Swarm Intell (2012) 6:207–232
DOI 10.1007/s11721-012-0070-7

An experimental analysis of design choices of multi-objective ant colony optimization algorithms

Manuel López-Ibáñez · Thomas Stützle

Received: 1 March 2012 / Accepted: 24 May 2012 / Published online: 4 July 2012

Abstract There have been several proposals on how to apply the ant colony optimization (ACO) metaheuristic to multi-objective combinatorial optimization problems (MOCOPs). This paper proposes a new formulation of these multi-objective ant colony optimization (MOACO) algorithms. This formulation is based on adding specific *algorithm components* for tackling multiple objectives to the basic ACO metaheuristic. Examples of these components are how to represent multiple objectives using pheromone and heuristic information, how to select the best solutions for updating the pheromone information, and how to define and use weights to aggregate the different objectives. This formulation reveals more similarities than previously thought in the design choices made in existing MOACO algorithms. The main contribution of this paper is an experimental analysis of how particular design choices affect the quality and the shape of the Pareto front approximations generated by each MOACO algorithm. This study provides general guidelines to understand how MOACO algorithms work, and how to improve their design.

Keywords Ant colony optimization · Multi-objective optimization · Multi-objective traveling salesman problem · Experimental analysis

1 Introduction

Ant colony optimization (ACO) (Dorigo et al. 1991b, 1996; Dorigo and Stützle 2004) is a swarm intelligence technique that was initially conceived for tackling single-objective combinatorial optimization problems. Given its success on these problems, ACO algorithms

This is an extended version of a paper that received the best paper award of the *Ant Colony Optimization and Swarm Intelligence* track of the GECCO 2010 conference.

M. López-Ibáñez (✉) · T. Stützle
CoDE–IRIDIA, Université libre de Bruxelles (ULB), CP 194/6, Av. F. Roosevelt 50, 1050 Brussels, Belgium
e-mail: manuel.lopez-ibanez@ulb.ac.be

T. Stützle
e-mail: stuetzle@ulb.ac.be

were soon extended to tackle multi-objective combinatorial optimization problems (MOCOPs), resulting in the introduction of multi-objective ant colony optimization (MOACO) algorithms. We focus on MOACO algorithms that do not make any assumption about the preferences of the decision maker. In other words, these MOACO algorithms tackle multi-objective problems in terms of Pareto optimality.

Several tens of papers have been written on MOACO algorithms and their applications. Few of them examine alternative design choices for the algorithm components of the proposed MOACO algorithms (Iredi et al. 2001; López-Ibáñez et al. 2004; Alaya et al. 2007). In addition, two articles that review MOACO algorithms from different angles have been published. García-Martínez et al. (2007) reviewed the existing MOACO algorithms available until 2007 and experimentally compared their performance using the bi-objective traveling salesman problem (bTSP) as a benchmark problem. The goal of this comparison was to identify the best algorithm and it did not attempt to give deeper insights into how the components of MOACO algorithms influence performance. A later review by Angus and Woodward (2009) provides a more fine-grained taxonomy of MOACO algorithms, but lacks an empirical analysis of their behavior.

In this article, we first define the algorithm components that constitute a MOACO algorithm, making a strong distinction between those components specific to multi-objective optimization (MO-specific), and those components related to the underlying ACO algorithm. Then, we identify the design choices proposed in each MOACO algorithm for implementing these MO-specific algorithm components, and we provide a taxonomy that classifies MOACO algorithms with respect to these algorithm components. This taxonomy allows us to identify commonalities between MOACO algorithms that have previously remained undetected. Moreover, we empirically analyze the behavior of these design choices, and we directly relate the performance of different MOACO algorithms and the shape of the Pareto front approximations generated by them to their design choices.

We choose the bTSP as an example application due to its prominent role as a standard benchmark for multi-objective algorithms (García-Martínez et al. 2007; Lust and Teghem 2010; Paquete and Stützle 2009). In fact, most MOACO algorithms are designed to solve bi-objective optimization problems. Hence, we consider the bi-objective variant of the traveling salesman problem (TSP). The bTSP is also a problem where heuristic information is useful and, thus, it allows to study the impact of different strategies for exploiting heuristic information in MOACO algorithms.

The experimental analysis in this paper significantly extends our earlier analysis (López-Ibáñez and Stützle 2010a, 2010b). Firstly, in this analysis, we take into account the parameter settings of the underlying ACO algorithm, in our case $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) (Stützle and Hoos 2000). In fact, the best parameter settings of the underlying ACO algorithm depend on the particular MOACO algorithm. Secondly, we consider the improvement of the ants' solutions by means of local search, which is known to be an essential component of many high-performing ACO algorithms. Thirdly, we also include in the analysis new design alternatives that have not been proposed in any previous MOACO algorithm. As a result we improve the understanding of MOACO algorithms and their design.

This article is structured as follows. In Sect. 2, we first introduce the TSP, the ant colony optimization (ACO) metaheuristic, and $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS). Section 3 introduces some basic notions of multi-objective optimization, discusses the MOACO algorithms we study, and gives our taxonomy of MOACO algorithms. Section 4 provides a systematic experimental analysis of MOACO algorithms and their design choices. We conclude in Sect. 5.

Fig. 1 A high-level view of the ACO metaheuristic for NP-hard problems

procedure ACOMetaheuristic
  ScheduleActivities
    ConstructSolutions
    LocalSearch *//optional*
    UpdatePheromones
  end-ScheduleActivities
end-procedure

2 Ant colony optimization and $\mathcal{MAX}$–$\mathcal{MIN}$ ant system

MOACO algorithms add a number of algorithm components to the ACO metaheuristic to make it amenable to tackle multi-objective problems. To make the paper self-contained, we first give a concise presentation of the ACO metaheuristic and $\mathcal{MM}$AS, the (single-objective) ACO algorithm that we use as the one underlying the MOACO algorithms we study. The TSP was the first problem to which ACO algorithms were applied, and the following presentation will adopt it as an example application.

2.1 The traveling salesman problem

In the TSP, one is given an edge-weighted complete graph $G = (V, E, c)$, where V is the set of $n = |V|$ vertices, E is the set of edges that fully connects the graph, and c is a cost function $c\colon E \to \mathbb{R}$ that assigns to each edge (i, j) a cost $c(i, j)$. Here we assume that the cost function is symmetric, that is, $c(i, j) = c(j, i)$. The goal in the TSP is to find in G a minimum cost Hamiltonian circuit $p = (p_1, \ldots, p_n)$, where the cost of the Hamiltonian circuit is

$$f(p) = c(p_n, p_1) + \sum_{i=1}^{n-1} c(p_i, p_{i+1}). \tag{1}$$

2.2 The ant colony optimization metaheuristic

In ACO algorithms, artificial ants are probabilistic solution construction procedures. The probabilistic decisions taken at each construction step are a function of (artificial) pheromone trails τ_{ij} and heuristic information η_{ij}, which are numerical values associated to each solution component (i, j) of a problem instance to be tackled—for example, an edge in the case of the TSP. In the solution construction, ants probabilistically favor solution components with high pheromone trail values and high heuristic desirability values. The pheromone trail values τ_{ij} represent the desirability of selecting solution component (i, j) learned during the execution of the algorithm; the heuristic information η_{ij} represents the desirability of a solution component as derived from the instance data. Once the solution construction is completed and the solutions are possibly improved by the application of a local search procedure, the pheromone trail values are modified by evaporation and pheromone deposit on some solution components. An algorithmic outline of the ACO metaheuristic is given in Fig. 1. The main features of the three procedures ConstructSolutions, LocalSearch, and UpdatePheromones are the following.

– ConstructSolutions. This procedure implements the solution construction by the ants. In this process each of the N_a ants generates a candidate solution exploiting the pheromone and heuristic information. Two parameters, $\alpha > 0$ and $\beta \geq 0$, determine the relative influence of pheromone vs. heuristic information in the solution construction.

– LocalSearch. The performance of ACO algorithms is often strongly enhanced by improving the ants' solutions using a local search procedure. In fact, local search is probably the most common example of additional techniques that can improve the performance of ACO algorithms. For an overview of other examples, we refer to Dorigo and Stützle (2004).
– UpdatePheromones. The update of the pheromones consists of two processes. The first implements pheromone evaporation, which is a mechanism to lower pheromone values. The amount of pheromone evaporation depends on a parameter $\rho \in [0, 1]$, which is called *evaporation rate*. The second process deposits pheromone on solution components that are contained in good solutions among those previously generated. Which solutions are chosen for the pheromone deposit and how much pheromone they deposit depends on the specific ACO algorithm.

2.3 $\mathcal{MAX}$–$\mathcal{MIN}$ ant system

The first ACO algorithm proposed in the literature was Ant System (Dorigo et al. 1991a, 1996). Soon after it, a number of improved variants were proposed, one of which is $\mathcal{MM}$AS (Stützle and Hoos 2000). $\mathcal{MM}$AS borrows several algorithm components from Ant System such as the solution construction mechanism. As in Ant System, in $\mathcal{MM}$AS solutions are constructed using the random proportional rule. For an ant k positioned in city i, the probability to choose city j as the next one is given by

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{h \in \mathcal{N}^k} [\tau_{ih}]^{\alpha} \cdot [\eta_{ih}]^{\beta}} \quad \text{if } j \in \mathcal{N}^k; \qquad \text{otherwise} \quad p_{ij} = 0; \tag{2}$$

where $\mathcal{N}^k$ is the feasible neighborhood of ant k. In the simplest case, $\mathcal{N}^k$ comprises all those cities that the ant has not yet visited. However, most ACO implementations for the TSP make use of candidate lists. The candidate list of a city i typically comprises a fixed length list of its nearest neighbors. An ant chooses among these candidates as long as feasible neighbors are available from this list. Only if all nearest neighbors have already been visited, a different city may be selected.

