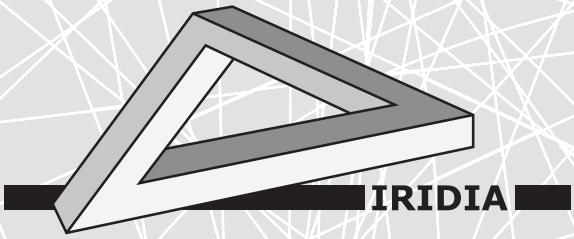

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Automatic Configuration of Multi-Objective ACO Algorithms

Manuel López-Ibáñez and Thomas Stützle

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2010-011

April 2010

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2010-011

Automatic Configuration of Multi-Objective ACO Algorithms

Manuel López-Ibáñez and Thomas Stützle

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{manuel.lopez-ibanez,stuetzle}@ulb.ac.be

Abstract. In the last few years a significant number of ant colony optimization (ACO) algorithms have been proposed for tackling multi-objective optimization problems. In this paper, we propose a software framework that allows to instantiate the most prominent multi-objective ACO (MOACO) algorithms. More importantly, the flexibility of this MOACO framework allows the application of automatic algorithm configuration techniques. The experimental results presented in this paper show that such an automatic configuration of MOACO algorithms is highly desirable, given that our automatically configured algorithms clearly outperform the best performing MOACO algorithms that have been proposed in the literature. As far as we are aware, this paper is also the first to apply automatic algorithm configuration techniques to multi-objective stochastic local search algorithms.

1 Introduction

The growing interest on solving optimization problems with respect to multiple, conflicting objectives has led researchers to propose extensions of well-known metaheuristics to tackle them. So far, evolutionary algorithms have received most of the research effort. However, there are also several proposals for applying the ant colony optimization (ACO) metaheuristic [1] to multi-objective combinatorial optimization problems (MCOPs). The majority of these multi-objective ACO (MOACO) algorithms deal with Pareto optimality, that is, they do not make a priori assumptions about the decision maker's preferences.

There are a number of design questions when extending ACO algorithms to MCOPs. First, the meaning of the pheromone information associated to a solution component is unclear in the multi-objective context, because the objective function is multi-dimensional and not scalar, and the output of the algorithm is a nondominated set of solutions and not a single solution. Some MOACO algorithms use several pheromone matrices, each of them associated to a different objective [2,3,4]. If multiple pheromone or heuristic matrices are used, they are aggregated during the solution construction by means of weights [2,4]. However, there are strong differences among MOACO algorithms in the way this aggregation takes place and how many different weights are used. Finally, a further design question is which ants are selected for depositing pheromone. Existing MOACO algorithms select some (or all) nondominated solutions [4], or they select the best solutions with respect to the objective associated to the pheromone matrix that is updated [2,3].

In previous work, we examined alternative design choices of a MOACO algorithm for the bi-objective TSP (bTSP) [5]. Later, we extended this formulation in order to replicate the design of several existing MOACO algorithms from the literature [6]. Our examination of algorithmic components concludes that there are many similarities among the algorithms proposed in the literature. In addition, our empirical results show that, for the bTSP, some algorithmic components achieve clearly superior results in comparison to others.

In this paper, we combine the results of our previous work into an algorithmic framework that may be configured appropriately to reproduce existing MOACO algorithms, and more importantly, it may be configured combining ideas from diverse MOACO algorithms. Thus, this framework facilitates the application of automatic algorithm configuration techniques to configure multi-objective algorithms. However, previous works that aim to produce high performing algorithms by combining a flexible software framework and an automatic tuning tool have dealt so far only with single-objective optimization problems [7]. To the best of our knowledge, there is no previous work on the automatic configuration of stochastic local search (SLS) algorithms for multi-objective optimization problems in terms of Pareto-optimality. Thus, this paper is the first application of such automatic configuration techniques to multi-objective algorithms. We tackle this challenge in a pragmatic way, using unary quality measures, which assign a scalar value to a nondominated set according to some reference criterion, as the optimization goal of an automatic configuration tool. In particular, we configure the proposed framework by applying Iterated F-Race (I/F-Race) [9] using both the hypervolume and the epsilon measures [8].

In summary, the main contributions of this paper are that (*i*) for the first time, we automatically configure MOACO algorithms using unary quality measures, a flexible framework for MOACO algorithms, and a high-performing automatic configuration tool; and that (*ii*) we show that the configurations found automatically are better than configurations implementing several MOACO algorithms proposed in the literature.

