The Impact of Design Choices of Multiobjective Ant Colony Optimization Algorithms on Performance: An Experimental Study on the Biobjective TSP

Manuel López-Ibáñez
IRIDIA, CoDE, Université Libre de Bruxelles
Brussels, Belgium
manuel.lopez-ibanez@ulb.ac.be

Thomas Stützle
IRIDIA, CoDE, Université Libre de Bruxelles
Brussels, Belgium
stuetzle@ulb.ac.be

ABSTRACT
Over the last few years, there have been a number of proposals of ant colony optimization (ACO) algorithms for tackling multiobjective combinatorial optimization problems. These proposals adapt ACO concepts in various ways, for example, some use multiple pheromone matrices and multiple heuristic matrices and others use multiple ant colonies.

In this article, we carefully examine several of the most prominent of these proposals. In particular, we identify commonalities among the approaches by recasting the original formulation of the algorithms in different terms. For example, several proposals described in terms of multiple colonies can be cast equivalently using a single ant colony, where ants use different weights for aggregating the pheromone and/or the heuristic information. We study algorithmic choices for the various proposals and we identify previously undetected trade-offs in their performance.

Categories and Subject Descriptors
I.2.8 [Computing Methodologies]: Artificial Intelligence—Problem Solving, Control Methods, and Search

General Terms
Algorithms

Keywords
Multiobjective Optimization, Ant Colony Optimization, Travelling Salesman Problem

1. INTRODUCTION
Ant colony optimization (ACO) [8] is one of the most prominent swarm intelligence techniques. Its initial application area was single objective combinatorial problems. However, the available ACO algorithms were soon extended to tackle multiobjective combinatorial optimization problems (MCOPs), where the quality of candidate solutions is evaluated according to several, conflicting objectives. A few tens of papers have described ACO applications to MCOPs. In addition, two recent papers have reviewed many of the available multiobjective ACO (MOACO) algorithms [2, 9]. Although some initial proposals of MOACO algorithms focused on problems that define a relative order between the objectives, most MOACO algorithms focused on problems that define a relative order between the objectives.

In contrast, we make an effort to identify the algorithmic components of each MOACO algorithm that are specific for tackling multiobjective problems. We implement those components on top of a common underlying ACO algorithm. We test both MAX-MIN ant system (MMAS) [19] and ant colony system (ACS) [7] as the underlying ACO algorithms. Finally, we design experiments that systematically test these algorithm components. For this experimental analysis we choose the example application to the biobjective traveling salesperson problem (bTSP). The experimental comparisons are done in two stages. First, we highlight for some algorithms the influence of some specific MOACO design. At a second stage, we compare the performance of several of these algorithms and identify the most promising proposals.

Thanks to this approach, we identify in this paper commonalities between various existing algorithms that have gone largely unnoticed until now. For example, we show how several MOACO algorithms that were originally formulated as multi-colony can easily be cast in an equivalent form as single colony algorithms, using some form of weighted aggregation of multiple pheromone and/or heuristic factors. In fact, we identify several MOACO algorithms as particular instantiations of a common parametrized MOACO algorithm.

Finally, our systematic experimental analysis allows us to draw a direct connection between the multiobjective algorithmic components utilized by existing MOACO algorithms and the quality and distribution of solutions in the Pareto fronts obtained. This knowledge will help to design better MOACO algorithms in the future.
2. PRELIMINARIES

In this paper, we apply MOACO algorithms to the bTSP, a problem that is a direct extension of the single objective TSP to a biobjective one. In a bTSP instance is given a complete graph $G=(V,A)$ with $n=|V|$ nodes $\{v_1,...,v_n\}$, and a set of arcs $A$. Each arc has assigned a vector of costs with two components $c_1(v_i,v_j)$ and $c_2(v_i,v_j)$, $i \neq j$, which corresponds to the cost of the first and the second objective, respectively. We consider here the symmetric bTSP, where it additionally holds that $c_2(v_i,v_j) = c_2(v_j,v_i)$, $i \neq j$, $q = 1, 2$.