The main difference between $\mathcal{MM}$AS and Ant System is in the update of the pheromone trails. In $\mathcal{MM}$AS only one solution deposits pheromone after each iteration. This is typically the iteration-best solution, which is the best solution generated by an ant in the current algorithm iteration, or the best-so-far solution, which is the best solution found since the start of the algorithm. $\mathcal{MM}$AS may also use a restart-best solution, which is the best solution after a re-initialization of the pheromone trails. Another difference between $\mathcal{MM}$AS and Ant System is that the range of pheromone levels in $\mathcal{MM}$AS is limited to an interval $[\tau_{\min}, \tau_{\max}]$, which ensures a minimum degree of search diversification. The pheromone update is implemented by the equation

$$\tau_{ij} \leftarrow \max\left\{\tau_{\min}, \min\left\{\tau_{\max}, (1-\rho) \cdot \tau_{ij} + \Delta\tau_{ij}^{\text{best}}\right\}\right\} \tag{3}$$

where $\Delta\tau_{ij}^{\text{best}}$ is set to

$$\Delta\tau_{ij}^{\text{best}} = \begin{cases} F(s^{\text{best}}) & \text{if edge } (i, j) \text{ is part of the best solution } s^{\text{best}}, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

$F(s^{\text{best}})$ is usually taken to be the reciprocal of the tour cost of the solution that is chosen for the pheromone deposit. The initial pheromone trail level τ_0 in $\mathcal{MM}$AS is set to $\tau_{\max}$.

For a more detailed description of $\mathcal{MM}$AS we refer to the original paper (Stützle and Hoos 2000); for more details on ACO we refer to Dorigo and Stützle (2004).

3 Multi-objective ant colony optimization

3.1 Bi-objective traveling salesman problem

We analyze the performance and the behavior of MOACO algorithms using their example application to the bTSP, which directly extends the single-objective TSP. In the bTSP, the cost function is now a vector-valued function $\mathbf{c}\colon E \to \mathbb{R}^2$ that assigns to each edge (i, j) a cost vector with components $c_1(i, j)$ and $c_2(i, j)$. The first and the second component correspond to the costs for the first and second objective, respectively. We assume here that the bTSP is symmetric, that is, we have $c_q(i, j) = c_q(j, i), i \neq j, q = 1, 2$. The objective in the bTSP is to find a set of Hamiltonian circuits that "minimizes" the tour cost $\mathbf{f} = (f_1, f_2)$; for each objective vector component f_q, the cost of the Hamiltonian circuit is computed according to (1).

If nothing is known about the preferences of the decision maker, the goal becomes to determine a set of Hamiltonian circuits that "minimizes" the objective vector $\mathbf{f}$ in terms of Pareto optimality. We say that a vector $\mathbf{u}$ *dominates* $\mathbf{v}$ ($\mathbf{u} \prec \mathbf{v}$) iff $\mathbf{u} \neq \mathbf{v}$ and $u_i \leq v_i,\ i = 1, \ldots, q$, with q being the number of objectives. $\mathbf{u}$ and $\mathbf{v}$ are *nondominated* iff $\mathbf{u} \not\prec \mathbf{v}$ and $\mathbf{v} \not\prec \mathbf{u}$. We use the same terminology for solutions, that is, we say that a solution s dominates another solution s' iff $\mathbf{f}(s) \prec \mathbf{f}(s')$. A solution is *Pareto optimal* iff there is no feasible solution s' for which we have $\mathbf{f}(s') \prec \mathbf{f}(s)$. Since finding the set of all Pareto-optimal solutions is typically intractable, the goal becomes to determine a set of mutually nondominated solutions that approximates the Pareto-optimal set.

3.2 Multi-objective ant colony optimization

Soon after the first successful applications of ACO algorithms, these have been extended to tackle multi-objective combinatorial optimization problems. Most of these approaches tackle the problems in the Pareto sense. Since the number of MOACO proposals goes into the tens, efforts have been made to review these and to identify their commonalities and differences (Angus and Woodward 2009; García-Martínez et al. 2007).

The first of these efforts was an article by García-Martínez et al. (2007), which classifies the existing algorithms according to the usage of one or several pheromone matrices and the usage of one or several heuristic information matrices. It also provides an experimental study of these algorithms using their example application to the bTSP. The comparison tries to replicate precisely the original algorithms, for example, respecting the original choice of the underlying ACO algorithm. Moreover, the analysis only assesses which algorithm is performing best, without giving insights into the reasons for the performance differences. In particular, no insight is obtained as to whether the observed differences are due to multi-objective (MO-specific) algorithm components of the MOACO algorithms, e.g., using a single versus multiple heuristic matrices, or whether the differences are due to different choices of the underlying ACO algorithm, e.g., whether Ant System or $\mathcal{MM}$AS was used in each MOACO algorithm.

The review by Angus and Woodward (2009) provides a more detailed classification of MOACO algorithms according to algorithm components, such as the pheromone deposit and decay, the type of solution construction, or how candidate solutions are evaluated. Unfortunately, this review does not consider any experimental analysis for identifying the impact

that specific MOACO design decisions have on performance. Therefore, the practical relevance of particular design decisions in MOACO algorithms remains unclear.

In this article, we adopt a different view from these two review articles. In a nutshell, our point of view can be summarized by the equation

$$MOACO = ACO + MO\text{-}specific\ components. \tag{5}$$

In other words, we see MOACO algorithms as being composed of an underlying ACO algorithm whose rules are used to construct solutions or update the pheromones. These algorithms are extended by specific algorithm components that make them amenable to tackling multi-objective optimization problems. In fact, the existing MOACO algorithms can be cast in such terms.

3.3 MOACO algorithms

In the following, we examine several MOACO algorithms with the goal of identifying their MO-specific components. We focus on MOACO algorithms that have been designed with Pareto optimization in mind. Hence, we exclude from our analysis a number of algorithms that were designed for lexicographic optimization (Mariano and Morales 1999; Gambardella et al. 1999; Gravel et al. 2002), or algorithms that diverge from the basic structure of the ACO metaheuristic, such as population-based ACO (Guntsch and Middendorf 2003; Angus 2007). We also exclude from our review the adaptation of MOAQ to the bTSP (García-Martínez et al. 2007), since we have already shown that its results are extremely poor in comparison with the other algorithms (López-Ibáñez and Stützle 2010b), and it does not contribute to our discussion.

3.3.1 BicriterionAnt

BicriterionAnt (Iredi et al. 2001) is a MOACO algorithm that uses different pheromone and heuristic matrices for each objective, which are combined by a weighted product aggregation. In other words, the computation of the values of τ_{ij} and η_{ij} that are used in (2) is done as

$$\tau_{ij} = \left(\tau^1_{ij}\right)^{(1-\lambda_k)} \cdot \left(\tau^2_{ij}\right)^{\lambda_k} \quad \text{and} \quad \eta_{ij} = \left(\eta^1_{ij}\right)^{(1-\lambda_k)} \cdot \left(\eta^2_{ij}\right)^{\lambda_k}, \tag{6}$$

where τ^1_{ij} and τ^2_{ij} are the pheromone trails associated to solution component (i, j) for each of the two pheromone matrices, and η^1_{ij} and η^2_{ij} are the heuristic information specific for each objective. A weight $\lambda_k \in \Lambda$, with $0 \leq \lambda_k \leq 1$, is associated to each ant k, and, hence, there are as many weights as ants ($|\Lambda| = N_a$).

In the original proposal, Iredi et al. (2001) suggest to update the pheromone matrices by using the set of nondominated solutions found in the current iteration P^{ib} by an amount equal to $\Delta\tau = 1/|P^{ib}|$. However, this approach only works for heterogeneous pheromone matrices, that is, when each matrix is mapped to different solution components. In the bTSP (and other problems), this is not the case, and such an update will result in multiple identical pheromone matrices. Hence, we follow García-Martínez et al. (2007), and we use $\Delta\tau^q = 1/f_q(s_a)$ to update each matrix, where f_q is the objective function value of objective $q = 1, 2$.

In the same paper, Iredi et al. (2001) discuss the use of multiple colonies. We will examine this possibility in Sect. 3.5.

3.3.2 Multiple ant colony system

Multiple Ant Colony System (MACS) (Barán and Schaerer 2003) differs from BicriterionAnt in the use of a single pheromone matrix instead of several. As BicriterionAnt, MACS uses multiple heuristic matrices, which are aggregated by weighted product using a different weight λ_k for each ant k, as shown above.

In MACS, as in BicriterionAnt, pheromone information is updated with nondominated solutions. However, we use $\Delta\tau = 1$ for the pheromone deposit, since all solutions update the same pheromone matrix.

3.3.3 COMPETants

The original proposal of COMPETants (Doerner et al. 2003) formulates it as a multi-colony approach with one colony for each objective. Each "colony" has one pheromone and heuristic matrix and constructs solutions independently, except for a number of ants (called "spies"), which aggregate the two pheromone matrices by weighted sum (with equal weight given to each matrix) using either the first or the second heuristic matrix (thus creating two solutions). Finally, a number of ants from each "colony" are used to update the pheromone matrix of their own "colony".

It is possible to formulate COMPETants as a single-colony algorithm that uses multiple pheromone and heuristic matrices, which are aggregated by weighted sum. First, we define weights as $\lambda \in \{0, 0.5, 1\}$. Ants using $\lambda = 0$ only consider the pheromone and heuristic matrices corresponding to the first objective, whereas ants using $\lambda = 1$ only consider the ones corresponding to the second objective. These two weights correspond to the "colonies" in the original formulation of COMPETants, whereas $\lambda = 0.5$ corresponds to the "spies". The aggregation of the pheromone matrices in COMPETants is equivalent to

$$\tau_{ij} = (1 - \lambda)\tau_{ij}^1 + \lambda\tau_{ij}^2. \tag{7}$$

To match exactly the original formulation of COMPETants, we consider only $\lambda \in \{0, 1\}$ when aggregating the heuristic matrices, that is, half the ants use only one heuristic matrix η^1 and the other half use the other heuristic matrix η^2. This effectively means that heuristic matrices are not aggregated.

The pheromone update method of COMPETants differs from the one of BicriterionAnt and MACS. In our single-colony formulation, we update each pheromone matrix with the best solutions with respect to the corresponding objective. We call this method *best-of-objective* update.

For the sake of simplicity, we disregard the fact that in the original COMPETants algorithm the number of ants for each weight was chosen adaptively based on the algorithm progress; here each weight is used by one third of the total number of ants. We also follow $\mathcal{MM}$AS so that each pheromone matrix is updated with only one solution, the best one with respect to the corresponding objective. We use $\Delta\tau = 1$ as pheromone deposit, because the solutions used to update each matrix are different, and they may have quite different objective values.

3.3.4 Pareto ant colony optimization

Pareto Ant Colony Optimization (P-ACO) (Doerner et al. 2004) is characterized by the use of multiple pheromone matrices, one for each objective, which are aggregated by means

of a weighted sum, instead of a weighted product as in BricriterionAnt. Each ant k uses a different weight λ_k for aggregating the pheromone matrices. Each pheromone matrix is updated with the best and the second-best solution for the corresponding objective. This is equivalent to the pheromone update in COMPETants, which we call *best-of-objective*, but using two solutions per pheromone matrix. We use a constant pheromone deposit of $\Delta\tau = 1$ as in COMPETants.