2 Experimental studies on MOACO algorithms

The available MOACO algorithms differ in a wide diversity of design choices. However, few authors experimentally justify their design choices or compare them to alternatives. Iredi et al. [4] investigated several design alternatives for a multi-colony approach. López-Ibáñez et al. [10] compared different design options for MOACO algorithms on the bi-objective quadratic assignment problem (bQAP). Alaya et al. [3] compared the performance of four MOACO variants for the multi-objective knapsack problem. The review by García-Martínez et al. [11] classifies existing MOACO algorithms according to the usage of one or several pheromone (or heuristic) matrices, and provides a comparison of some of the original algorithms using the bTSP as a case study. The algorithms they tested differ substantially with respect to underlying ACO parameters, e.g., some of them are based on the classical Ant System, whereas others build upon the typically better performing $\mathcal{MAX}$-$\mathcal{MIN}$ ($\mathcal{MM}$AS) and Ant Colony System (ACS). Therefore, one cannot conclude from the quality of the achieved results on the impact that specific design decisions of each MOACO algorithm have on MOACO performance. In recent research, we first studied alternative design choices for extending ACO algorithms to MCOPs [5] in a systematic manner by keeping other design choices fixed. The design choices studied comprise also those that have been proposed in previous MOACO algorithms. Next, we examined the particular *combinations* of design choices that define existing MOACO algorithms, keeping other factors, such as the underlying ACO algorithm, fixed [6]. The results of these studies showed important differences between various design choices.

3 A configurable MOACO framework

Based on our previous work, we identified particular design choices that are clearly superior to others for our case study, the bTSP. At the same time, we realized that some combinations of design choices are promising, but there is no corresponding MOACO algorithm in the literature.

Algorithm 1 MO-ACO framework

while not stopping criteria met **do**
  for each colony $c \in \{1, \ldots, N^{\text{col}}\}$ **do**
    for each ant $k \in \{1, \ldots, N^{\text{a}}\}$ **do**
      $\boldsymbol{\lambda} :=$ NextWeight(Λ, k, *iteration*)
      $\tau := \begin{cases} \text{Aggregation}(\boldsymbol{\lambda}, \{\tau^1, \ldots, \tau^d\}) & \text{if } \texttt{multiple } [\tau] \\ \tau & \text{if } \texttt{single } [\tau] \end{cases}$
      $\eta := \begin{cases} \text{Aggregation}(\boldsymbol{\lambda}, \{\eta^1, \ldots, \eta^d\}) & \text{if } \texttt{multiple } [\eta] \\ \eta & \text{if } \texttt{single } [\eta] \end{cases}$
      $s :=$ ConstructSolutiond(τ, η)
      $s :=$ WeightedLocalSearch($s, \boldsymbol{\lambda}$) // Optional
      $S^{\text{global}} := S^{\text{global}} \cup \{s\}$
    end for
  end for
  for each colony $c \in \{1, \ldots, N^{\text{col}}\}$ **do**
    $S_c :=$ MultiColonyUpdate(S^{global})
    $S_c^{\text{upd}} :=$ Selection(S_c)
    UpdatePheromones($S_c^{\text{upd}}, N^{\text{upd}}$)
  end for
end while
Output: P^{bf}

Therefore, the next logical step is to propose a configurable MOACO framework that allows us to instantiate several existing algorithms, and variants that have never been proposed before.

We propose the MOACO framework described in Algorithm 1. The framework allows the use of multiple colonies (N^{col}), where each colony constructs solutions independently of others according to its own pheromone information. The colonies cooperate in two ways: (1) by exchanging solutions for updating the pheromone information, and (2) by using a common archive of nondominated solutions for detecting dominated ones. This multi-colony architecture is inspired by the proposal of Iredi et al. [4], which we found more flexible and consistent than other definitions of "colony" [6]. Within each colony, a number of ants construct solutions from the pheromone and heuristic information of their own colony. This pheromone information may be represented either as a single matrix or as multiple matrices that must be aggregated somehow (Aggregation). The same applies to the heuristic information. There are several forms of aggregation, and all of them require the use of a weight. The sequence of weights is determined by the procedure NextWeight. Once there is a unique pheromone and heuristic matrix, a solution is constructed (ConstructSolution), possibly improved by a local search (WeightedLocalSearch) and added to a common archive S^{global}. Once all ants from all colonies have finished constructing solutions, procedure MultiColonyUpdate distributes these solutions among the colonies. From the solutions assigned to each colony, procedure Selection decides which ones will be used for updating the pheromone information (UpdatePheromones). The MOACO algorithm continues until a certain number of iterations or a time limit is reached.

Table 1 describes the configurable settings of the proposed framework. Some settings are only significant for certain values of other settings. For example, the possible settings of MultiColonyUpdate only make a difference when $N^{\text{col}} > 1$, otherwise all solutions are assigned to the single colony. Both settings, *origin* and *region*, were originally proposed by Iredi et al. [4]. Update by origin assigns each solution from S^{global} to its original colony, whereas update by region divides S^{global} in equal parts among the colonies in such a way that each colony roughly

Table 1. Algorithmic components of the proposed MOACO framework.

