represents the pseudo-optimal Pareto front.

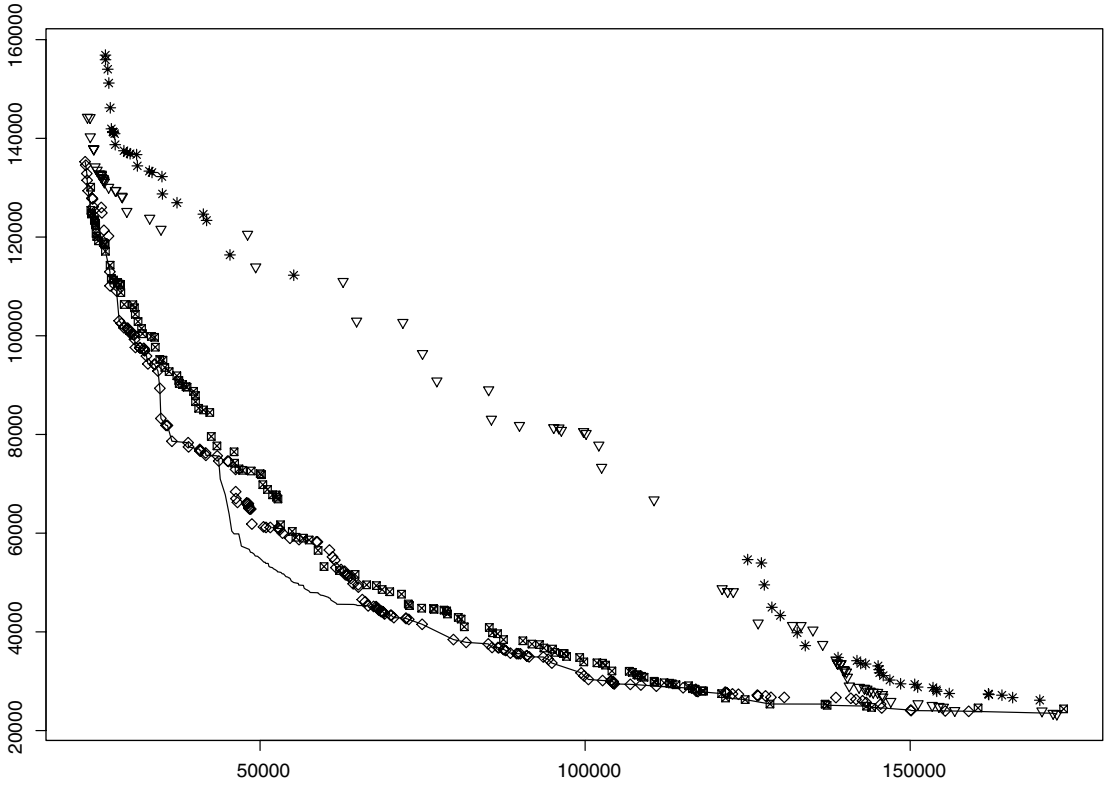

Fig. 21. Pareto fronts of MACS, UnsortBicriterion, MOAQ, and COMPETants in the Kroad100 instance. The line which joins the overall non-dominated solutions represents the pseudo-optimal Pareto front.

the existence of just two MOACO algorithms within the family could be not enough to consider this as a generic behavior.

On the other hand, it could be argued that MOAQ and MACS, both belonging to the single pheromone-several heuristic matrices family, show a similar behavior as regards the values obtained in metric M_3^*, where the two MOACO algorithms always present high values. However, these high values actually correspond to different kinds of Pareto fronts as, in view of the figures shown in Section 6.1.2, while MOAQ generates good non-dominated solution sets in the extremes of the Pareto fronts, but does not cover their central parts, MACS obtains a well-distributed set all over the fronts.

7. Concluding remarks and future works

In the current contribution, we have classified the existing Pareto-based MOACO algorithms into a taxonomy and developed an experimental study comparing their performance 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, SPEA2 and NSGA-II. Besides, we have drawn the conclusion that belonging to a specific family is not enough to achieve good performance but the operation mode of the ACO algorithm itself is more determinant for the quality of the Pareto fronts generated.