The goal in the bTSP is to find the set of Hamiltonian tours $p = (p_1,...,p_n)$ that “minimizes” the total tour cost, which is given by

$$f_q(p) = c_q(v_{p(i)},v_{p(i+1)}) + \sum_{i=1}^{n-1} c_q(v_{p(i)},v_{p(i+1)}) \quad q = 1, 2.$$  

Here we assume that the bTSP is solved without knowing the preferences of the decision maker. The goal then is to identify a set of feasible solutions that “minimizes” the objective vector $\vec{f}$ in the sense of Pareto optimality. A vector $\vec{u}$ dominates $\vec{v}$ $(\vec{u} \preceq \vec{v})$ if $\vec{u}_i \leq \vec{v}_i$, $i = 1, ..., d$, where $d$ is the number of objectives. $\vec{u}$ and $\vec{v}$ are non-dominated if $\vec{u} \neq \vec{v}$ and $\vec{v} \neq \vec{u}$. We apply the same notation to solutions, meaning that a solution $s$ dominates another solution $s'$ if $\vec{f}(s) \preceq \vec{f}(s')$. A solution is Pareto optimal iff there is no feasible solution $s'$ for which we have $\vec{f}(s) \preceq \vec{f}(s')$. Since finding the set of all Pareto optimal solutions for an MCOP is typically intractable, the goal usually becomes to determine a set of mutually non-dominated solutions that approximates the Pareto-optimal set.

3. MULTI-OBJECTIVE ACO

Since the end of 1990s, different proposals have appeared in the literature extending single-objective ACO algorithms for tackling MCOPs. The majority of these MOACO algorithms tackle problems in the Pareto sense, where no a priori assumptions on the preferences of the decision maker are made. Recently, two reviews of MOACO algorithms have been published [2, 9]. The comprehensive review by Garcia-Martinez et al. [9] describes the available algorithms, classifies them according to the usage of one or several pheromone matrices and the usage of one or several heuristic information matrices, and also provides a comparison of some of the original algorithms using the bTSP as a case study. This methodology indicates which of the original algorithms performs better for the bTSP. However, one cannot conclude whether the observed performance differences are due to any particular, multiobject-specific algorithmic component of the MOACO algorithm, e.g., the use of single versus multiple pheromone matrices, or due to the use of ACS versus ant system as the underlying ACO algorithm. The more recent review by Angus and Woodward [2] provides a detailed classification of the available MOACO algorithms based on additional features, such as the pheromone update and decay, the type of solution construction, or how candidate solutions are evaluated. In contrast to the earlier review [9], this later review does not provide any experimental analysis, and hence, the practical relevance of the design decisions in MOACO algorithms remains unclear.

Among the available papers on MOACO algorithms, few do some experimental analysis of their proposals. Iredi et al. are among the first to propose ACO extensions for tackling multiobjective problems and they investigate some design options based on the usage of multiple colonies [11]. Alaya et al. [1] study MOACO algorithms for the biobjective knapsack problem and compare the performance of four variants that differ in the number of colonies and the aggregation of pheromone matrices. An earlier article compared different design options for MOACO algorithms on the biobjective quadratic assignment problem (bQAP) [13], and this work was recently extended to the bTSP [12].

There are a number of design questions when extending ACO algorithms to MCOPs. First, since the quality of solutions cannot be represented by a single scalar value, the meaning of the pheromone information associated to a solution component is unclear in the multiobjective context. In fact, some MOACO algorithms use several pheromone matrices, each one of them associated to a different objective [1, 6, 11]. Another important design decision is the choice of ants that deposit pheromone. These may be some or all nondominated solutions [11], or a number of the best solutions with respect to the objective associated to the pheromone matrix that is updated [1, 6]. During the solution construction, multiple pheromone or heuristic matrices are often combined by means of weights [6, 11], although existing MOACO algorithms differ in the number of weights and the manner in which the are used.

Interestingly, 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. In fact, some authors give the name “colony” to any group of ants that share some characteristics but ants from different colonies may still use the same pheromone and heuristic information to construct solutions. For example, COMPETants [5] (see also section 4.5) is described as multi-colony when, in fact, it can be seen as weighted aggregation of the pheromone matrices. Iredi et al. [11] characterize “colonies” as multi-start versions of single-colony algorithms, each of them possibly with different settings, which may cooperate through the exchange of solutions. Following this definition, each colony has independent pheromone information and an associated number of ants that use exclusively that information to construct solutions. We believe that this latter definition is more consistent with the idea of cooperation between independent colonies.