Component	Domain	Description
N^{col}	$\mathbb{N}^+$	Number of colonies
MultiColonyUpdate	{ origin, region }	How solutions are assigned to colonies for update [4]
N^{upd}	$\mathbb{N}^+$	Max. number of solutions that update each $[\tau]$ matrix
Selection	nondominated solutions, Best-of-objective, Best-of-objective-per-weight	Which solutions are selected for updating the pheromone matrices
$\lvert\Lambda\rvert$	$\mathbb{N}^+$	Number of weights per colony
NextWeight	one weight per iteration, all weights per iteration	How weights are used at each iteration
$[\tau]$	{ single, multiple }	Number of pheromone matrices
$[\eta]$	{ single, multiple }	Number of heuristic matrices
Aggregation	weighted sum, weighted product, random	How weights are used to aggregate different matrices

corresponds to one region of the objective space. The alternatives for the Selection component are:

Nondominated solutions. The solutions used for updating the pheromone information are the nondominated solutions in S_c^{upd}. When there are more nondominated solutions than N^{upd}, we apply the truncation mechanism of SPEA2 [12] to select only N^{upd} solutions. It is possible to combine this Selection method and multiple pheromone matrices [4], however, in the case of the bTSP and with $\Delta\tau = 1$, this would result in updating both matrices with the same value, so we do not empirically explore this combination.

Best-of-objective. Selects from the current S_c^{upd}, the N^{upd} best solutions with respect to each objective. In the case of multiple pheromone matrices, each pheromone matrix is updated using the N^{upd} solutions associated to the corresponding objective. Otherwise, the $d \cdot N^{\text{upd}}$ solutions update the single pheromone matrix.

Best-of-objective-per-weight. We keep a list for each weight λ and each objective of the N^{upd} best solutions for each objective generated using λ. In the particular case of $\lambda = 0$, we only keep one list for the first objective, and we do the same for $\lambda = 1$ and the second objective. When using multiple pheromone matrices, each matrix is updated using only solutions from lists associated to the same objective. This update method is used by the existing mACO-1 and mACO-2 algorithms [3]. It is not clear how this approach should be extended to multiple colonies, since then solutions may be exchanged among colonies with different weights.

The set of weights (Λ) is finite and equally distributed in the interval $[0, 1]$. If there are multiple colonies, Λ is partitioned among the colonies. The options tested for NextWeight are either that all ants in one colony use the same weight at a certain iteration (*one-weight-per-iteration*), or that all weights are used at each iteration (*all-weights-per-iteration*). In the case of one-weight-per-iteration, the weight used by each colony in successive iterations follows an ordered sequence of the elements of Λ, and the order is reversed when the last weight in the sequence is reached. In the case of *all-weights-per-iteration*, when the number of ants N^{a} is larger than the number of weights $|\Lambda|$, then several ants will use the same weight. The aggregation of the pheromone matrices is computed once per weight per iteration.

Lastly, our framework includes three options for Aggregation:

Table 2. Taxonomy of MOACO algorithms as instantiations of the proposed framework (d is the number of objectives, N^{a} is the number of ants).

Algorithm	$\lvert\tau\rvert$	$\lvert\eta\rvert$	Aggregation	$\lvert\Lambda\rvert$	Selection
MOAQ [13,11]	1	d	–	d ($\Lambda = \{0, 1\}$ if $d = 2$)	nondominated solutions
P-ACO [2]	d	d	weighted sum	N^{a}	Best-of-objective
MACS [14]	1	d	weighted product	N^{a}	nondominated solutions
BicriterionAnt [4]	d	d	weighted product	N^{a}	nondominated solutions
COMPETants [15]	d	d	weighted sum	$d+1$ ($\Lambda = \{0, 0.5, 1\}$)	Best-of-objective
mACO-1 [3]	d	d	random (τ)/w. sum (η)	$d+1$ ($\Lambda = \{0, 0.5, 1\}$)	Best-of-objective-per-weight
mACO-2 [3]	d	d	weighted sum	$d+1$ ($\Lambda = \{0, 0.5, 1\}$)	Best-of-objective-per-weight
mACO-3 [3]	1	1	–	–	nondominated solutions
mACO-4 [3]	d	1	random (τ)	d	Best-of-objective

Weighted sum. The two pheromone (or heuristic) matrices are aggregated by a weighted sum: $(1-\lambda)\tau^1_{ij} + \lambda\tau^2_{ij}$.