Several ideas for future developments arise from this study: (i) to analyze the influence of adding local search optimizers to the MOACO algorithms, as usually done in ACO to solve the TSP, and to compare the performance of the resulting techniques against that of memetic algorithms such as Jaszkiewicz's MOGLS [34], as some authors have recently started to do for the Quadratic Assignment Problem (where the use of local optimizers is mandatory to achieve good performance when using ACO algorithms) in works such as [36]; and (ii) to study the performance of MOACO algorithms in other complex multi-objective combinatorial optimization problems.

Appendix A. Evolutionary multi-objective optimization

Evolutionary computation uses computational models of evolutionary processes as key elements in the design and implementation of computer-based problem solving systems. There is a variety of evolutionary computational models that have been proposed and studied which are referred as evolutionary algorithms (EAs) [2]. Concretely, four well-defined EAs have served as the basis for much of the activity in the field: genetic algorithms (GAs) [40], evolution strategies [43], genetic programming (GP) [35] and evolutionary programming [23].

An EA maintains a population of trial solutions, imposes random changes to these solutions, and incorporates selection to determine which ones are going to be maintained in future generations and which will be removed from the pool of trials. But there are also important differences between them. Focusing on the kind of EA considered on this paper, GAs emphasize models of genetic operators as observed in nature, such as crossover (recombination) and mutation, and apply these to abstracted chromosomes with different representation schemes according to the problem being solved.

EAs are very appropriate to solve multi-objective problems as, thanks to the use of a population of solutions, EAs can search many Pareto-optimal solutions in the same run. Generally, multi-objective EAs (MOEAS) [7,13] only differ from the rest of EAs in the fitness function and/or in the selection mechanism.

The evolutionary approaches in multi-objective optimization can be classified in three groups: *plain aggregating approaches, population-based non-Pareto approaches and Pareto-based approaches* [13,7].

The first group constitutes the extension of classical methods to EAs. The objectives are artificially combined, or aggregated, into a scalar function according to some understanding of the problem, and then the EA is applied in the usual way. Optimizing a combination of the objectives has the advantage of producing a single compromise solution but some of the problems found in the classical multi-objective approaches (see Section 2.2) remain: (i) it can be difficult to define the combination weights in order to obtain acceptable solutions, and (ii) if the optimal solution generated can not be finally accepted, new runs of the EA may be required until a suitable solution is found.

Population-based non-Pareto approaches allow us to exploit the special characteristics of EAs. A non-dominated individual set is obtained instead of only one solution. In order to do so, the selection mechanism is changed. Generally, the best individuals according to each of the objectives are selected, and then these partial results are combined to obtain the new population. An example of a multi-objective GA of this group is vector evaluated genetic algorithm (VEGA) [42].

Finally, *Pareto-based approaches* seem to be the most active research area on multi-objective EAs nowadays. In fact, algorithms included within this family are divided in two different groups: first and second generation [7]. They all attempt to promote the generation of multiple non-dominated solutions, as the former group, but directly making use of the Pareto-optimality definition (see Section 2.2).

This way, to calculate the probability of reproduction of each individual in this approach, the solutions are compared by means of the dominance relation. Different equivalence groups are defined depending on the dominance of their constituent individuals among the remainder and those individuals belonging to the "good" classes (those groups including individuals dominating a large number of the remainder) are assigned a higher selection probability than those in the "bad" classes.

The difference between the first and the second generation Pareto-based approaches arise on the use of elitism. Algorithms included within the first generation group [7,13], such as Niched Pareto Genetic Algorithm (NPGA), Non-dominated Sorting Genetic Algorithm (NSGA) and Multiple-Objective Genetic Algorithm (MOGA), do not present this characteristic. On the other hand, second generation Pareto-based multi-objective EAs are based on the consideration of an auxiliary population where the non-dominated solutions generated among the different generations are stored. Examples of the latter family are Strength Pareto EA (SPEA) and SPEA2 [48] and NSGA-II [15], among others. As can be seen, several of the latter algorithms are elitist versions of the corresponding first generation ones.