In the original proposal, Doerner et al. (2004) consider just a single heuristic matrix, since it was difficult to define appropriate heuristic information for each objective in their benchmark problem. In later publications, however, Schilde et al. (2009) use multiple heuristic matrices, one for each objective. The heuristic matrices are aggregated in the same way as the pheromone matrices, that is, by means of a weighted sum. The design choice between using only one or multiple heuristic matrices is crucial for the performance of the algorithm. Here, we will assume that the default configuration of P-ACO uses multiple heuristic matrices, since, as we will show later, this is the most advantageous configuration.

3.3.5 m-ACO variant 1 (m-ACO$_1$)

Alaya et al. (2007) proposed four alternatives for the design of a MOACO algorithm. The formulation presented for the first variant, m-ACO$_1$, has one "colony" per objective, and an "extra colony" that builds solutions by aggregating the pheromone matrices of the other two colonies in a stochastic manner. In particular, at each construction step, ants from the extra colony select randomly one objective and use the pheromone information associated to that objective. Each colony uses the heuristic information of the corresponding objective, whereas the extra colony aggregates these multiple heuristic values into a single one.

As in the case of COMPETants, m-ACO$_1$ can be formulated as a single-colony approach that resembles more closely the other algorithms reviewed above. This formulation of m-ACO$_1$ uses multiple pheromone matrices and multiple heuristic matrices, one for each objective, which are aggregated using three weights $\lambda \in \{0, 0.5, 1\}$. The pheromone matrices are "aggregated" by choosing at each construction step, with a probability $(1 - \lambda)$, the first matrix and with probability λ the second matrix. We call this approach *weighted random aggregation*. The multiple heuristic matrices are aggregated by means of weighted sum. With $\lambda = 0$ or $\lambda = 1$, only one of the pheromone and heuristic matrices is used, which mimics the original formulation of m-ACO$_1$.

In the original formulation of m-ACO$_1$, the pheromone information of each colony is updated by solutions of the same colony. Only one solution is used to update each colony's pheromone matrix. The update of the extra colony is slightly different: each pheromone matrix is updated with the best solution for each objective generated by the extra colony. That is, two solutions are selected for the update, but only one solution updates each pheromone matrix. Different colonies in the original formulation correspond to a different λ in our formulation. Let us consider first the case of the extra colony, which corresponds to $\lambda = 0.5$. We keep two lists of best solutions generated using this weight, each list ordered with respect to a different objective. Then, each pheromone matrix is updated with the best solution of the list corresponding to the same objective. In the case of $\lambda = 0$, we keep only the list for the first objective, and, hence, only the first pheromone matrix is updated. For $\lambda = 1$, we keep only the list for the second objective, and the best of this list is used to update the second pheromone matrix. This update method, which we call *best-of-objective-per-weight*, replicates the pheromone update of m-ACO$_1$. As in COMPETants and P-ACO, we use a pheromone deposit of $\Delta\tau = 1$.

3.3.6 m-ACO variant 2 (m-ACO$_2$)

The second variant by Alaya et al. (2007) only differs from m-ACO$_1$ in the pheromone aggregation, which in m-ACO$_2$ is done by summing the pheromone matrices of each objective. In fact, this corresponds to a *weighted sum aggregation* using a weight vector $(1, 1)$. When using $\mathcal{MM}$AS as the underlying ACO algorithm, or any other scale invariant ACO algorithm (Birattari et al. 2007), this is equivalent to using $\lambda = 0.5$. Hence, m-ACO$_2$ is very similar to P-ACO (Sect. 3.3.4) but using only three weights ($\lambda \in \{0, 0.5, 1\}$), and several ants per weight.

3.3.7 m-ACO variant 3 (m-ACO$_3$)

The third variant proposed by Alaya et al. (2007) uses a single pheromone matrix. This pheromone matrix is updated by using nondominated solutions. In particular,

$$\Delta\tau_{ij} = \begin{cases} 1 & \text{if edge } (i, j) \text{ appears in a solution in } P, \\ 0 & \text{otherwise;} \end{cases}$$

where P is a set of nondominated solutions. In general, it could be the set of nondominated solutions found in the current iteration (iteration-best set), or since the start of the run (best-so-far set). Alaya et al. (2007) emphasize that every pheromone value is updated only once, independent of the number of solutions that contain it. This is different from other algorithms that use a "nondominated update" such as MACS and BicriterionAnt. However, the general principle is the same, and only the quantity of pheromone deposit differs. The heuristic information is also a single matrix, which is built by aggregating the heuristic information corresponding to each objective into a single matrix: $\eta_{ij} = \eta^1_{ij} + \eta^2_{ij}$. Finally, since there is a single pheromone matrix and a single heuristic matrix, the random proportional rule of m-ACO$_3$ is the same as for the single-objective ACO algorithms (2).

3.3.8 m-ACO variant 4 (m-ACO$_4$)

In the fourth variant proposed by Alaya et al. (2007), there is one pheromone matrix per objective. These matrices are aggregated by means of weighted random aggregation as in m-ACO$_1$. There is only one single heuristic matrix, which is built as in the m-ACO$_3$ variant. Finally, each pheromone matrix is updated with the best solution for each objective, which exactly matches the *best-of-objective* update method used by COMPETants and P-ACO.

3.4 Taxonomy of single-colony MOACO algorithms

Table 1 categorizes the algorithms reviewed above according to several algorithm components: the use of a single or multiple pheromone matrices, a single or multiple heuristic matrices, the aggregation method, the number of weights, and the pheromone update method. When there is only a single pheromone matrix and a single heuristic matrix, as in m-ACO$_3$, aggregation is not necessary.

We consider that all algorithms in Table 1 are single-colony MOACO algorithms. In the next section, we discuss the use of multiple colonies in MOACO.

Table 1 Taxonomy of single-colony MOACO algorithms. $[\tau]$ refers to the use of a single or multiple pheromone matrices, $[\eta]$ refers to the use of a single or multiple heuristic matrices, Num. weights denotes the number of values used for λ, and the pheromone update method (Ph. update) can be either nondominated solutions (ND), best-of-objective (BO) or best-of-objective-per-weight (BOW). Also, q is the number of objectives, and N_a is the number of ants

Algorithm	$[\tau]$	$[\eta]$	Aggregation	Num. weights	Ph. update
BicriterionAnt	q	q	Weighted product	N_a	ND
MACS	1	q	Weighted product	N_a	ND
COMPETants	q	q	Weighted sum	$q+1$ ($\Lambda = \{0, 0.5, 1\}$)	BO
P-ACO	q	q	Weighted sum	N_a	BO
m-ACO$_1$	q	q	Random (τ) Weighted sum (η)	$q+1$ ($\Lambda = \{0, 0.5, 1\}$)	BOW
m-ACO$_2$	q	q	Weighted sum	$q+1$ ($\Lambda = \{0, 0.5, 1\}$)	BOW
m-ACO$_3$	1	1	–	–	ND
m-ACO$_4$	q	1	Random (τ)	1 ($\Lambda = \{0.5\}$)	BO

3.5 Multi-colony BicriterionAnt

The idea of using multiple "colonies" of ants can be found in many MOACO algorithms. However, the definition of "colony" is far from consistent across these proposals. For example, in the original description of COMPETants (Doerner et al. 2003), a group of ants that uses the same pheromone and heuristic information to construct solutions is said to be a multi-colony approach because the ants may use different weights in aggregating diverse information corresponding to different objectives. The same can be said of the m-ACO$_1$ and m-ACO$_2$ algorithms. By contrast, Iredi et al. (2001) describe a "multi-colony" algorithm as a multi-start version of a single-colony algorithm, where each of the colonies may have different settings. According to this definition, each colony uses its own pheromone information and has an associated number of ants that use exclusively that information to construct solutions. Cooperation among the colonies is then enabled through the exchange of solutions. This definition of multi-colony MOACO is more consistent with the idea of independent colonies and we adopt it here. It has the advantage of being easily applicable to all the algorithms reviewed above, as we have shown elsewhere (López-Ibáñez and Stützle 2012a). For the sake of simplicity, we only consider in this paper the multi-colony variant of BicriterionAnt (MC-BicriterionAnt).

In MC-BicriterionAnt, different colonies may be forced to specialize in different regions of the Pareto front by various means (Iredi et al. 2001). First, colonies may be assigned different weights for aggregation, either completely disjoint sets or sets that share common weights. We call this component *multi-colony weights*. More formally, a set of equally distributed weights in the interval [0, 1] is generated. Then, in the case of *disjoint* multi-colony weights, this set is partitioned into equal disjoint sub-intervals per colony, that is, $\lambda_{c,i} = ((c-1) \cdot N_{\text{weights}} + (i-1))/(N_{\text{weights}} \cdot N_{\text{col}})$, $i = 1, \ldots, N_{\text{weights}}$, $c = 1, \ldots, N_{\text{col}}$. In the case of *overlapping* multi-colony weights, the sub-intervals overlap by a given percentage, which is fixed to 50 % here.

Iredi et al. (2001) propose that colonies cooperate to identify the best solutions by storing the solutions constructed by all colonies in a single archive. Nondominated solutions from this archive are assigned back to the various colonies to update their pheromone matrices. We call *multi-colony update* the procedure that assigns solutions to each colony for updating

its pheromone information. In the simplest case, *update by origin*, each solution is assigned to the same colony that constructed it. An alternative, called *update by region*, is to assign a region of the Pareto front to each colony, and solutions located in a particular region update the pheromone information of the corresponding colony. In the bi-objective case, a trivial approximation of *update by region* is to sort the nondominated front according to one objective, divide it into as many equal-sized parts as colonies, and use each part to update each colony.

In the experimental part, we will analyze the actual impact of these alternative choices on the behavior of MC-BicriterionAnt.