Weighted product. The two pheromone (or heuristic) matrices are aggregated by a weighted product: $(\tau^1_{ij})^{(1-\lambda)} \cdot (\tau^2_{ij})^{\lambda}$

Random. At each construction step, an ant selects the first of the two pheromone matrices if $U(0,1) < 1-\lambda$, where $U(0,1)$ is a uniform random number, otherwise it selects the other matrix.

Our previous work [6] has examined the design of several multi-objective ACO algorithms from the literature. This previous study has informed the design of the proposed framework. As a result, the framework is flexible enough to allow us to replicate those MOACO algorithms. We provide in Table 2 the configuration of algorithmic components necessary to instantiate several well-known MOACO algorithms.

4 Automatic Configuration of MOACO framework

Instead of using a trial-and-error approach to identify good instantiations of the proposed MOACO framework, we follow the work of KhudaBukhsh et al. [7] (and earlier work discussed in that paper) in the sense that we use an efficiently implemented, flexible software framework together with an automated algorithm configuration tool for obtaining very high-performing algorithmic variants. However, this is the first time that such an approach is applied in a multi-objective context.

Specifically, we automatically configure our MOACO framework using a new implementation of Iterated F-race (I/F-Race) [9] as the automatic configuration method. As the evaluation criteria used by I/F-Race, we tested two unary measures for evaluating the output of multi-objective algorithms, namely, the hypervolume and the (additive) epsilon measure [8]. The hypervolume is the volume of the objective space weakly dominated by a nondominated set and bounded by a reference point that is strictly dominated by all known points. The larger the hypervolume, the better is the corresponding nondominated set. The additive epsilon measure provides the minimum value that must be subtracted from all objectives of a nondominated set so that it weakly dominates a reference set. This reference set is usually the nondominated set of all known solutions. A smaller epsilon measure is preferable.

Each run of I/F-Race uses a maximum budget of 1 000 experiments and it is repeated five times, for each quality measure, to assess the variability of the automatic configuration process. As training instances, we generated 36 bTSP Random Uniform Euclidean instances for each of

$n = \{100, 200, 300\}$ nodes (108 instances in total). Each experiment, that is, each run of the MOACO framework on each instance, is stopped after n CPU-seconds. We provide to I/F-Race appropriate domains of the MOACO framework components. In particular, we use the domains described in Table 1 for the categorical components, with the restrictions described in the text for the Selection component. For the remaining components, we use the following domains: $N^{\text{col}} = \{1, 2, 3, 5, 10\}$, $|\Lambda| = \{2, 6, 12, 24\}$, $N^{\text{upd}} = \{1, 2, 5, 10\}$. We use $\mathcal{MM}$AS and its default parameter settings for the TSP [16] as the underlying ACO algorithm with some exceptions. We use 24 ants per colony, which is a value close to the one used in the MOACO literature for the bTSP [11] and it allows us to divide the ants exactly among several values of $|\Lambda|$. We also use $\Delta\tau = 1$ for the amount of pheromone deposited by an ant, and the evaporation rate is set to $\rho = 0.05$. As for the optional local search, we use a 2-opt algorithm that exploits candidate lists of length 20. More details on the local search are given in our previous work [5].

We perform runs of I/F-Race with and without local search. The rationale for separating both analyses is that, on the one side, it is clear that MOACO algorithms with local search by far outperform those without local search for the bTSP [5]; on the other side, given the large amount of work on MOACO algorithms without the usage of local search [11], it is interesting to examine whether the automated configuration of the MOACO framework may obtain competitive results in comparison to those algorithms.

We have implemented the MOACO framework in `C` using ACOTSP[1] as the underlying ACO package, and compiled it with `gcc`, version 3.4. All experiments reported in the following are carried out on AMD Opteron 2216 dual-core 2.4 GHz processors with 2 MB L2-Cache under Rocks Cluster GNU/Linux. The implementation is sequential and experiments run on a single core.

The results of the automatic configuration with respect to hypervolume and epsilon measures are very similar. In particular, the configurations obtained when maximizing the hypervolume are very consistent. In all five runs of I/F-Race, the best configuration uses multiple colonies (varying among three, five and ten colonies), selection by best-of-objective, one weight per iteration, multiple pheromone and heuristic matrices, and aggregation by weighted product. The differences are in the multi-colony update, I/F-Race chooses three times *update by region*, and two times *by origin*; in the number of solutions used for update, which varies among $N^{\text{upd}} = \{1, 2, 5, 10\}$; and in the number of weights, which varies among $|\Lambda| = \{2, 6, 12\}$. The results of I/F-Race when optimizing for the epsilon measure are more varied. The settings that were common to all five runs are multiple colonies (either five or ten), multiple heuristic matrices, and aggregation by weighted product. One run of I/F-Race chose the use of nondominated solutions for the Selection component; surprisingly this was selected together with $N^{\text{upd}} = 1$, which effectively means that the chosen solution would be the best for one of the objectives. That same configuration uses a single pheromone matrix and multiple heuristic matrices. The other four runs of I/F-Race with epsilon measure follow the results obtained with the hypervolume measure and produce configurations using selection by best-of-objective and multiple pheromone matrices. For the other parameters, the automatically configured settings vary among $N^{\text{upd}} = \{1, 2, 5, 10\}$ and $|\Lambda| = \{6, 12, 24\}$.