Finally, it is important to notice that, although the Pareto-based ranking correctly assigns all non-dominated individuals the same fitness, it does not guarantee that the Pareto set is uniformly sampled. When multiple equivalent optima exist, finite populations tend to converge to only one of them, due to

stochastic errors in the selection process. This phenomenon is known as genetic drift [14]. Since preservation of diversity is crucial in the field of multi-objective optimization, several multi-objective EAs have incorporated the *niche* and species concepts [27] for the purpose of favoring such behavior.

The next two subsections are devoted to introduce the two second generation Pareto-based MOGAs considered in this contribution as baselines for the MOACO algorithms implemented, NSGA-II and SPEA2.

A.1. Non-dominated sorting genetic Algorithm II

The so-called non-dominated sorting genetic Algorithm II (NSGA-II) was designed by Deb et al. in [15] to alleviate three difficulties of MOEAs which use non-dominated sorting and sharing: (i) $O(K \cdot N^3)$ computational complexity (where K is the number of objectives and N is the population size); (ii) nonelitist approach; and (iii) the need of specifying a sharing parameter.

NSGA-II uses two key concepts, ranking and crowding-distance. In order to define the former, NSGA-II classifies every chromosome in the population P in several fronts. The first front is composed of the non-dominated solutions. Chromosomes belonging to the second front are solutions which are only dominated by at least one solution in the first front. Those belonging to the third front are solutions only dominated by solutions in the first front and at least one of the second, and so on. Finally, the ranking of the chromosome p, p_{rank}, is the index of the front which the chromosome p belongs to.

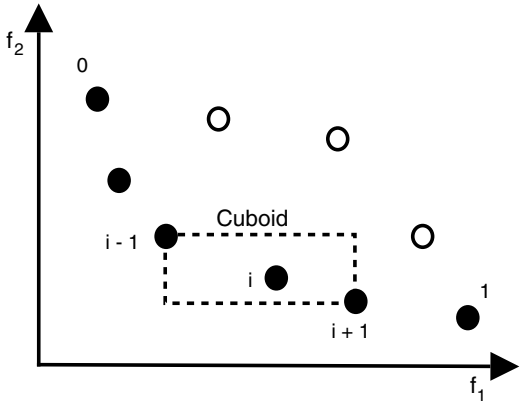

Fig. 22. Crowding-distance calculation. Points marked in filled circles are solutions of the same non-dominated front. Figure taken from [15].

On the other hand, the crowding-distance is a value which estimates the density for every chromosome in the population P. It is the average side length of the cuboid showed in Fig. 22. The crowding-distance of the chromosome p will be called p_{distance}.

According to the crowding-distance, NSGA-II defines the Crowded-Comparison Operator, $\prec_n$. Given two chromosomes p and q, $p \prec_n q$ if $p_{\text{rank}} < q_{\text{rank}}$ or $p_{\text{rank}} = q_{\text{rank}}$ and $p_{\text{distance}} > q_{\text{distance}}$.

Finally, the main loop of the algorithm is as follows. Initially, a random population P_0 is created. The population is sorted based on the non-dominance criterion. Each solution is assigned a fitness equal to its non-dominance level, p_{rank}. Thus, minimization of fitness is assumed. At first, the usual binary tournament selection, recombination, and mutation operators are used to create an offspring population Q_0 of size N, the number of chromosomes in P_0. Since elitism is introduced by comparing the current population with the best non-dominated solutions previously found, the procedure is different after the initial generation.

Then, at every iteration, a combined population $R_t = P_t \cup Q_t$ is formed. The population R_t is of size $2 \cdot N$. Then, R_t is classified in fronts according to dominance. Since all previous and current population members are included in R_t, elitism is ensured. Now, if the size of the first front in the combined population is smaller than N, NSGA-II chooses all the members from that front for the new population. The remaining population members are chosen from subsequent non-dominated fronts in the order of their ranking. Thus, solutions from the second front are chosen next, followed by solutions from the third front, and so on. This procedure is continued until no more sets can be accommodated. Let us say that the ith front is the last non-dominated set beyond which no other set can be accommodated. To choose exactly N population members, we sort the solutions of the ith front using the crowded-comparison, $\prec_n$, operator in descending order and choose the best solutions needed to fill all the new population slots. The NSGA-II procedure is also graphically illustrated in Fig. 23. The new population P_{t+1} of size N is now used for selection, crossover, and mutation to create a new population Q_{t+1} of size N. Notice that we use a binary tournament selection operator but the selection criterion is now based on the crowded-comparison operator, $\prec_n$. Since this operator requires both the rank and crowded distance of each solution in the population,