4. ANALYSIS OF MOACO ALGORITHMS

In this section, we review existing MOACO algorithms with the aim to differentiate between algorithmic design choices for multiobjective optimization, such as the use of several pheromone matrices, and other design choices, such as those that concern the underlying ACO algorithm.

Table 1 summarizes the MOACO algorithms reviewed in this paper. They are classified according to the following algorithmic components: the number of pheromone matrices, the number of heuristic matrices, the method used to aggregate multiple pheromone (heuristic) matrices, the number of weights used for aggregation, and the method used to select the solutions that update the pheromone information. In the following sections, we will explain how these algorithmic components suffice to characterize the MOACO algorithms reviewed in this paper.

For each of the MOACO algorithms in Table 1, we examine experimentally specific design choices. To this aim,
we have implemented all MOACO algorithms in C using ACOTSP [18] as the underlying ACO package, and compiled with gcc, version 3.4. Experiments are carried out on AMD Opteron 2216 dual-core 2.4 GHz processors with 2 MB L2-Cache under Rocks Cluster GNU/Linux. Given that the implementation is sequential, all codes run on a single core.

We performed experiments with two underlying ACO algorithms, namely MMAS [19] and ACS [7]. In some cases, we noticed significant differences in performance depending on the choice of the underlying ACO algorithm; we give some examples below. Basic parameters follow the default values defined for MMAS and ACS with two exceptions. We typically use $\Delta \tau = 1$ for the amount of pheromone deposited by an ant, and the evaporation rate is set to $\rho = 0.05$. All algorithms use $m = 24$ ants, which is a value close to the 20 ants used by García-Martínez et al. [9] and also divisible by 2 and 3 colonies. We do not use local search, since this may hide differences between other algorithmic components [12].

All experiments have been run on the instances kroAB100, kroAB200, euclidAB300 taken from Luis Paquete’s webpage at http://eden.dei.uc.pt/~paquete/tsp. The experimental comparison between pairs of algorithms is done by means of a graphical technique [14, 15] based on the empirical attainment function (EAF) [10]. The EAF of an algorithm describes the statistical distribution of its output across multiple runs by giving an estimated probability of attaining (dominating or being equal to) a point in the objective space. In each figure, for example, Fig. 2, two side-by-side plots describe differences between the EAFs of two algorithms. Each side identifies points where the probability is higher for one algorithm (the one indicated below the plot) than for the algorithm in the other side. Strong differences (plotted darker) indicate that one algorithm reaches better performance than the other one in that region of the objective space. For reasons of space, we only provide here representative examples of the results obtained. All remainder plots can be found in an online supplementary page at http://iridia.ulb.ac.be/supp/IridiaSupp2010-003.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$[\tau]$</th>
<th>$[\eta]$</th>
<th>Aggregation</th>
<th>Num. Weights</th>
<th>Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOAQ [9, 16]</td>
<td>1 $d$</td>
<td>$d$</td>
<td>weighted sum</td>
<td>$N_a$</td>
<td>Nondominated solutions</td>
</tr>
<tr>
<td>MACS [3]</td>
<td>$d$</td>
<td>$d$</td>
<td>weighted product</td>
<td>$N_a$</td>
<td>Nondominated solutions</td>
</tr>
<tr>
<td>COMPETants [5]</td>
<td>$d$</td>
<td>$d$</td>
<td>weighted sum</td>
<td>$d + 1 (\Lambda = {0, 0.5, 1})$</td>
<td>Best-of-objective</td>
</tr>
<tr>
<td>mACO-1 [1]</td>
<td>$d$</td>
<td>$d$</td>
<td>random $(\tau)$, weighted sum $(\eta)$</td>
<td>$d + 1 (\Lambda = {0, 0.5, 1})$</td>
<td>Best-of-objective (per weight)</td>
</tr>
<tr>
<td>mACO-2 [1]</td>
<td>$d$</td>
<td>$d$</td>
<td>weighted sum</td>
<td>$d + 1 (\Lambda = {0, 0.5, 1})$</td>
<td>Best-of-objective (per weight)</td>
</tr>
<tr>
<td>mACO-3 [1]</td>
<td>1</td>
<td>1</td>
<td>weighted sum</td>
<td>$N_a$</td>
<td>Nondominated solutions</td>
</tr>
<tr>
<td>mACO-4 [1]</td>
<td>1</td>
<td>1</td>
<td>random $(\tau)$</td>
<td>$1 (\Lambda = {0.5})$</td>
<td>Best-of-objective</td>
</tr>
</tbody>
</table>

For the algorithm in the other side. Strong differences (plotting darker) indicate that one algorithm reaches better performance than the other one in that region of the objective space. For reasons of space, we only provide here representative examples of the results obtained. All remainder plots can be found in an online supplementary page at http://iridia.ulb.ac.be/supp/IridiaSupp2010-003.