We repeat the automatic configuration with 2-opt local search enabled. In this case, the resulting configurations are even more varied. This probably indicates that once local search is enabled, the particular settings of the other parameters are less important. Focusing on the common settings, all resulting configurations use multiple colonies (nine times ten colonies, and once five colonies). Moreover, in most configurations found using the hypervolume measure there are multiple pheromone and heuristic matrices and Selection uses best-of-objective, whereas in

[1] Available from `http://www.aco-metaheuristic.org/aco-code`.

those found using the epsilon measure is more common the use of a single pheromone matrix and Selection uses nondominated solutions.

5 Comparison with existing MOACO algorithms

We now compare the automatically configured MOACO algorithms with the results obtained by existing MOACO algorithms from the literature. A recent comparison of MOACO algorithms for the bTSP identified BicriterionAnt as the best-performing overall [6], so we will use this algorithm for comparison.

First, we choose the best automatically configured MOACO algorithm by performing 15 runs of each configuration returned by I/F-Race on each of the 108 tuning instances. From these results, we choose the configuration (AutoMOACO) with the largest mean hypervolume and smallest mean epsilon measure. In this case, both measures agree on the best configuration: ten colonies, multi-colony update by origin, selection of best-of-objective, one-weight-per-iteration, $N^{\text{upd}} = 2$, $|\Lambda| = 12$, aggregation by weighted product, and multiple pheromone and heuristic matrices.

We compare BicriterionAnt and AutoMOACO on three instances (`kroAB100`, `kroAB200`, and `euclidAB300`), different from the ones used for tuning, and taken from Luis Paquete's webpage (`http://eden.dei.uc.pt/~paquete/tsp`). As before, we perform 15 runs of each algorithm on each test instance and each run has a time limit of n seconds. For the comparison, instead of relying on the same quality measures used for tuning, we examine the differences between the algorithms' empirical attainment functions (EAFs) [17,18]. The EAF estimates from several independent runs the probability of a multi-objective algorithm obtaining or dominating a certain point in the objective space [19]. A plot of the EAF differences between two algorithms gives on each side the differences in favour of either algorithm. Larger differences are indicated in darker color. Large differences indicate one algorithm has a higher probability of dominating a certain region of the objective space than the other algorithm. Figure 1 gives plots comparing BicriterionAnt and AutoMOACO on the test instances. In all plots, there are only differences in favor of AutoMOACO (dark points occur only on the right plots), which indicates that AutoMOACO completely outperforms BicriterionAnt.

In a second comparison, we consider the MOACO variants with local search. We repeat the same procedure described above for choosing the best configuration from the ten runs of I/F-Race (five runs with each quality measure), that is, we perform 15 independent runs of each configuration on the 108 tuning instances, and choose the configuration with the best mean value of the quality measures. Both measures again agree on the best automatically configured MOACO using local search (AutoMOACO+ls): ten colonies, multi-colony update by region, one-weight-per-iteration selection of nondominated solutions, $N^{\text{upd}} = 1$, $|\Lambda| = 12$ and a single pheromone matrix.

We compare AutoMOACO+ls, on the three test instances mentioned above, to the algorithm suggested by our earlier studies on MOACO components [5]. Based on that paper, we select a configuration with weighted sum aggregation (given that weighted product aggregation was not studied there), one colony, multiple pheromone and heuristic matrices, and one weight per iteration. Although it is difficult to appreciate the EAF differences in the graphical plots, as illustrated by Fig. 2, the differences are strongly in favor of AutoMOACO+ls and there are only small differences in favor of the literature-best version in very few points of the objective space. To confirm these differences, we apply a non-parametric Wilcoxon rank-sum test on the hypervolume values obtained by AutoMOACO+ls and the literature-best version. The test rejects the hypothesis that both algorithms reach similar hypervolume at a significant level of 0.05. Since AutoMOACO+ls obtains a lower mean hypervolume than the literature-best version, we