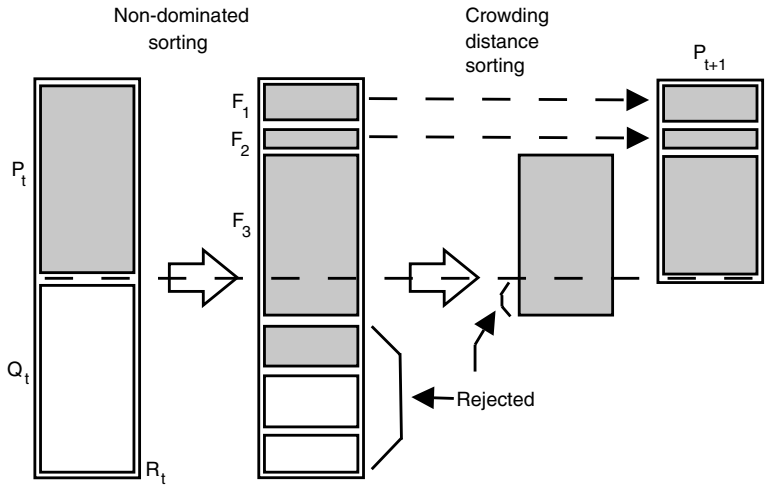

Fig. 23. NSGA-II procedure. Figure taken from [15].

we calculate these quantities while forming the new population, as shown in Fig. 23.

A.2. Strength Pareto EA2

Strength Pareto EA2 (SPEA2), proposed by Zitzler et al. in [48], was developed as an improvement of SPEA [49]. In contrast to its predecessor, SPEA2 incorporates a fine-grained fitness assignment strategy, a density estimation technique, and an enhanced archive truncation method.

This algorithm uses two separate populations of chromosomes, P with N chromosomes as the current population, and P' which will contain the N' best chromosomes found so far.

At every iteration, the algorithm copies all the non-dominated individuals in P_t and P'_t to P'_{t+1}. If the size of P'_{t+1} exceeds N' then it reduces P'_{t+1} by means of the truncation operator; otherwise if the size of P'_{t+1} is less than N' then it fills P'_{t+1} with dominated individuals in P_t and P'_t. Then, it performs a binary tournament selection with replacement on P'_{t+1} in order to fill a mating pool. Finally, the algorithm applies the recombination and mutation operators to the mating pool and set P_{t+1} to the resulting population.

In order to assign a fitness value to every chromosome p of populations P and P', the algorithm uses the following rule:

$$\text{fitness}(p) = R(p) + D(p).$$

In this equation, $R(p)$ is the raw fitness of chromosome p, and $D(p)$ its density. The raw fitness, R, is calculated as follows:

$$R(p) = \sum_{i \succ p} S(i); \quad S(i) = |j; j \in P \cup P' \wedge i \succ j|,$$

where $i \succ j$ indicates that solution i dominates solution j. $S(i)$ is called the strength of solution i.

At the other hand, $D(p)$ is computed using an adaptation of Silverman's kth nearest neighbor method [44]. The algorithm calculates $D(p)$ as follows:

$$D(p) = 1/(\sigma_p^k + 2),$$

where σ_p^k is the distance between the chromosome p and its kth nearest neighbor. Usually k is equal to $\sqrt{N + N'}$.