4 Empirical analysis of MOACO algorithms

4.1 Experimental setup and methodology

The goal of our experimental analysis is not to identify the best MOACO algorithm, but rather to understand the relationship between algorithmic design choices and the results produced by each algorithm. As said before, we use the bTSP as a benchmark problem in our study. In particular, we consider six benchmark instances. The first four, `kroAB100`, `kroAB150`, `kroAB200`, and `euclidAB500` have been taken from Luis Paquete's webpage at http://eden.dei.uc.pt/~paquete/tsp. In these instances, the two cost matrices are generated independently of each other; each cost matrix gives the Euclidean distance between points that are distributed uniformly at random in a square of a specific side length. The number in the instance identifier corresponds to the number of nodes. In addition, we generated in the same way two larger instances, `euclidAB700`, and `euclidAB1000`. The three smaller instances are used for MOACO algorithms without local search, while the three larger ones for MOACO algorithms that incorporate local search.

For experiments on the three largest instances, we hybridize the MOACO algorithms with local search. Each solution constructed by an ant is the starting point of a first-improvement local search based on the 2-exchange neighborhood (2-opt). This is a single-objective local search that works on a weighted sum aggregation of the two distance matrices. This weighted sum aggregation in the local search uses the same weight as the ant used to construct the solution. In the few MOACO algorithms where ants do not use weights, we assign a different weight to each ant that is used only by the local search. The local search uses the standard speed-up techniques for the TSP, such as "don't-look" bits and fixed-radius nearest-neighbor search. The latter is supported by a list per node of the 20 nearest neighbors. This list is sorted in non-increasing order of the edge costs in the weighted sum aggregation of the two distance matrices, and, hence, we keep different lists for different weights.

As usually done in the single-objective case, when tackling the three largest instances, the ants' solution construction is made more efficient by using a candidate list of size 20, which is obtained by sorting edges according to dominance ranking (Lust and Jaszkiewicz 2010).

We perform 10 repetitions of each experiment. Each repetition uses a different seed for the random number generator. Each run is stopped after $300 \cdot (n/100)^2$ CPU-seconds in the case without local search, and $4 \cdot (n/100)^2$ CPU-seconds in the case with local search. The MOACO algorithms are implemented in `C` based on an existing MOACO framework (López-Ibáñez and Stützle 2012a), and compiled with `gcc`, version 4.4. Each experiment is run on a single core of an AMD Opteron 6128 CPU (2 GHz, 512 KB L2 cache size) running under Rocks Cluster GNU/Linux version 4.2.1/CentOS 4.

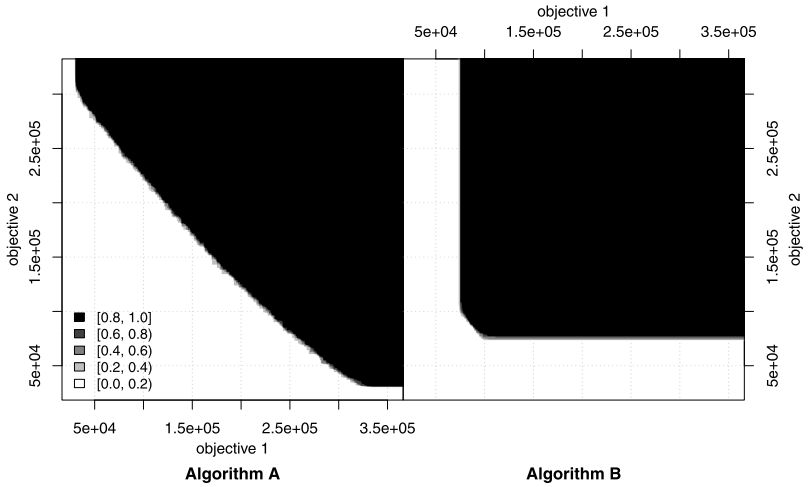

Fig. 2 EAFs of two algorithms A and B. The level of gray indicates the value of the EAF at each point in the objective space, that is, the estimated probability of attaining each point in a single run of the corresponding algorithm

As said above, all MOACO algorithms use $\mathcal{MM}$AS as the underlying ACO algorithm. By choosing one fixed ACO algorithm, we remove one factor that influences the performance of the MOACO algorithms and, in this way, we are able to attribute possible performance differences to specific design choices made in the multi-objective components. We follow the design of $\mathcal{MM}$AS for single-objective optimization (Sect. 2) as much as possible, except for a few details. In particular, the number of solutions that update the pheromone matrices is specified by the original MOACO algorithms. Moreover, we use a constant amount of pheromone deposit (4), except when multiple pheromone matrices are updated with non-dominated solutions, in which case we use the reciprocal of the tour cost with respect to the objective associated to each pheromone matrix, thus, the pheromone deposit is different on each matrix.

Since in many cases the outcomes of different algorithms are incomparable in the Pareto sense, we consider the hypervolume (Zitzler and Thiele 1998; Fonseca et al. 2006) as a unary quality measure that allows us to completely rank Pareto fronts. Neither the hypervolume, nor any other quality measure, give precise information about the differences between the Pareto fronts produced by two algorithms (Zitzler et al. 2003). We therefore visualize these differences by means of a graphical technique (López-Ibáñez et al. 2006, 2010) based on the empirical attainment function (EAF) (Grunert da Fonseca et al. 2001).

The attainment function gives for each point in the objective space the probability that a single run of an algorithm *attains* it, that is, an objective vector generated by the algorithm dominates the point or is equal to it. The attainment function is generally unknown, but it can be estimated by collecting data from several independent runs and by computing the corresponding EAF. Figure 2 is a side-by-side plot of the EAFs of two different algorithms applied to the same bTSP instance. The level of gray at each point gives the (estimated) probability of attaining that point (objective vector) in a single run of the algorithm. A black point indicates that the algorithm attained that point vector in more than 80 % of the runs.

In order to examine differences in the results obtained by two algorithms, we compute the differences between their EAFs at each point in the objective space, as shown in Fig. 3.

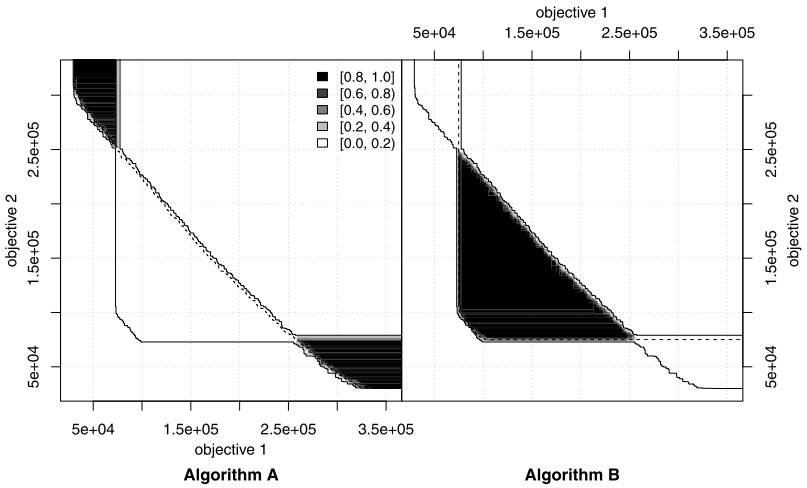

Fig. 3 EAF differences between the two algorithms whose EAFs are shown in Fig. 2. For each point in the objective space, the level of gray gives the difference in the estimated probability of attaining that point in a single run of algorithm A versus a single run of algorithm B. The *left side* shows the EAF differences in favor of algorithm A, while the *right side* shows the differences in favor of algorithm B

The sign of the differences indicates which algorithm has a higher probability of attaining each point in a single run. We plot separately the differences of the same sign, such that at each side the differences are in favor of the algorithm labeled at the bottom. The shade of gray now encodes the magnitude of the differences. For example, a black point indicates that one algorithm attained that objective vector in at least 80 % more runs than the other algorithm. If the same point is white on both sides, it indicates that either both algorithms have roughly the same probability of attaining that point, or neither algorithm has a probability larger than 20 %. In the example shown in Fig. 3, algorithm A attains solutions with extreme objective values that are never attained by algorithm B (as shown on the left plot), whereas algorithm B is always better in a large region of the center of the Pareto front approximation (as shown on the right plot).

An *attainment surface* is the minimum set of objective vectors that are attained with a given probability. For example, the 50 % (median) attainment surface is the nondominated set of objective vectors that are attained by at least 50 % of the runs. In Fig. 3, the median attainment surface of each algorithm is represented with a dashed line. We also plot with continuous lines the grand-best attainment surface and the grand-worst attainment surface, that is, respectively, the nondominated points attained by at least one run of any of the two algorithms, and the nondominated points always attained by both algorithms in all runs. All plots in this paper are generated using the `eaf` R package available at http://cran.r-project.org/package=eaf.

4.2 Analysis of the underlying ACO parameters

The default $\mathcal{MM}$AS settings for the single-objective TSP are not necessarily well suited when applying MOACO algorithms to the bTSP (López-Ibáñez and Stützle 2012a). Therefore, as a first step, we study the impact of the parameter settings of $\mathcal{MM}$AS on each MOACO algorithm by a full factorial analysis. We choose the best settings identified in this

Table 2 Parameters and parameter values that were considered for the analysis of $\mathcal{MM}$AS's impact on the performance of MOACO algorithms. N'_a is a surrogate parameter that is used to determine the number of ants, N_a

Parameter	Values
ρ	0.02, 0.10, 0.50, 0.90, 0.98
β	1, 3, 5
N_a	$6 \cdot N'_a \cdot \lfloor n/100 \rfloor$ without LS
	$6 \cdot N'_a$ with LS
N'_a	4, 8, 16

analysis to carry out later a fairer comparison of the MOACO algorithms, so that differences in performance can be attributed to the multi-objective algorithmic components rather than to poor parameter settings of the underlying ACO algorithm. The parameters and their values considered in this analysis are shown in Table 2. Instead of specifying directly the number of ants (N_a), we specify a surrogate parameter (N'_a), which is multiplied by six in order to ensure that N_a is divisible by three, as required by COMPETants, m-ACO$_1$ and m-ACO$_2$. When no local search is used, N_a depends on the instance size as usual in $\mathcal{MM}$AS for single-objective problems. Our experimental design results in $5 \cdot 3 \cdot 3 = 45$ candidate parameter settings.