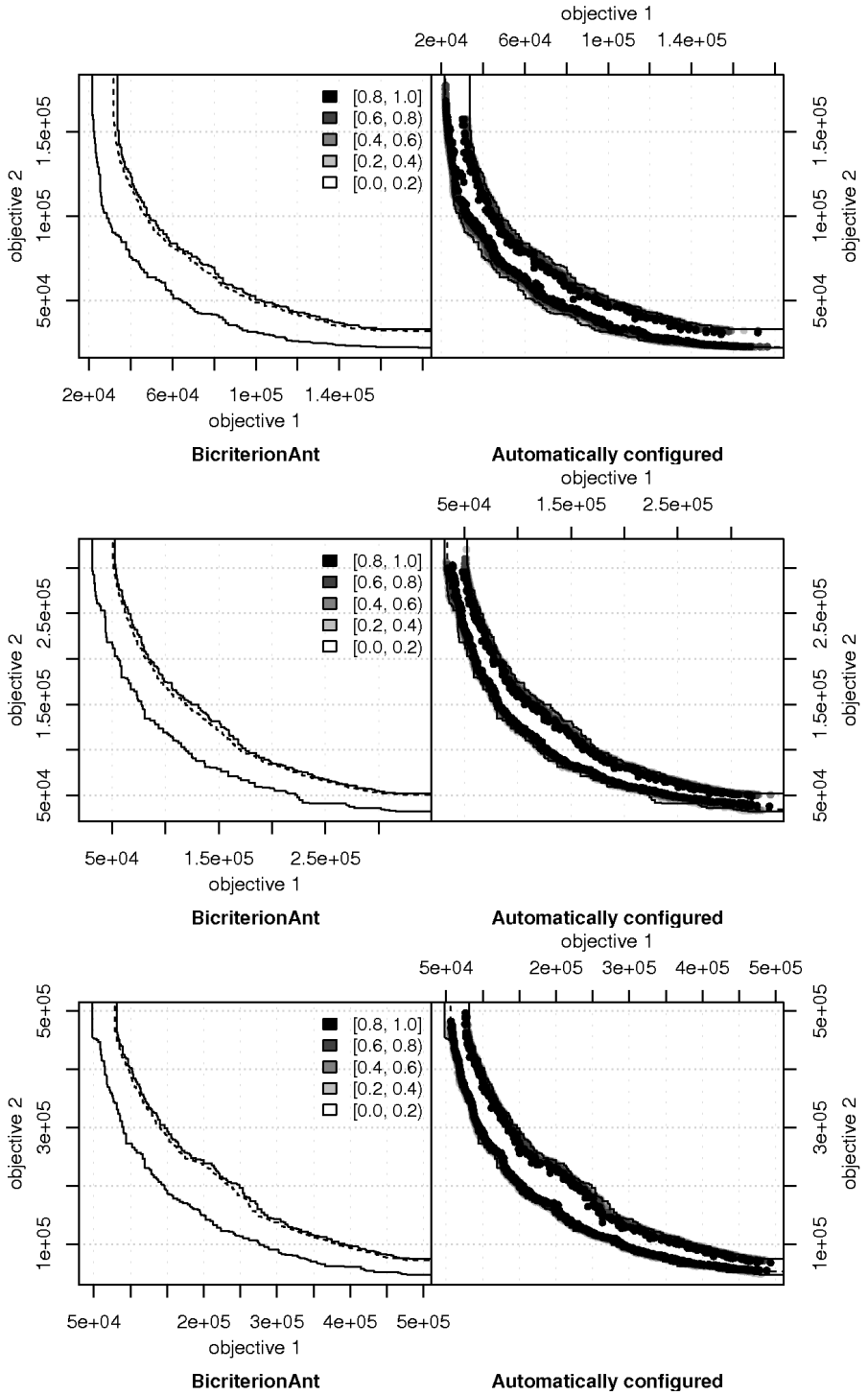

Fig. 1. The plots show the differences between the EAF of BicriterionAnt vs. AutoMOACO on instances `kroAB100` (top), `kroAB200` (middle) and `euclidAB300` (bottom). The magnitude of the differences is encoded in grey-scale.