When the size of P'_{t+1} is lesser than N', the algorithm fills this external population with the best solutions in $(P_t \cup P'_t) - P'_{t+1}$, according to their fitness values. Otherwise, if the size of P'_{t+1} is greater than N', the algorithm iteratively removes individuals from P'_{t+1} until $|P'_{t+1}| = N'$. Here, at each iteration, the algorithm removes the individual i, with $i \leqslant_d j$ for all $j \in P'_{t+1}$ where

$$i \leqslant_d j \iff \begin{array}{l} \forall k,\ 0 < k < |P'_{t+1}| \ : \ \sigma_i^k = \sigma_j^k \ \vee \\ \exists k,\ 0 < k < |P'_{t+1}| \ : \ [(\forall l,\ 0 < l < k \\ \quad : \ \sigma_i^l = \sigma_j^l) \wedge \sigma_i^k < \sigma_j^k] \end{array}$$

with σ_i^k denoting the distance of i to its kth nearest neighbor in P'_{t+1}.

A.3. Metrics of performance

Experimentally comparing different optimization techniques always involves the notion of performance. In the case of multi-objective optimization,

the definition of quality is substantially more complex than for single-objective optimization problems, because the optimization goal itself consists of multiple objectives [7]:

- The distance of the resulting non-dominated set to the real Pareto-optimal front should be minimized.
- A good (in most cases uniform) distribution of the solutions found is desirable. The assessment of this criterion might be based on a certain distance metric.
- The extent of the obtained non-dominated front should be maximized, i.e., for each objective, a wide range of values should be covered by the non-dominated solutions.

Several individual metrics aiming at measuring the achievement of the previous goals by the Pareto set derived from a specific multi-objective algorithm have been proposed in the literature [7,13,47]. Some of them, proposed by Zitzler et al. in [47], are reviewed as follows:

Given a set of pairwise non-dominated decision vectors $X' \subseteq X$, a neighborhood parameter $\sigma > 0$ (to be chosen appropriately), and a distance metric $\|\cdot\|$:

1. Function M_1 gives the average distance to the Pareto-optimal set $\overline{X} \subseteq X$:

$$M_1(X') = 1/|X'| \sum_{a' \in X'} \min\{\|a' - \overline{a}\|; \overline{a} \in \overline{X}\}.$$

2. Function M_2 takes the distribution in combination with the number of non-dominated solutions found into account:

$$M_2(X') = 1/|X' - 1| \sum_{a' \in X'} |\{b' \in X'; \|a' - b'\| > \sigma\}|.$$

3. Function M_3 considers the extent of the front described by X':

$$M_3(X') = \sqrt{\sum_{i=1}^{m} max\{\|a'_i - b'_i\|; a', b' \in X'\}}.$$

Analogously, Zitzler et al. define three metrics M_1^*, M_2^* and M_3^* on the objective space. Let $Y', \overline{Y} \subseteq Y$ be the sets of objective vectors that correspond to X' and $\overline{X}$, respectively, and $\sigma^* > 0$ and $\|\cdot\|^*$ as before:

$$M_1^*(Y') = 1/|Y'| \sum_{p' \in Y'} \min\{\|p' - \overline{p}\|^*; \overline{p} \in \overline{Y}\},$$

$$M_2^*(Y') = 1/|Y' - 1| \sum_{p' \in Y'} |\{q' \in Y'; \|p' - q'\|^* > \sigma^*\}|,$$

$$M_3^*(Y') = \sqrt{\sum_{i=1}^{n} \max\{\|p'_i - q'_i\|^*; p', q' \in Y'\}}.$$

While M_1 and M_1^* are intuitive, M_2 and M_3 (respectively, M_2^* and M_3^*) need further explanation. The distribution metrics give a value within the interval $[0, |X'|]$ ($[0, |Y'|]$). The higher the value of the metric, the better the distribution for an appropriate neighborhood parameter (e.g., $M_2^*(Y') = |Y'|$ means that, for each objective vector, there is no other objective vector within a σ^*-distance to it). Functions M_3 and M_3^* use the maximum extent in each dimension to estimate the range to which the front spreads out. In the case of two objectives, this equals the distance of the two outer solutions.

The previous metrics allows us to determine the absolute, individual quality of a Pareto front. On the other hand, other metrics whose aim is to compare the performance of two different multi-objective algorithms by comparing the Pareto sets generated by each of them, have also been introduced in the literature. One of the most used among these metrics was that proposed by Zitzler et al. in [47], which compares a pair of non-dominated sets by computing the fraction of each set that is covered by the other:

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

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 $1 - C(X'', X')$.