We ran the default setting of all eight MOACO algorithms with respect to their MO-specific components with the above settings for the underlying $\mathcal{MM}$AS. Each of the 45 candidate parameter settings was run 10 times with different random seeds on each of the six instances. As a way to analyze the interactions among the parameters, we considered a multi-way ANOVA. We found that there are important interactions, especially between ρ, N'_a and the particular MOACO algorithm. However, the results do not satisfy the normality and homoscedasticity requirements for carrying out ANOVA. Thus, this approach does not allow us to generalize or summarize the experimental results, apart from some general observations as, for example, a setting of $\beta = 5$ is significantly better than other choices when not using local search. As an alternative, we use the non-parametric Friedman-test by blocks (Conover 1999), where each block is one repetition on one instance. Within each block, the different configurations of each MOACO algorithm are ranked according to their hypervolume. For conciseness, we report the full results as supplementary material (López-Ibáñez and Stützle 2012b). Here, we simply choose for further analysis the configuration of each algorithm that obtains the lowest sum of ranks over all blocks. The configurations chosen are summarized in Table 3. Although most algorithms perform better when using a high value of N'_a and β, there are major differences between several algorithms, which confirms our previous assumption that there are no ACO settings that are good for all MOACO algorithms. In the next section, we use the best configuration of each algorithm to carry out a more in-depth analysis of their MO-specific components.

4.3 Analysis of the MO-specific components

We now analyze the effect of various design choices in MOACO algorithms. As a first step, we examine the outcomes of the MOACO algorithms when using the ACO settings described in Table 3. We plot the 50 %-attainment surfaces of these algorithms for the `kroAB200` instance in Fig. 4. We use two plots for clarity, and plot the 50 %-attainment surface of BricriterionAnt in both plots as a reference for comparison. The first observation is that BicriterionAnt and MACS obtain the best Pareto front approximations in terms of shape and quality. COMPETants, m-ACO$_1$ and m-ACO$_2$ show obvious gaps in their Pareto front approximation. Finally, m-ACO$_3$ and m-ACO$_4$ produce very narrow approximations,

Table 3 Best settings of $\mathcal{MM}$AS underlying each of the MOACO algorithms

Algorithm	Without LS			With LS		
	N'_a	β	ρ	N'_a	β	ρ
BicriterionAnt	16	5	0.98	16	5	0.02
MACS	4	5	0.9	16	5	0.02
COMPETants	4	1	0.5	16	3	0.02
P-ACO	16	5	0.02	16	5	0.9
m-ACO$_1$	4	3	0.02	16	5	0.1
m-ACO$_2$	16	3	0.9	8	5	0.98
m-ACO$_3$	16	3	0.5	16	5	0.02
m-ACO$_4$	16	5	0.5	16	1	0.98

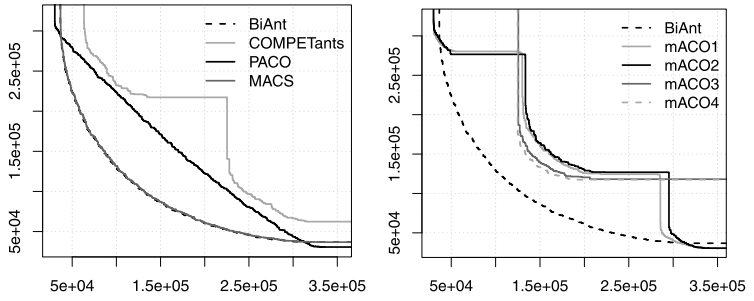

Fig. 4 50 % attainment surfaces of the various MOACO algorithms studied in the experiments. Objectives 1 and 2 are given in the x- and y-axis, respectively, in these plots and in the plots shown in Figs. 5 to 9 and 11

which are closer to the Pareto front in the center, but that fail to approximate the extremes of the Pareto front.

In order to explain these remarkably different behaviors, we examine in this section one algorithmic component at a time. We focus our analysis on only four algorithms: BicriterionAnt, P-ACO, m-ACO$_2$ and MACS. However, as we will show, the analysis generalizes to all MOACO algorithms, since our analysis explains the behavior of all algorithms in terms of their particular algorithmic design. We illustrate our discussion with plots of the EAF differences, limited to the 200 nodes (without local search) and 1000 nodes (with local search) instances. The full set of plots for all instances is available as supplementary material (López-Ibáñez and Stützle 2012b).

Weighted sum vs. weighted product In our earlier experiments (López-Ibáñez and Stützle 2010b), we observed that the aggregation of multiple pheromone and/or heuristic matrices by means of weighted product gives better results than using weighted sum aggregation. Sucha result is surprising at first. Given a computation time limit, MOACO algorithms using weighted product perform substantially less iterations than the same algorithms using weighted sum and, thus, they evaluate fewer candidate solutions. For example, BicriterionAnt is able to perform four times more iterations when using weighted sum instead of weighted product aggregation. The new experiments carried out here confirm these results when not using local search and on a wider range of MOACO algorithms and instances. Figure 5 shows the EAF differences between these two alternatives for four MOACO algo-

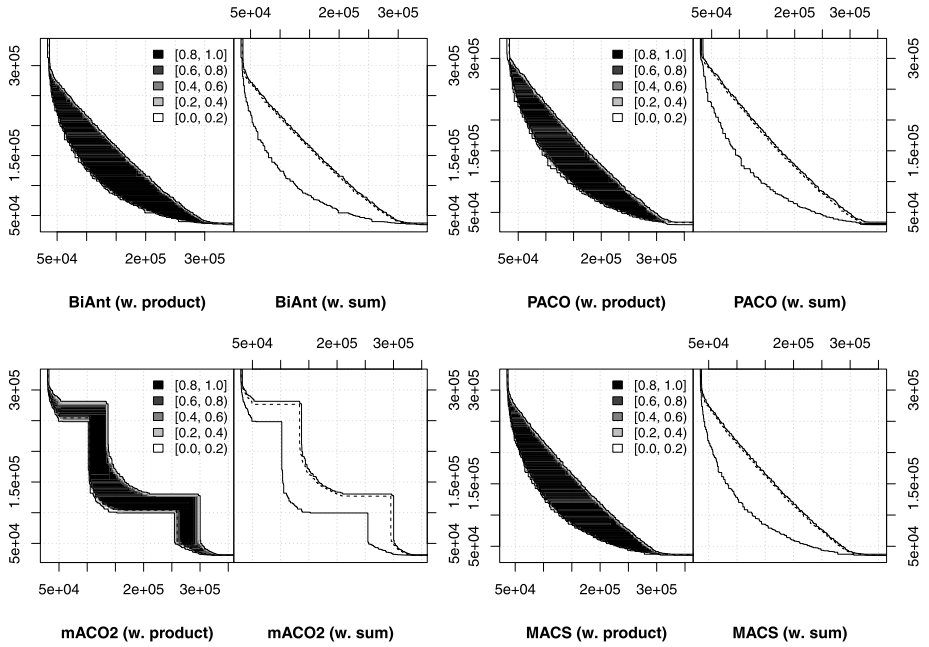

Fig. 5 Comparison of MOACO algorithms using either weighted product (w. product) or weighted sum (w. sum) aggregation by plots of EAF differences. The results are given for various MOACO algorithms without local search on instance `kroAB200`

rithms in the `kroAB200` instance. In all cases, a weighted product aggregation improves the quality of the approximation in the center of the Pareto front.

The explanation of this behavior is that, in the random Euclidean bi-objective TSP, the heuristic matrices are not correlated, and hence, a weighted sum tends to average out the value of all components of the matrices. A weighted product aggregation better differentiates between components that have a high heuristic value in both matrices and the rest.

However, when the same MOACO algorithms incorporate local search, the benefit of using a weighted product aggregation mostly disappears, because the influence of heuristic information is smaller, and its positive bias is not compensated by the computation cost of the weighted product aggregation.

Single vs. multiple heuristic matrices A few MOACO algorithms, such as m-ACO_3 and m-ACO_4 and some variants of P-ACO, aggregate the heuristic information corresponding to each objective in a manner that is independent of any dynamic weight setting. This is equivalent to aggregating multiple heuristic matrices into a single one at the start of the algorithm. The main benefit is speed, since a single heuristic matrix is computed once at the start of the algorithm. This approach also avoids defining a set of weights and a weight setting strategy. However, a single heuristic matrix strongly biases the solution construction towards the center of the Pareto front, where both objectives are balanced. If the heuristic information plays a strong role in generating high-quality solutions, then this bias may be stronger than other components that favor diversity, such as the use of multiple pheromone matrices.

This is the effect shown in Fig. 6. The plots corresponding to BicriterionAnt, which uses multiple pheromone matrices, and MACS, which uses a single pheromone matrix, are

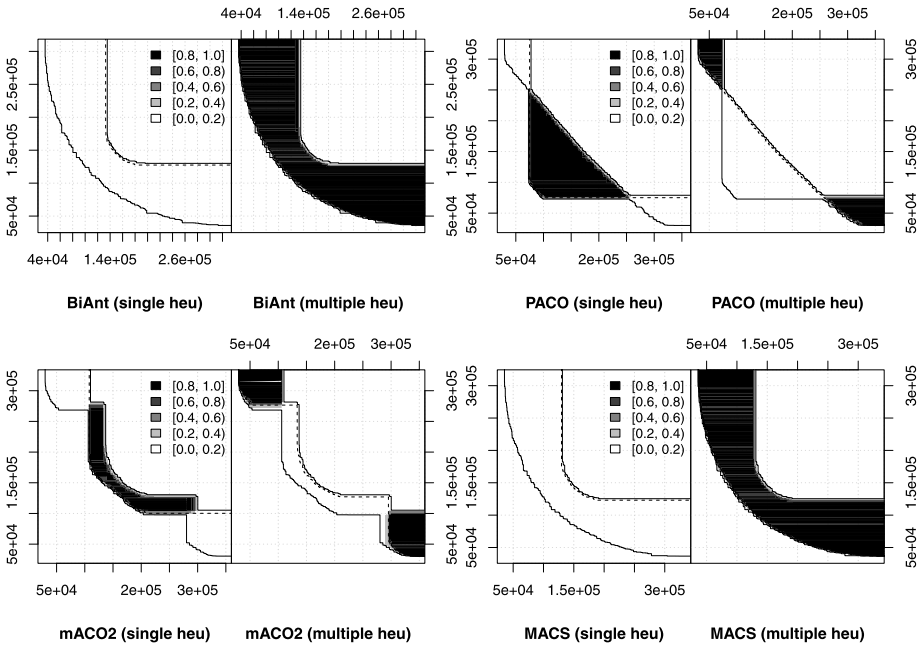

Fig. 6 Comparison of MOACO algorithms using a single (single heu) or multiple (multiple heu) heuristic matrices by plots of EAF differences. The results are given for various MOACO algorithms without local search on instance `kroAB200`

extremely similar. This indicates that obtaining a wider Pareto front is mostly independent of the number of pheromone matrices, but it requires using multiple heuristic matrices. This conclusion is further confirmed by the plot corresponding to P-ACO (top-right plot in Fig. 6). P-ACO also uses multiple pheromone matrices but, in contrast to BicriterionAnt, it uses best-of-objective pheromone update, which means that each pheromone matrix is updated using the best solution for each objective. Hence, the pheromone update of P-ACO is strongly biased towards the extremes of the Pareto front. However, the plot shows that this bias is not enough to produce a wide Pareto front approximation when using a single heuristic matrix. On the contrary, since the single heuristic matrix gives equal importance to both objectives, the search is biased towards high-quality solutions with approximately the same value of the two objectives (i.e., the center of the Pareto front approximation), and it fails to approximate the extremes of the Pareto front.