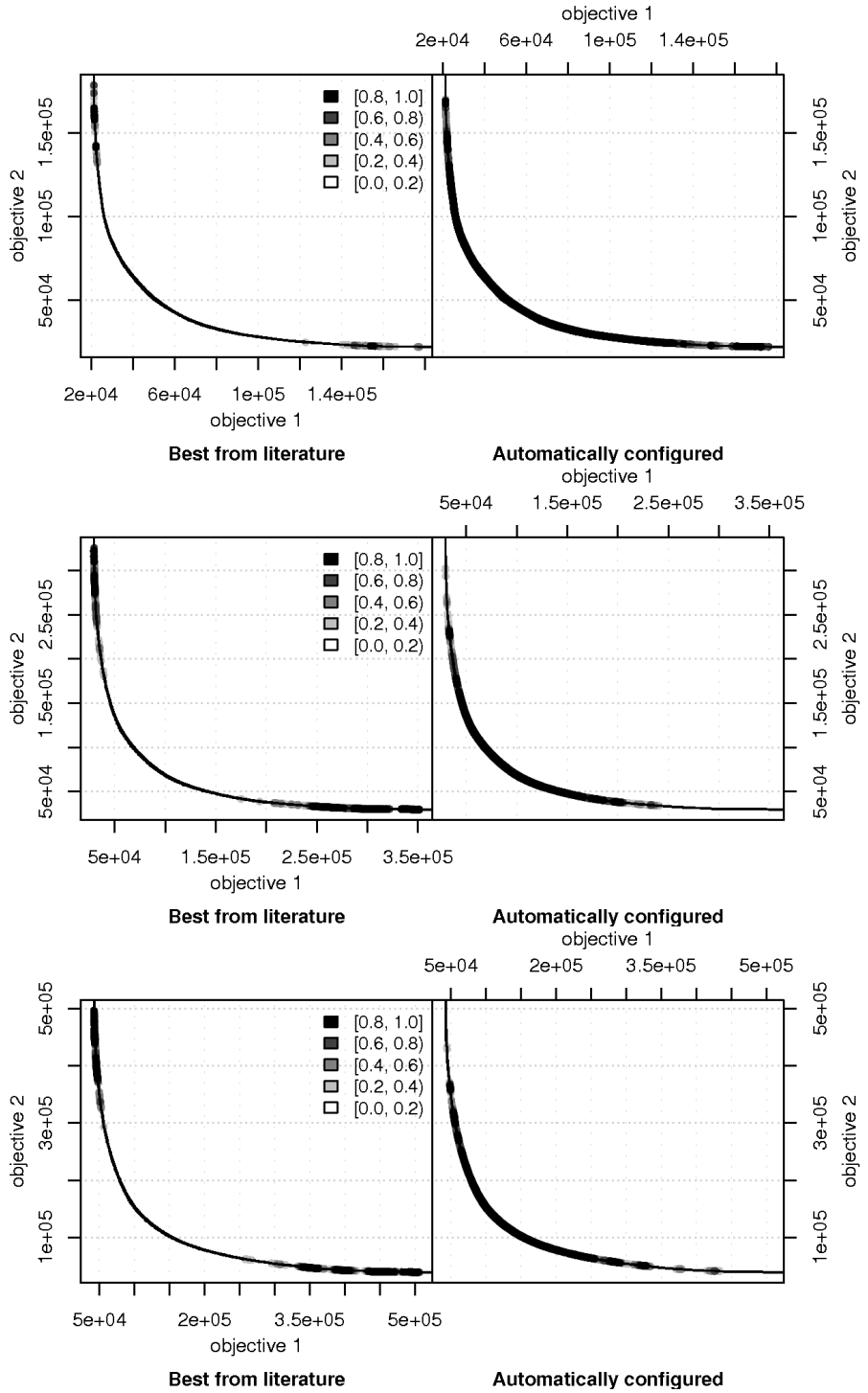

Fig. 2. The plot shows the differences between the EAFs of two MOACO algorithms using local search: best configuration from the literature [5] vs. AutoMOACO+ls. Results are shown for `kroAB100` (top), `kroAB200` (middle) and `euclidAB300` (bottom). The magnitude of the differences is encoded in grey-scale.

conclude that there is a statistically significant difference in favor of AutoMOACO+ls. We repeat this procedure for each instance and for each quality measure (hypervolume and epsilon). In all cases the statistical tests are in favor of AutoMOACO+ls, which confirms that AutoMOACO+ls outperforms the best MOACO algorithm with local search from the literature in these bTSP instances.

6 Conclusions

In this paper, we propose a flexible, configurable ACO framework for multi-objective problems in terms of Pareto optimality. This framework is the result of our review and synthesis effort of different MOACO algorithms from the literature. The framework may be configured to instantiate those existing algorithms, and others never examined before in the literature. We demonstrate the potential of this approach by automatically configuring our framework for the bTSP and comparing it with a state-of-the-art MOACO algorithm for this problem.

We automatically configure our MOACO framework by applying I/F-Race using unary quality measures to evaluate the nondominated sets returned by multi-objective optimization algorithms. We repeated the automatic configuration with two different unary measures, the hypervolume and the epsilon measure, and we found that there are some small differences in the quality of the algorithmic configurations obtained. Although it may be possible to use both measures at the same time, such approach would require defining a consensus method in case both measures contradict each other. The proposed approach is simple, pragmatic and effective. However, further work is underway to investigate the use of different unary and binary measures, and how to adequately combine several quality measures at the same time.