References

[1] B. Barán, M. Schaerer, A multiobjective ant colony system for vehicle routing problem with time windows, in: Proc. Twenty first IASTED International Conference on Applied Informatics, Insbruck, Austria, February 10–13, 2003, pp. 97–102.

[2] T. Bäck, Evolutionary Algorithms in Theory and Practice, Oxford University Press, 1996.

[3] A. Baykasoglu, T. Dereli, I. Sabuncu, A multiple objective ant colony optimization approach to assembly line balancing problems, in: 35th International Conference on Computers and Industrial Engineering (CIE35), Istanbul, Turkey, 2005, pp. 263–268.

[4] B. Bullnheimer, G. Kotsis, C. Strauss, A new rank-based version of the ant system: A computational study, Central European Journal for Operations Research and Economics 7 (1) (1999) 25–38.
[5] P. Cardoso, M. Jesús, A. Márquez, MONACO-multi-objective network optimisation based on an ACO, in: Proc. X Encuentros de Geometría Computacional, Seville, Spain, June 16–17, 2003.
[6] V. Chankong, Y.Y. Haimes, Multiobjective Decision Making Theory and Methodology, North-Holland, 1983.
[7] C.A. Coello, D.A. Van Veldhuizen, G.B. Lamont, Evolutionary Algorithms for Solving Multi-objective Problems, Kluwer, 2002.
[8] O. Cordón, I. Fernández de Viana, F. Herrera, L. Moreno, A new ACO model integrating evolutionary computation concepts: The best–worst ant system, in: M. Dorigo, M. Middendorf, T. Stützle (Eds.), Proc. of ANTS2000—From Ant Colonies to Artificial Ants, Brussels, Belgium, September, 2000, pp. 22–29.
[9] O. Cordón, I. Fernández de Viana, F. Herrera, Analysis of the best–worst ant system and its variants on the TSP, Mathware and Soft Computing 9 (2–3) (2002) 177–192.
[10] O. Cordón, I. Fernández de Viana, F. Herrera, Analysis of the best–worst ant system and its variants on the QAP, in: M. Dorigo, G. Di Caro, M. Sampels (Eds.), Ant Algorithms, Proc. of ANTS2002, Lecture Notes in Computer Science, vol. 2463, Springer-Verlag, Berlin, Germany, 2002, pp. 228–234.
[11] O. Cordón, F. Herrera, T. Stützle, A review on the ant colony optimization metaheuristic: Basis, models and new trends, Mathware and Soft Computing 9 (2–3) (2002) 141–175.
[12] I. Das, J. Dennis, A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems, Structural Optimization 14 (1997) 63–69.
[13] K. Deb, Multi-objective Optimization Using Evolutionary Algorithms, Wiley, 2001.
[14] K. Deb, D.E. Goldberg, An investigation of niche and species formation in genetic function optimization, in: Proc. Third International Conference on Genetic Algorithms (ICGA'89), Hillsdale, USA, 1989, pp. 42–50.
[15] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[16] 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 11 (2) (2003) 115–141.
[17] 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 131 (1–4) (2004) 79–99.
[18] M. Dorigo, G. Di Caro, The ant colony optimization metaheuristic, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, UK, 1999, pp. 11–32.
[19] M. Dorigo, L. Gambardella, Ant colony system: A cooperative learning approach to the travelling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.
[20] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics—Part B 26 (1) (1996) 29–41.
[21] 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, pp. 251–258.
[22] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.
[23] D.B. Fogel, System Identification Trough Simulated Evolution, A Machine Learning Approach, Ginn Press, USA, 1991.
[24] L.M. Gambardella, M. Dorigo, Ant-Q: A reinforcement learning approach to the traveling salesman problem, in: Proc. Twelfth International Conference on Machine Learning (ML-95), Tahoe City, CA, USA, 1995, pp. 252–260.
[25] L. Gambardella, E. Taillard, G. Agazzi, MACS-VRPTW: A multiple ant colony system 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.
[26] X. Gandibleux, M. Sevaux, K. Sörensen, V. T'kindt (Eds.), Metaheuristics for Mutiobjective Optimisation, Lecture Notes in Economics and Mathematical Systems, vol. 535, Springer-Verlag, 2004.
[27] D.E. Goldberg, J. Richardson, Genetic algorithms with sharing for multimodal function optimization, in: Proc. Second International Conference on Genetic Algorithms (ICGA'87), Hillsdale, USA, 1987, pp. 41–49.
[28] M. Gravel, W.L. Price, C. Gagné, Scheduling continuous casting of aluminium using a multiple objective ant colony optimization metaheuristic, European Journal of Operational Research 143 (1) (2002) 218–229.
[29] M. Guntsch, M. Middendorf, A population based approach for ACO, applications of evolutionary computing, in: Proc. of EvoWorkshops2002: EvoCOP, EvoIASP, EvoSTim, 2279, Kinsale, Ireland, 2002, pp. 71–80.
[30] M. Guntsch, M. Middendorf, Solving multi-objective permutation problems with population based ACO, in: Proc. Evolutionary Multi-Criteria Optimization (EMO'03), Faro, Portugal, 2003, pp. 464–478.
[31] A.E. Hans, Multicriteria optimization for highly accurate systems, in: W. Stadler (Ed.), Multicriteria Optimization in Engineering and Sciences, Mathematical Concepts and Methods in Science and Engineering, vol. 19, Plenum Press, 1988, pp. 309–352.
[32] S. Iredi, D. Merkle, M. Middendorf, Bi-criterion optimization with multi colony ant algorithms, in: Proc. First International Conference on Evolutionary Multi-criterion Optimization (EMO'01), Lecture Notes in Computer Science 1993, 2001, pp. 359–372.
[33] A. Jaszkiewicz, Multiple Objective Metaheuristic Algorithms for Combinatorial Optimization, Habilitation thesis, 360, Poznan University of Technology, 2001.
[34] A. Jaszkiewicz, Genetic local search for multi-objective combinatorial optimization, European Journal of Operational Research 137 (1) (2002) 50–71.
[35] J. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, The MIT Press, 1992.