When using local search, as shown in Fig. 7, EAF differences are not easily visible because of the wide range of the Pareto front compared to the small distance between the grand-best and grand-worst attainment surfaces. Therefore, we do not examine the EAF differences for the algorithms using local search in the rest of the paper, but later we compare these algorithms in terms of the hypervolume measure. Nonetheless, the attainment surfaces shown in Fig. 7 illustrate that the shape of the Pareto front is determined by the number of weights used by the local search. In particular, m-ACO$_2$ uses only three weights, whereas the other three algorithms use N_a weights. On the other hand, the shape is not affected by the number of heuristic matrices used by each algorithm. This indicates that the bias of the local search is much stronger than the bias of the heuristic information. When comparing the algorithms according to the hypervolume, algorithms using a single heuristic matrix produce

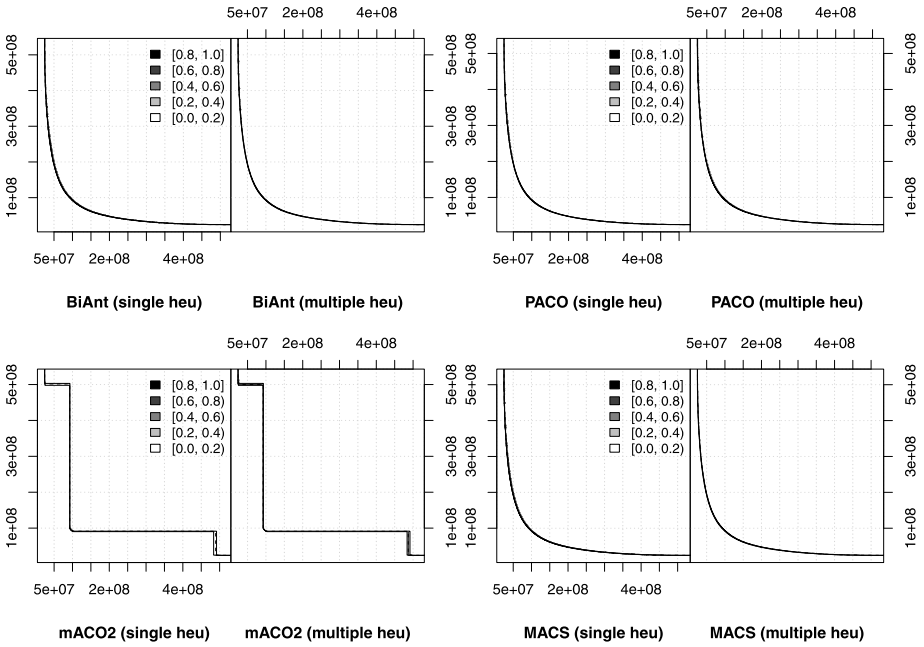

Fig. 7 Comparison of MOACO algorithms using a single (single heu) or multiple (multiple heu) heuristic matrices by plots of EAF differences. The results are given for various MOACO algorithms with local search on instance `euclidAB1000`

better results since they are able to perform more iterations. For example, MACS performs around two times more iterations when using a single heuristic matrix rather than two, and, hence, it obtains better solutions.

Pheromone update: nondominated vs. best-of-objective Figure 8 compares the different choices available for pheromone update in the MOACO literature, that is, nondominated update and best-of-objective update (best-of-objective-per-weight in the case of m-ACO$_2$). Interestingly, the nondominated update approach obtains better results in all cases, independently of the number of pheromone matrices (MACS uses only one, the others use two), the type of aggregation (P-ACO and m-ACO$_2$ use weighted sum, the others use weighted product) and the number of weights (m-ACO$_2$ uses only three, the others use as many as N_a). The best-of-objective update method is only better on the very extremes of the Pareto front.

When local search is used, the clear advantage of nondominated update disappears.

One weight per iteration (1wpi) vs. all weights per iteration (awpi) All algorithms proposed in the MOACO literature use the whole set of weights at every iteration (*awpi* approach). In a previous work (López-Ibáñez and Stützle 2010a), we proposed to use only one weight at each iteration and change the weight at every iteration (*1wpi* approach). The benefits of the *1wpi* approach are evident in Fig. 9. The *1wpi* approach does not significantly change the shape of the Pareto front approximation, however, it leads to an overall improvement with respect to the Pareto front approximation obtained with *awpi*. The main reason is that by using the same weight at each iteration, the aggregation of multiple matrices can be computed once per iteration, instead of recomputing it for each ant. This leads to a large increase of the number of iterations (roughly by a factor of ten) that can be performed within

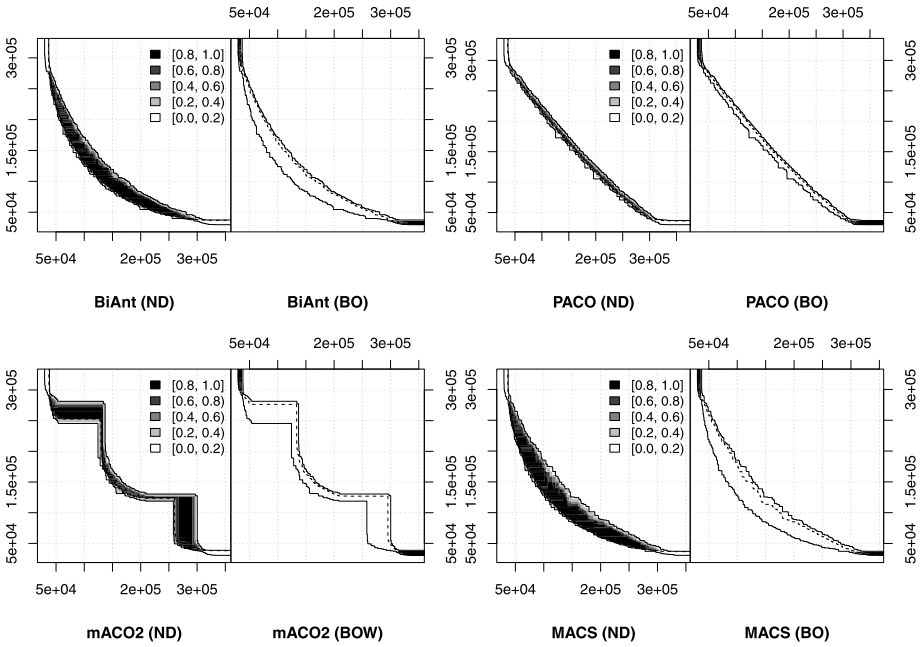

Fig. 8 Comparison of MOACO algorithms using nondominated (ND) vs. best-of-objective (BO) pheromone update by plots of EAF differences. The results are given for various MOACO algorithms without local search on instance `kroAB200`

Table 4 Experimental setup of Multi-Colony BicriterionAnt

Parameter	Values
N_{col}	2, 5, 10
Multi-colony weights	*Disjoint*, *overlapping*
Multi-colony update	*Origin*, *region*

the same computation time limit allowing the algorithm to reach better solutions. These results are consistent for all the instances tested, with and without local search.

4.4 Multi-colony BicriterionAnt

In this section, we analyze the multi-colony (MC) BicriterionAnt. In our previous work (López-Ibáñez and Stützle 2010b), we already noticed that using more than one colony was always beneficial. However, there are several additional design choices that only make sense when using multiple colonies (Table 4). We examine here the effect of these parameters.

As a first step, we run all possible combinations of the parameters shown in Table 4 with all possible combinations of the underlying ACO parameters described in Table 2. As above in the single-colony algorithms, the goal of this first step is to understand the impact of the ACO settings, and determine a good set of settings to perform further analysis.

For analyzing the influence of all ACO settings, we tried to perform ANOVA on each instance, yet, the ANOVA requirements are not fully satisfied. The only overall observations we are able to reach is that without local search a strong heuristic information is needed.

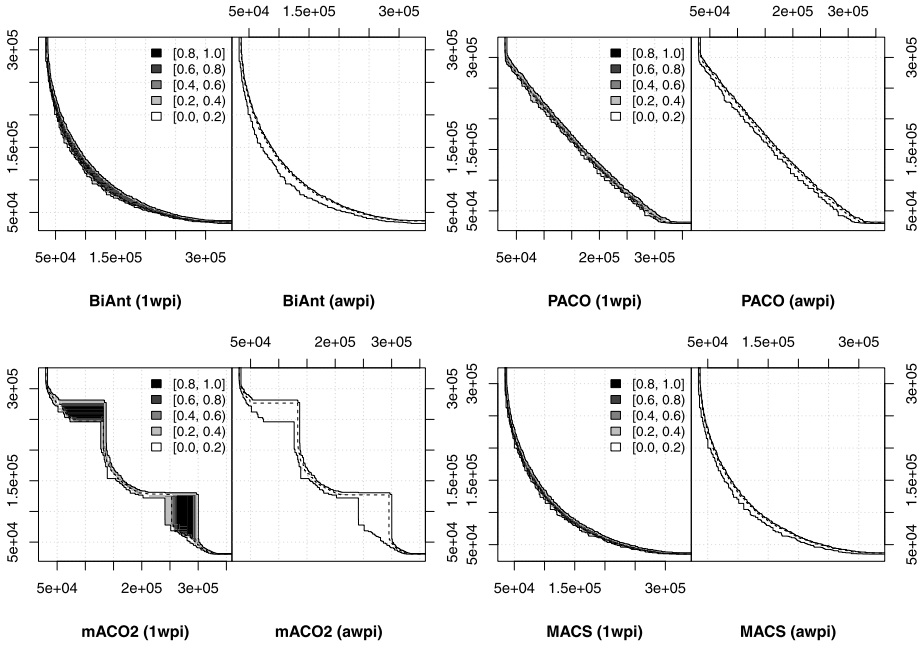

Fig. 9 Comparison of MOACO algorithms using one weight per iteration (1wpi) vs. all weights per iteration (awpi) by plots of EAF differences. The results are given for various MOACO algorithms without local search on instance `kroAB200`

Table 5 Best underlying ACO settings per algorithm

Algorithm	Without LS			With LS		
	N'_a	β	ρ	N'_a	β	ρ
MC BicriterionAnt	16	5	0.98	8	3	0.98

Results obtained with $\beta = 5$ are significantly better than with other settings. By contrast, when using local search this is not the case.