### 4.1 MOAQ

In the original application of Multiple Objective Ant-Q (MOAQ) [16], the design of irrigation water distribution networks, the order of the objectives was fixed as in lexicographical optimization. García-Martínez et al. [9] mention the difficulty to generalize MOAQ beyond its initial application, and, in particular, to adapt it to the multiobjective TSP in terms of Pareto optimality. They decided to use an alternative variant of MOAQ that tries to capture the main ideas for multiobjective optimization proposed by the original MOAQ. In particular, they use a single pheromone matrix and several heuristic matrices, one for each objective. For the pheromone update they use all nondominated solutions of the current iteration $P^i$, and each ant deposits an amount of pheromone inversely proportional to the sum of the values of each of the two objectives. In our implementation, we do not use the objective function values, which may be incommensurable and require normalization, and, for simplicity we let each ant deposit a constant amount of pheromone $\Delta \tau = 1.0$. García-Martínez et al. [9] further divide the ants in two groups, and each group only uses the heuristic information corresponding to one objective. We reformulate this by giving half of the ants the weight $\lambda = 0$ and the other half the weight $\lambda = 1$ for aggregating the multiple heuristic factors in the action choice rule:

$$P_k^i = \frac{[\tau_{ij}]^\alpha \cdot [(1 - \lambda_k)\eta_{ij}^1 + \lambda_k\eta_{ij}^2]^\beta}{\sum_{i \in N_k^i}[\tau_{il}]^\alpha \cdot [(1 - \lambda_k)\eta_{il}^1 + \lambda_k\eta_{il}^2]^\beta} \quad \text{if } j \in N_k^i,$$

where $N_k^i$ is the feasible neighborhood of ant $k$, that is, those cities not visited yet by ant $k$, $\tau_{ij}$ is the pheromone strength on arc $(i, j)$, $\eta_{ij}^1$ and $\eta_{ij}^2$ are the heuristic information associated with the first and second objective, and $\lambda_k$ is the weight of ant $k$, which is either one or zero.

The fact that MOAQ does not aggregate the heuristic information for each objective but instead alternates between them, leads to radical differences in the shape of the nondominated fronts that are generated depending on whether heuristic information is used ($\beta = 2$) or not ($\beta = 0$). When using heuristic information, MOAQ is unable to obtain trade-off solutions (those located in the center of the Pareto front). However, without heuristic information the results are (as expected) very poor, but the nondominated front has a very different shape (Fig. 1).

![Figure 1: Nondominated solutions obtained by 15 runs of MOAQ with $\beta = 0$ and $\beta = 2$. ACS, 25 000 iterations, kroAB200.](http://iridia.ulb.ac.be/supp/IridiaSupp2010-003)
4.2 Pareto Ant Colony Optimization

Doerner et al. [6] proposed Pareto Ant Colony Optimization (P-ACO), which is characterized by the use of multiple pheromone matrices, one for each objective, aggregated by means of a weighted sum. Each ant \( k \) uses a different weight \( \lambda_k \) for aggregating the pheromone matrices. Moreover, pheromone matrices are updated with the best and second-best solution for each objective. In the original paper, the authors used just one heuristic matrix, since it was difficult to define appropriate heuristic information for each objective in the problem. However, in later publications [17], they use multiple heuristic matrices, one for each objective, which are aggregated in the same way as the pheromone matrices, and the decision rule of P-ACO is given by

\[
p_j^k = \frac{\left[ (1 - \lambda_k) \tau_{ij}^1 + \lambda_k \tau_{ij}^2 \right]^\alpha \cdot \left[ (1 - \lambda_k) \eta_{il}^1 + \lambda_k \eta_{il}^2 \right]^\beta}{\sum_{i \in N^k} \left[ (1 - \lambda_k) \tau_{il}^1 + \lambda_k \tau_{il}^2 \right]^\alpha \cdot \left[ (1 - \lambda_k) \eta_{il}^1 + \lambda_k \eta_{il}^2 \right]^\beta}
\]

\( \tau_{ij}^1, \tau_{ij}^2 \) are the pheromone trails for arc \((i,j)\) for each pheromone matrix, \( \eta_{il}^1, \eta_{il}^2 \) are the heuristic values for the corresponding objectives, and \( \lambda_k \) is a weight, with \( 0 \leq \lambda_k \leq 1 \).