[36] M. López-Ibáñez, L. Paquete, T. Stützle, On the design of ACO for the biobjective quadratic assignment problem, in: M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Montada, T. Stützle (Eds.), Proc. of the Fourth International Workshop on Ant Colony Optimization (ANTS 2004), Lecture Notes in Computer Science, vol. 3172, Springer-Verlag, 2004, pp. 214–225.

[37] 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, Mexico, June, 1999.

[38] C.E. Mariano, E. Morales, MOAQ: An Ant-Q algorithm for multiple objective optimization problems, in: W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Hnavar, M. Jakiela, R.E. Smith (Eds.), Proc. of the Genetic and Evolutionary Computing Conference (GECCO 99), San Francisco, California, USA, July 1999, pp. 894–901.

[39] P.R. McMullen, An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives, Artificial Intelligence in Engineering 15 (3) (2001) 309–317.

[40] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 1996.

[41] D. Pinto, B. Barán, Solving multiobjective multicast routing problem with a new ant colony optimization approach, in: Second IFIP/ACM Latin American Networking Conference (LANC'05), Cali Colombia, 2005, pp. 11–19.

[42] J.D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, in: J.J. Grefenstette (Ed.), Genetic Algorithms and Their Applications: Proc. of the 1st Int. Conf. on Genetic Algorithms, Lawrence Erlbaum, 1985, pp. 93–100.

[43] H.P. Schwefel, Evolution and Optimum SeekingSixth-Generation Computer Technology Series, John Wiley and Sons, 1995.

[44] B.W. Silverman, Density Estimation for Statistics and Data Analysis, Chapman and Hall, London, 1986.

[45] T. Stützle, H.H. Hoos, MAX–MIN ant system, Future Generation Computer Systems 16 (8) (2000) 889–914.

[46] V. T'kindt, N. Monmarché, F. Tercinet, D. Laügt, An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem, European Journal of Operational Research 142 (2) (2002) 250–257.

[47] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, Evolutionary Computation 8 (2) (2000) 173–195.

[48] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength Pareto evolutionary algorithm, in: K. Giannakoglou et al. (Eds.), EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Athens, Greece, September 2001, pp. 12–21.

[49] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 257–271.