Instead of ANOVA, we again use ranking by blocks to identify the best overall ACO settings, which are shown in Table 5. The overall conclusion is that a strong evaporation seems to be necessary. The explanation for this setting is that high evaporation leads to a faster convergence of the algorithm. A faster convergence is required, since the use of multiple colonies reduces the number of iterations the algorithm is able to perform by a factor close to the number of colonies.

In a second step, we examine the algorithmic components specific to multi-colony BicriterionAnt (Table 4). Once the ACO settings are fixed as above, the requirements of ANOVA are mostly satisfied. For the sake of conciseness, we only give an overview of the most interesting observations. First, the use of ten colonies was always the best setting in our experiments, with a strong difference over the single-colony approach (see Fig. 11). This is somewhat surprising, since a large number of colonies in BicriterionAnt incurs a considerable overhead caused by the extra pheromone matrices. Given a computation time limit, the number of iterations that the algorithm is able to perform decreases by a factor of the number of colonies. These results are, nonetheless, consistent with all our previous experiments

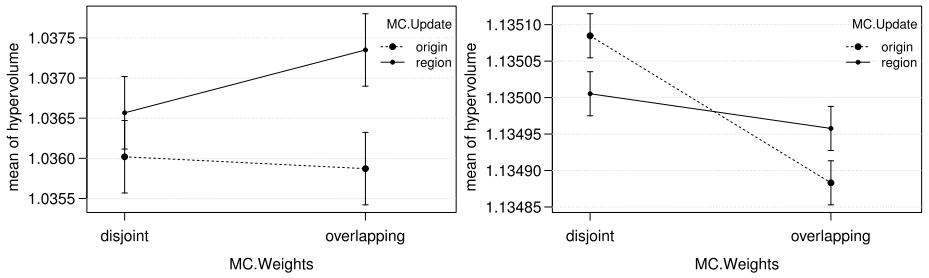

Fig. 10 Interaction between MC weights and MC update parameters (*left*: `kroAB200` without LS; *right*: `euclidAB1000`, with LS)

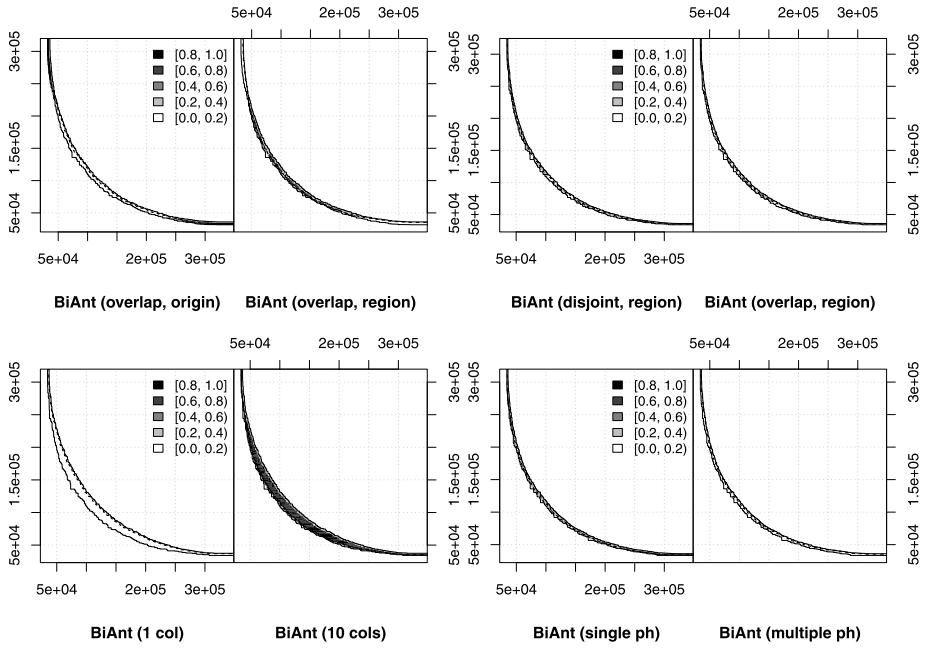

Fig. 11 Comparison of various settings for multi-colony BicriterionAnt by plots of EAF differences. The results are given for BicriterionAnt without local search on instance `kroAB200`

(López-Ibáñez and Stützle 2010b, 2012a). These results suggest that the specialization introduced by the multiple colonies is key to obtain a diverse Pareto front approximation.

We also observed a strong interaction between the parameter multi-colony weights (*disjoint* or *overlapping*) and the parameter multi-colony update (*origin* or *region*). Figure 10 shows the interaction plots for one instance tackled without local search (left) and one instance tackled with local search (right). We also compare the EAFs of various settings in Fig. 11. From these plots, we extract the following observations.

In the case without local search, we observed that it is better to combine *update by region* with *overlapping* rather than *disjoint* weight intervals. It is less clear which choice of weight intervals works better when combined with *update by origin*. A possible explanation is that when colonies share weights, they are more likely to find similar solutions, and,

hence, exchanging them (*update by region*) helps the colonies to specialize. Despite what the interaction plot may suggest, the EAF plots (Fig. 11, top right) show that, when using 10 colonies, the difference between *disjoint* and *overlapping* intervals is not very strong. In fact, when using 10 colonies, the main difference is that exchanging solutions (*update by region*) leads to a better performance in the center of the Pareto front, as shown in Fig. 11 (top left).

In the case with local search, *disjoint* weight intervals are strongly better, presumably because the local search uses a more varied set of weights, generating a more diverse set of solutions. Also, the weights of the local search itself impose a stronger division between the colonies, which makes exchanging solutions between colonies (*update by region*) not beneficial anymore. However, it is still beneficial if weights overlap, since one colony may generate very good solutions in the region of other colonies. This would explain why *update by origin* becomes worse than *update by region* when combined with *overlapping* weight intervals, while the opposite is true when *update by origin* and *update by region* are combined with *disjoint* weight intervals (Fig. 10, right).

The main conclusions are as follows. First, the use of multiple colonies in the sense of Iredi et al. (2001) (i.e., with independent pheromone matrices) is very beneficial despite its high cost in computation time. We predict it will be even more beneficial in other problems where the overhead of the MOACO algorithm is lower with respect to the evaluation cost of solutions. Second, if the solution construction is strong enough to force a specialization of each colony to a different region, for example, when using a strong weighted local search, then it is better that colonies use *disjoint* intervals, i.e., do not share weights, and do not exchange solutions (i.e., *update by origin*). Otherwise, it is better that the colonies share weights (i.e., *overlapping* intervals) and exchange solutions among them (i.e., *update by region*).

4.5 Comparison of the best MOACO algorithms

As a final step in our analysis, we examine the quality obtained by the best MOACO algorithms. We choose MACS, P-ACO and BicriterionAnt (with $N_{\text{col}} = 1$ and $N_{\text{col}} = 10$), since they give the best results overall with and without local search. For each algorithm, we use the best ACO settings that were identified above. For BicriterionAnt ($N_{\text{col}} = 10$), we use *overlapping* multi-colony weights and *update by region* when not using local search, and *disjoint* multi-colony weights and *update by origin* when using local search, following the analysis done in the previous section. All algorithms use the one-weight-per-iteration strategy, since it was always better than the alternative.

Figure 12 gives the hypervolume of the selected MOACO algorithms without local search on three bTSP instances. The results confirm our previous analysis, that is, MACS and BicriterionAnt ($N_{\text{col}} = 1$) produce better results than P-ACO, and the results of BicriterionAnt are further enhanced by using multiple colonies.

Figure 13 is the corresponding plot when the algorithms use local search. In this case, P-ACO is much better than MACS and BicriterionAnt ($N_{\text{col}} = 1$), because when using local search, weighted sum aggregation and best-of-objective update are better than the alternatives. However, the use of multiple colonies in BicriterionAnt is still the best approach.

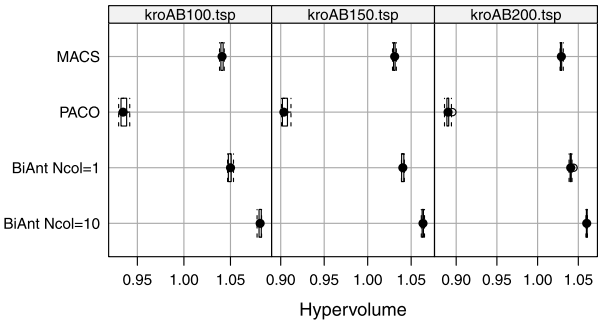

Fig. 12 Boxplot of the hypervolume obtained in 10 runs by four MOACO algorithms without local search

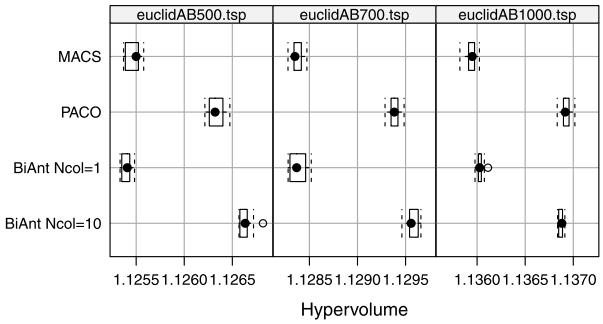

Fig. 13 Boxplot of the hypervolume obtained in 10 runs by four MOACO algorithms with local search

5 Conclusions

This paper has presented a thorough empirical analysis of the various design choices made in MOACO algorithms. Our analysis shows that some design choices have a profound impact on the quality and shape of the Pareto front approximation obtained.