The result of the automatic configuration is a MOACO algorithm that mixes features from existing algorithms and is able to outperform them in the bTSP. This result is a further confirmation that automatic configuration of a flexible framework may surpass human-designed algorithms [7]. The main contribution of this paper, however, is not the fine-tuned algorithm, but rather the MOACO framework itself and the method proposed here for automatically fine-tuning it. First, the proposed framework can be applied to any problem for which the reviewed MOACO algorithms have been applied so far. Second, the proposed configuration method may be used to fine-tune the framework to the particular problem and, possibly, improve over existing results. Moreover, the proposed configuration method can be applied to any multi-objective optimization algorithm, which so far are either not fine-tuned at all or by trial-and-error. Future work would extend the MOACO framework to include more diverse algorithmic components, and incorporate ideas from multi-objective evolutionary algorithms.

Acknowledgments. This work was supported by the META-X 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

1. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press (2004)
2. Doerner, K.F., Gutjahr, W.J., Hartl, R.F., Strauss, C., Stummer, C.: Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. Annals of Operations Research **131** (2004) 79–99

3. Alaya, I., Solnon, C., Ghédira, K.: Ant colony optimization for multi-objective optimization problems. In: ICTAI 2007. Vol. 1. IEEE Computer Society Press, Los Alamitos, CA (2007) 450–457
4. Iredi, S., Merkle, D., Middendorf, M.: Bi-criterion optimization with multi colony ant algorithms. In Zitzler, E., et al., eds.: EMO 2001. Vol. 1993 of LNCS. Springer, Heidelberg (2001) 359–372
5. López-Ibáñez, M., Stützle, T.: An analysis of algorithmic components for multiobjective ant colony optimization: A case study on the biobjective TSP. In: EA 2009. Vol. 5975 of LNCS., Springer, Heidelberg (to appear) 134–145
6. López-Ibáñez, M., Stützle, T.: The impact of design choices of multi-objective ant colony optimization algorithms on performance: An experimental study on the biobjective TSP. In: GECCO 2010. ACM press, New York, NY (2010) Accepted.
7. KhudaBukhsh, A.R., Xu, L., Hoos, H.H., Leyton-Brown, K.: SATenstein: Automatically building local search SAT solvers from components. In: IJCAI-09. (2009) 517–524
8. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation **7**(2) (2003) 117–132
9. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-race and iterated F-race: An overview. In Bartz-Beielstein, T., et al., eds.: Experimental Methods for the Analysis of Optimization Algorithms. Springer (2010) To appear.
10. López-Ibáñez, M., Paquete, L., Stützle, T.: On the design of ACO for the biobjective quadratic assignment problem. In Dorigo, M., et al., eds.: ANTS 2004. Vol. 3172 of LNCS. Springer, Heidelberg (2004) 214–225
11. García-Martínez, C., Cordón, O., Herrera, F.: 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) (2007) 116–148
12. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In Giannakoglou, K., et al., eds.: EUROGEN 2001 (2002) 95–100
13. Mariano, C.E., Morales, E.: MOAQ: An Ant-Q algorithm for multiple objective optimization problems. In Banzhaf, W., et al., eds.: GECCO-1999. Vol. 1. Morgan Kaufmann Publishers, San Francisco, CA (1999) 894–901
14. Barán, B., Schaerer, M.: A multiobjective ant colony system for vehicle routing problem with time windows. In: Proceedings of the Twentyfirst IASTED International Conference on Applied Informatics, Insbruck, Austria (2003) 97–102
15. Doerner, K.F., Hartl, R.F., Reimann, M.: 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) (2003) 115–141
16. Stützle, T., Hoos, H.H.: $\mathcal{MAX}$-$\mathcal{MIN}$. Future Generation Computer Systems **16**(8) (2000) 889–914
17. López-Ibáñez, M., Paquete, L., Stützle, T.: Hybrid population-based algorithms for the bi-objective quadratic assignment problem. Journal of Mathematical Modelling and Algorithms **5**(1) (2006) 111–137
18. López-Ibáñez, M., Paquete, L., Stützle, T.: Exploratory analysis of stochastic local search algorithms in biobjective optimization. In Bartz-Beielstein, T., et al., eds.: Experimental Methods for the Analysis of Optimization Algorithms. Springer (2010) To appear.
19. Grunert da Fonseca, V., Fonseca, C.M., Hall, A.O.: Inferential performance assessment of stochastic optimisers and the attainment function. In Zitzler, E., et al., eds.: EMO 2001. Vol. 1993 of LNCS. Springer, Heidelberg (2001) 213–225