The distinction between using only one or several heuristic matrices is very important. Figure 2 shows the differences between using a single or multiple heuristic matrices. With a single heuristic matrix, a better performance is obtained in the center of the approximation to the Pareto front, while the usage of multiple heuristic information matrices leads to clearly better results on the extremes of the front.

4.3 BicriterionAnt

Iredi et al. [11], besides giving a general discussion of how to tackle MCOP problems with ACO algorithms, proposed also one specific MOACO algorithm that uses multiple pheromone and heuristic matrices, which are aggregated by weighted product as

\[
p_j^k = \frac{\left[ \tau_{ij}^1 \right]^\alpha \cdot \left[ \eta_{il}^1 \right]^\beta}{\sum_{i \in N^k} \left[ \tau_{il}^1 \right]^\alpha \cdot \left[ \eta_{il}^1 \right]^\beta}
\]

Here, each of \( \tau_{ij}^1 \) and \( \tau_{ij}^2 \) is the pheromone trail strength associated with each of the two pheromone matrices. A weight \( \lambda_k \) is associated to each ant. In the original proposal, Iredi et al. [11] suggest to update the pheromone matrices by using the nondominated front 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 (bQAP and other problems), this is not the case, and such an update will result in multiple identical pheromone matrices. García-Martínez et al. [9] therefore propose to use the objective function value of each objective to update each matrix \( \Delta \tau = 1/f_k(s_k) \).

In addition, Iredi et al. [11] proposed the use of multiple colonies, where a colony is a group of ants that construct solutions according to some particular pheromone information. Different colonies may be forced to specialize in different regions of the Pareto frontier by various means. First, when using weights to aggregate pheromone information, the set of weights available are divided among the different colonies. Second, before updating the pheromone matrices, the solutions from all colonies are put in a single archive (either the iteration-best or best-so-far archive) and dominated solutions are removed. This way all colonies contribute information to identify the best solutions. Third, the solutions are then reassigned to the various colonies either by update by origin or by update by region. In the first case, each solution is used for updating the pheromone information of the colony that generated it. In the second case, each colony may be (loosely) assigned a region of the Pareto frontier, and solutions update the pheromone information of the colony of its corresponding region. In the biobjective case, a trivial implementation 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.

The effect of using one or several colonies is shown in Figure 3. The usage of three colonies results in clearly superior performance when compared to the single colony case.

The choice between a weighted sum aggregation (analogously to P-ACO) or a weighted product aggregation (the default in the proposal of Iredi et al. [11]) of the pheromone and heuristic information may also affect the results. In BicriterionAnt, the usage of the weighted product aggregation appears to be better (Fig. 4); and the differences are stronger when algorithms stop after the same number of iterations.

4.4 Multiple Ant Colony System

The defining characteristic of Multiple Ant Colony System (MACS) [3] is the use of one heuristic matrix for each objective and a single pheromone matrix. The heuristic matrices are aggregated by weighted product, and each ant uses a different weight. The decision rule in MACS is

\[
p_j^k = \frac{\left[ \tau_{ij}^1 \right]^\alpha \cdot \left[ \eta_{il}^1 \right]^\beta}{\sum_{i \in N^k} \left[ \tau_{il}^1 \right]^\alpha \cdot \left[ \eta_{il}^1 \right]^\beta}
\]

In addition, pheromone information is updated with nondominated solutions. Finally, note that MACS can be seen as a generalization of the MOAQ defined by García-Martínez et al. [9] that uses more than two weight vectors to aggregate the heuristic information.

4.5 COMPETants

COMPETants [5] is described 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 \( \lambda = 0.5 \)) using either the first or the second heuristic matrix (thus creating two solutions). Finally, a number of ants \( N_{op} \) from each “colony” are used to update the pheromone matrix of each “colony”.