- The use of very few weights for aggregating multiple pheromone or heuristic matrices (as done, for example, in COMPETants, m-ACO$_1$ and m-ACO$_2$) generates Pareto front approximations that are very good in the direction of those weights, but extremely poor in the areas not covered by them. This effect is strongest when applying a weighted local search to the solutions constructed by the ants. The conclusion is that the more effective the generation of solutions, the more disconnected the resulting Pareto front approximation. Using a higher number of weights alleviates this problem.
- The weighted sum aggregation of pheromone or heuristic matrices has difficulty approximating the convex part of the Pareto front. By contrast, a weighted product aggregation produces a much better approximation of the center of the Pareto front. However, if an effective local search is able to generate high-quality solutions in the center of the Pareto front, then the high computation cost of a weighted product aggregation cancels its benefits.
- In problems where heuristic information is essential to reach high-quality solutions, a single heuristic matrix, as used by m-ACO$_3$ and m-ACO$_4$, biases the algorithm towards the center of the Pareto front, leading to a very narrow Pareto front and a poor approximation of the extremes of the Pareto front.
- The strategy that uses one weight per iteration (1wpi) is computationally more efficient than using all weights at each iteration (awpi), without any noticeable downsides in terms

of quality. Hence, 1wpi is clearly the best choice for the bTSP, and we believe that this strategy may be useful for other problems where the aggregation of the pheromone and heuristic matrices can be precomputed once per iteration.
– The multi-colony approach used by BicriterionAnt is always better than the single-colony BicriterionAnt. This is even true on large instances, where the overhead of additional colonies is larger. Therefore, we recommend to consider the use of multiple colonies when applying MOACO to a new problem. Moreover, most MOACO algorithms can be extended to use multiple colonies, as we have shown in a previous work (López-Ibáñez and Stützle 2012a).

The above conclusions answer many questions on what causes the striking performance difference between various MOACO algorithms. These conclusions also explain why such performance differences are more evident on the bTSP. The reasons include the particular shape of the Pareto front in the bTSP, the need for a very strong heuristic information, and the fact that previous analyses did not take local search into account. As we show here, choosing the best algorithm, or more concretely, making the best design choices when designing a MOACO algorithm for a specific problem requires that these factors are taken into consideration.

An interesting extension of this work would be to repeat this analysis on problems with different characteristics than the bTSP. For example, ACO algorithms for the bQAP do not use heuristic information. An analysis of MOACO algorithms on problems for which an effective local search is not available would also be of interest.

Most of the MOACO algorithms examined here were proposed specifically for bi-objective optimization problems, and how to best extend them to tackle more than two objectives is unclear. Undoubtedly, the conclusions reached here may differ in the case of more than two objectives. However, we hope that such research will follow the component-wise experimental analysis presented in this paper.

The taxonomy and analysis of MOACO algorithms described here was a crucial step in the development of a configurable MOACO framework, which we automatically configured in order to find the best design of a MOACO algorithm for the bTSP (López-Ibáñez and Stützle 2012a). The analysis presented here complements those results by providing insights into the working principles of MOACO algorithms, which will be instrumental towards defining new design choices for inclusion into the MOACO framework. An open challenge is how to best combine these two complementary approaches: a systematic analysis of the alternative design choices for algorithmic components, and an automatic configuration of these components to find the best overall design. The ultimate goal would be to automatically find the best design of a metaheuristic such as MOACO for a specific problem, and, at the same time, provide insights on why such design is the best.

Acknowledgements The research leading to the results presented in this paper has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement nº246939. This work was also supported by the Meta-X project, funded by the Scientific Research Directorate of the French Community of Belgium. Manuel López–Ibáñez and Thomas Stützle acknowledge support of the F.R.S.-FNRS of which they are a post-doctoral researcher and a research associate, respectively.

References

Alaya, I., Solnon, C., & Ghédira, K. (2007). Ant colony optimization for multi-objective optimization problems. In *19th IEEE international conference on tools with artificial intelligence (ICTAI 2007)* (Vol. 1, pp. 450–457). Los Alamitos: IEEE Computer Society Press.

Angus, D. (2007). Population-based ant colony optimisation for multi-objective function optimisation. In M. Randall, H. A. Abbass, & J. Wiles (Eds.), *Lecture notes in computer science: Vol. 4828. Progress in artificial life (ACAL)* (pp. 232–244). Heidelberg: Springer.

Angus, D., & Woodward, C. (2009). Multiple objective ant colony optimisation. *Swarm Intelligence*, *3*(1), 69–85.

Barán, B., & Schaerer, M. (2003). A multiobjective ant colony system for vehicle routing problem with time windows. In *Proceedings of the twenty-first IASTED international conference on applied informatics*, Innsbruck, Austria (pp. 97–102).

Birattari, M., Pellegrini, P., & Dorigo, M. (2007). On the invariance of ant colony optimization. *IEEE Transactions on Evolutionary Computation*, *11*(6), 732–742.

Conover, W. J. (1999). *Practical nonparametric statistics* (3rd ed.). New York: Wiley.

Doerner, K. F., Hartl, R. F., & Reimann, M. (2003). Are COMPETants more competent for problem solving? The case of a multiple objective transportation problem. *Central European Journal for Operations Research and Economics*, *11*(2), 115–141.

Doerner, K. F., Gutjahr, W. J., Hartl, R. F., Strauss, C., & Stummer, C. (2004). Pareto ant colony optimization: a metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research*, *131*, 79–99.

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

Dorigo, M., Maniezzo, V., & Colorni, A. (1991a). The ant system: an autocatalytic optimizing process. Tech. Rep. 91-016, Revised, Dipartimento di Elettronica, Politecnico di Milano, Italy.

Dorigo, M., Maniezzo, V., & Colorni, A. (1991b). Positive feedback as a search strategy. Tech. Rep. 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy.

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

Fonseca, C. M., Paquete, L., & López-Ibáñez, M. (2006). An improved dimension-sweep algorithm for the hypervolume indicator. In *Proceedings of the 2006 congress on evolutionary computation (CEC 2006)* (pp. 1157–1163). Piscataway: IEEE Press.

Gambardella, L. M., Taillard, É. D., & Agazzi, G. (1999). 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* (pp. 63–76). London: McGraw-Hill.

García-Martínez, C., Cordón, O., & Herrera, F. (2007). A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research*, *180*(1), 116–148.

Gravel, M., Price, W. L., & Gagné, C. (2002). Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic. *European Journal of Operational Research*, *143*(1), 218–229.

Grunert da Fonseca, V., Fonseca, C. M., & Hall, A. O. (2001). Inferential performance assessment of stochastic optimisers and the attainment function. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, & D. Corne (Eds.), *Lecture notes in computer science: Vol. 1993. Evolutionary multi-criterion optimization (EMO 2001)* (pp. 213–225). Heidelberg: Springer.

Guntsch, M., & Middendorf, M. (2003). Solving multi-objective permutation problems with population based ACO. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, & L. Thiele (Eds.), *Lecture notes in computer science: Vol. 2632. Evolutionary multi-criterion optimization (EMO 2003)* (pp. 464–478). Heidelberg: Springer.

Iredi, S., Merkle, D., & Middendorf, M. (2001). Bi-criterion optimization with multi colony ant algorithms. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, & D. Corne (Eds.), *Lecture notes in computer science: Vol. 1993. Evolutionary multi-criterion optimization (EMO 2001)* (pp. 359–372). Heidelberg: Springer.

López-Ibáñez, M., & Stützle, T. (2010a). An analysis of algorithmic components for multiobjective ant colony optimization: a case study on the biobjective TSP. In P. Collet, N. Monmarché, P. Legrand, M. Schoenauer, & E. Lutton (Eds.), *Lecture notes in computer science: Vol. 5975. Artificial evolution: 9th international conference, evolution artificielle, EA 2009* (pp. 134–145). Heidelberg: Springer.

López-Ibáñez, M., & Stützle, T. (2010b). The impact of design choices of multi-objective ant colony optimization algorithms on performance: an experimental study on the biobjective TSP. In M. Pelikan & J. Branke (Eds.), *Proceedings of the genetic and evolutionary computation conference, GECCO 2010* (pp. 71–78). New York: ACM Press.

López-Ibáñez, M., & Stützle, T. (2012a). The automatic design of multi-objective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*. doi:10.1109/TEVC.2011.2182651.

López-Ibáñez, M., & Stützle, T. (2012b). An experimental analysis of design choices of multi-objective ant colony optimization algorithms: supplementary material. http://iridia.ulb.ac.be/supp/IridiaSupp2012-006/.

López-Ibáñez, M., Paquete, L., & Stützle, T. (2004). On the design of ACO for the biobjective quadratic assignment problem. In M. Dorigo et al. (Eds.), *Lecture notes in computer science: Vol. 3172. Ant colony optimization and swarm intelligence, 4th international workshop, ANTS 2004* (pp. 214–225). Heidelberg: Springer.

López-Ibáñez, M., Paquete, L., & Stützle, T. (2006). Hybrid population-based algorithms for the bi-objective quadratic assignment problem. *Journal of Mathematical Modelling and Algorithms*, *5*(1), 111–137.

López-Ibáñez, M., Paquete, L., & Stützle, T. (2010). Exploratory analysis of stochastic local search algorithms in biobjective optimization. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, & M. Preuss (Eds.), *Experimental methods for the analysis of optimization algorithms* (pp. 209–222). Berlin: Springer.

Lust, T., & Jaszkiewicz, A. (2010). Speed-up techniques for solving large-scale biobjective TSP. *Computers & Operations Research*, *37*(3), 521–533.

Lust, T., & Teghem, J. (2010). Two-phase Pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, *16*(3), 475–510.

Mariano, C. E., & Morales, E. (1999). MOAQ: an ant-Q algorithm for multiple objective optimization problems. In W. Banzhaf, J. M. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. J. Jakiela, & R. E. Smith (Eds.), *Proceedings of the genetic and evolutionary computation conference, GECCO 1999* (pp. 894–901). San Francisco: Morgan Kaufmann.

Paquete, L., & Stützle, T. (2009). Design and analysis of stochastic local search for the multiobjective traveling salesman problem. *Computers & Operations Research*, *36*(9), 2619–2631.

Schilde, M., Doerner, K. F., Hartl, R. F., & Kiechle, G. (2009). Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, *3*(3), 179–201.

Stützle, T., & Hoos, H. H. (2000). $\mathcal{MAX}$–$\mathcal{MIN}$ ant system. *Future Generations Computer Systems*, *16*(8), 889–914.

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—a comparative case study. In A. E. Eiben, T. Bäck, M. Schoenauer, & H. P. Schwefel (Eds.), *Lecture notes in computer science: Vol. 1498. Parallel problem solving from nature, PPSN V* (pp. 292–301). Heidelberg: Springer.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & Grunert da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, *7*(2), 117–132.