However, COMPETants can be implemented as a single colony algorithm that uses two pheromone and heuristic matrices, which are aggregated by weighted sum. The only weights used are \( \Lambda = \{0, 0.5, 1\} \). The ants using \( \lambda = 0.5 \) are special in the sense that they do not aggregate both heuristic matrices but they create one solution using only one of the heuristic matrices. Finally, each pheromone matrix would be updated with the \( N_{op} \) best solutions for the corresponding objective (best-of-objective update), which is basically the same update method used by P-ACO.

For the sake of simplicity, we disregard the fact that in the original 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.
4.6 mACO Variant 1 (mACO-1)

Alaya et al. [1] proposed four alternatives for the design of a MOACO algorithm. The first variant (mACO-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 considers heuristic information for its corresponding objective, whereas the extra “colony” aggregates these multiple heuristic factors into a single heuristic factor. The pheromone information of each colony is updated by solutions of the same colony. Only one solution is used to update each colony, the one corresponding to the associated objective. The update of the extra colony is slightly different. It keeps a best solution for each objective, and updates each pheromone matrix with the corresponding solution.

This proposal can be formulated in terms of a single colony algorithm with multiple pheromone matrices and multiple heuristic matrices. The algorithm makes use of three weights \( \lambda = \{ 0, 0.5, 1 \} \), a random aggregation rule for the pheromone information and a weighted sum aggregation for the heuristic information. Therefore, the decision rule in mACO-1 can be defined as

\[
p^j_h = \frac{[\tau^j_h]^\alpha \cdot [1 - \lambda_k \eta^j_h + \lambda_k \eta^j_h]^\beta}{\sum_{l \in \Lambda^h_k} [\tau^l_h]^\alpha \cdot [1 - \lambda_k \eta^l_h + \lambda_k \eta^l_h]^\beta} \quad \text{if } j \in \Lambda^h_k, (1)
\]

where \( \tau^r \) is a pheromone matrix that is randomly chosen with a probability \( \lambda \)

\[
\tau^r \sim \text{rand}(\lambda)\{\tau^1, \tau^2\} = \begin{cases} \tau^1 & \text{if } U(0, 1) < 1 - \lambda, \\ \tau^2 & \text{otherwise} \end{cases}
\]

where \( U(0, 1) \) is an uniform random value within \([0, 1]\) and \( \lambda \) is a parameter that biases the choice.

4.7 mACO Variant 2 (mACO-2)

The second variant (mACO-2) by Alaya et al. [1] only differs from mACO-1 in the pheromone aggregation. In mACO-2, pheromone matrices are aggregated by summing the pheromone matrices of each objective. In fact, this corresponds to a weighted sum aggregation using a weight vector \((1, 1)\). For both MMMAS and ACS, there is little difference between this and using \( \lambda = 0.5 \) [4]. Hence, mACO-2 is very similar to P-ACO (Section 4.2) but using only 3 weights \((A = \{0, 0.5, 1\})\), and several ants per weight.

Differences in quality between mACO-1 and mACO-2 appear to be minor. The shape of the nondominated fronts is essentially the same and there are only minor differences among the performance of the two algorithms (Fig. 5).

4.8 mACO Variant 3 (mACO-3)

The third variant proposed by Alaya et al. [1] uses instead a single pheromone matrix. This pheromone matrix is updated by using nondominated solutions. In particular,

\[
\Delta \tau_{ij} = \begin{cases} 1 & \text{if } \{i, j\} \text{ appears in a solution in } P \\ 0 & \text{otherwise} \end{cases}
\]

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. [1] emphasize that every pheromone value is updated only once despite how many solutions contain it. This is in contrast with other algorithms that use a “nondominated update” such as MOAQ, MACS, and BicriterionAnt. The heuristic factor is also a single matrix. Since in some problems there are several heuristic factors per objective, these are aggregated in a fix manner into
MOACO algorithms share more similarities than differences. Therefore, BicriterionAnt performs better on a wider range of the objective space. However, the solutions generated are not very widely spread and therefore BicriterionAnt performs better on a wider range of the objective space. A very different behavior is given by P-ACO with a single pheromone matrix. With this variant, very good results in the center of the front are obtained, improving quite strongly over those of BicriterionAnt in this area. However, the solutions generated are not very widely spread and therefore BicriterionAnt performs better on a wider range of the objective space.

5. COMPARISON OF ALGORITHMS

As a final step, we compare the MOACO algorithms. Given the limited space, a pairwise comparison of the MOACO algorithms was deemed not to be feasible. Instead, we choose the BicriterionAnt algorithm with three colonies (because of its overall very good performance) as the baseline to which we compare all others. In Figures 7 to 9 we show the differences in the empirical attainment functions obtained on the largest instance, euclidAB300, where also the differences among the algorithms have been most pronounced.

BicriterionAnt is clearly superior to MACS and COMPETants across the largest part of the nondominated front. Only in the extremes towards the best solutions for each of the single objectives, MACS and to a somewhat larger extent COMPETants obtain slightly better solutions. When compared to the mACO-x variants, BicriterionAnt obtains clearly better results. In particular, mACO-4 is dominated by the results of BicriterionAnt (similar results hold for mACO-3, which generates fronts of a similar shape as mACO-4); mACO-2 performs in very small parts of the nondominated front slightly better than BicriterionAnt.

Finally, the comparison to P-ACO with either one or multiple heuristic matrices confirms the large differences among these two variants. Except for the extremes of the nondominated front towards each of the single objectives, P-ACO with multiple pheromone matrices is clearly of inferior performance to BicriterionAnt. This result can be seen as an indication that the combination of solution components that are very good for each of the single objectives are not necessarily appropriate to generate good solutions in the center of the nondominated front. A very different behavior is given by P-ACO with a single pheromone matrix. With this variant, very good results in the center of the front are obtained, improving quite strongly over those of BicriterionAnt in this area. However, the solutions generated are not very widely spread and therefore BicriterionAnt performs better on a wider range of the objective space.

6. CONCLUSIONS

There are a number of interesting observations that can be taken from our review. The first is that many existing MOACO algorithms share more similarities than differences.
By using the definition of multiple colonies proposed by Iredi et al. [11], some of the "multi-colony" MOACO algorithms cannot be said to use multiple colonies, and in fact, they become extreme variants (using 3 weights) of P-ACO [17]. This is the case of COMPETants, mACO-1, and mACO-2.

Another conclusion is that the variant of MOAQ proposed by García-Martínez et al. [9] is an extreme case (with as many weights as objectives) of MACS [3].

Our computational analysis confirms some conclusions obtained by García-Martínez et al. [9]. For example, COMPETants and MOAQ are rather poor performing algorithms for the bTSP, whereas MACS and BicriterionAnt are among the best performing ones. However, our analysis shows that the poor performance of COMPETants and MOAQ is because they do not aggregate the multiple heuristic information they use, which makes them unable to obtain solutions in the center of the nondominated front. On the other hand, algorithms that aggregate multiple heuristic matrices by means of several weights, such as MACS and BicriterionAnt, obtain a much better distribution of solutions along the nondominated front. Finally, the algorithms that aggregate the heuristic information in a fixed way (mACO-3, mACO-4, and the single heuristic variant of P-ACO [9]) obtain very narrow Pareto fronts but of very high quality. Our systematic analysis allows us to claim that these results are not caused by some particularity of the underlying ACO algorithm but entirely due to the multiobjective algorithmic design choices of each proposed algorithm.

Another interesting observation is that the use of a mod-
erate number of colonies in BicriterionAnt, which implies several pairs of pheromone matrices, turns out to improve the results noticeably along the whole nondominated front. This result is in stark contrast with the fronts obtained by P-ACO, mACO-1 and mACO-2, which use just one pair of pheromone matrices.

For the bTSP, we conclude that the use of multiple heuristic factors is essential to obtain a wide spread Pareto front. This is not enough, however, and several weights (definitely more than 3) must be used to aggregate these heuristic factors. As it has been observed before, the use of a small number of colonies further improves the results. We strongly believe, that these conclusions apply generally to problems where the heuristic information is important.

The analysis presented here should be extended to problems with different characteristics. Particularly interesting would be the application to the bQAP, where heuristic information is not useful. Moreover, further research should test the interactions between these components and local search. The results here should provide guidelines for problems similar to the bTSP, but more work is necessary to have a complete understanding for other types of problems.

Acknowledgments. This work was supported by the METAX project, an Action de Recherche Concertée funded by the Scientific Research Directorate of the French Community of Belgium. Thomas Stützle acknowledges support from the Belgian F.R.S.-FNRS, of which he is a Research Associate. The authors also acknowledge support from the FRFC project “Méthodes de recherche hybrides pour la résolution de problèmes complexes”.

References


