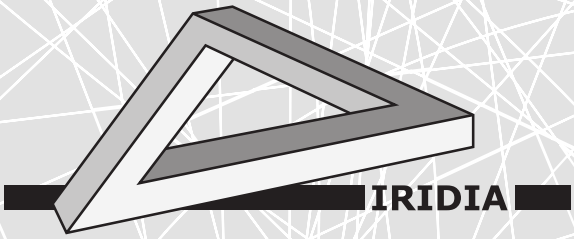


Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Automatically Improving the Anytime Behaviour of Optimisation Algorithms

Manuel López-Ibáñez and Thomas Stützle

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2012-012

May 2012

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/2012-012

Automatically Improving the Anytime Behaviour of Optimisation Algorithms

Manuel López-Ibáñez[a,*], Thomas Stützle[a]

[a]*IRIDIA, Université Libre de Bruxelles (ULB), CP 194/6, Av. F. Roosevelt 50 – B-1050 Brussels, Belgium*

Abstract

Optimisation algorithms with good anytime behaviour try to return as high-quality solutions as possible independently of the computation time allowed. Designing algorithms with good anytime behaviour is a difficult task, because performance is often evaluated subjectively, by plotting the trade-off curve between computation time and solution quality. Yet, the trade-off curve may be modelled also as a set of mutually nondominated, bi-objective points. Using this model, we propose to combine an automatic configuration tool and the hypervolume measure, which assigns a single quality measure to a nondominated set. Our goal is to improve the anytime behaviour of optimisation algorithms by means of automatically finding algorithmic configurations that produce the best nondominated sets. Moreover, the recently proposed weighted hypervolume measure is used here to incorporate the decision-maker's preferences into the automatic tuning procedure. We report on the improvements reached when applying the proposed method to two relevant scenarios: (*i*) the design of parameter variation strategies for MAX-MIN Ant System, and (*ii*) the tuning of the anytime behaviour of SCIP, an open-source mixed integer programming solver with more than 200 parameters.

Keywords: metaheuristics, anytime algorithms, automatic configuration, offline tuning

1. Introduction

Many optimisation algorithms generate a sequence of feasible solutions that are increasingly better approximations of the optimal solution. The main advantage of this design is that a run of the algorithm may be terminated at an arbitrary time or after a predefined termination criterion is reached. Upon termination, the algorithm returns the best solution found since the start of the run. For such an algorithm, the choice of the termination criterion may crucially determine its performance. If the parameter settings of an algorithm result in fast convergence to good solutions, the same parameter settings may prevent the algorithm from adequately exploring the search space to find better solutions if given ample time. On the other hand, algorithms designed with higher exploration capabilities may produce poor results if terminated too early.

*Corresponding author

Email addresses: manuel.lopez-ibanez@ulb.ac.be (Manuel López-Ibáñez), stuetzle@ulb.ac.be (Thomas Stützle)

An illustrative example is MAX-MIN Ant System (MMAS) [31]. MMAS is a high-performing ant colony optimisation (ACO) algorithm [13] that was designed to have a relatively long initial exploration phase with a subsequent transition to a strong exploitation phase. When rather long runs are performed, e.g., 1 000 or more algorithm iterations, MMAS is among the best ACO algorithms for travelling salesman problem (TSP) instances of up to a few thousand cities [31]. For shorter runs, however, other ACO algorithms, such as ant colony system [12], show a faster convergence and, hence, they outperform MMAS [13, 32].

ACO algorithms are just an example of metaheuristics, which often have several parameters that allow the user to control the trade-off between exploration of new solutions and exploitation of the best solutions found. However, a default parameter configuration is often used in practice, without taking into account differences in computational environment or termination criteria. This practice frequently leads to suboptimal results.

One way to address this issue is to not rely on default parameter settings, but instead to automatically configure the parameter settings for the particular termination criterion at hand. In the case of MMAS, it is well-known that different parameter settings significantly accelerate its convergence and, hence, its performance with shorter termination criteria. This approach, however, requires to reconfigure the algorithm for every possible termination criterion. Instead, it would be desirable to have an algorithm that produces good results independently of the termination criterion. Such algorithms are said to have good *anytime* behaviour [35].

Going back to the example of MMAS, recent results [27, 32] have shown that non-default settings and a dynamic variation of parameter settings may significantly improve its anytime behaviour. Finding these settings is an arduous task, since the original single-objective problem, where the goal is to obtain the best solution quality with a predefined termination criterion, is now transformed into a bi-criteria problem, where both solution quality and computation time are considered.

In this paper, we formulate the problem of finding parameter settings that improve the anytime behaviour of an algorithm as a set-valued bi-objective optimisation problem. In this formulation, the output of an algorithm run is not simply the quality of the best solution found, but a set of *(quality, time)* vectors, each of which represents an improvement of the best solution found at a particular time from the start of the algorithm. The optimisation problem becomes to find the configuration that produces the best possible set. In this paper, we propose to automatically configure algorithms for this bi-objective optimisation problem. The experimental results presented here indicate the large potential of this approach.

The outline of the paper is as follows. Section 2 reviews the concept of anytime optimisation and introduces our proposal for modelling the anytime behaviour as a bi-objective optimisation problem. In Section 3, we propose how to use this model to perform automatic algorithm configuration with the goal of improving anytime behaviour. In Section 4, we apply this proposal to the design of parameter adaptation strategies for MMAS. Section 5 discusses how our proposal enables a decision maker to incorporate preferences regarding the anytime behaviour of an algorithm to the automatic configuration procedure. A second case study is considered in Section 6, where we tune the anytime behaviour of SCIP, an open-source mixed integer programming solver with more than 200 parameters. Finally, Section 7 provides a summary of our results and discusses possible extensions of the present work.

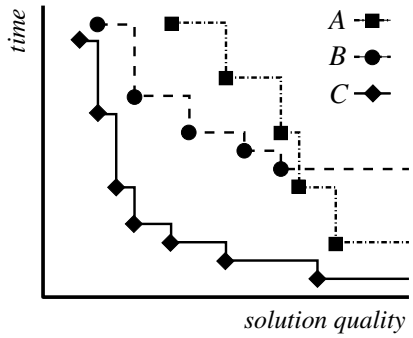


Figure 1: Example of the SQT curves of three algorithms A, B, and C.

2. Modelling the anytime behaviour of an algorithm as a bi-objective optimisation problem

Dean and Boddy [10] describe an anytime algorithm as one that, first, may be interrupted at any moment and return a solution and, second, it keeps steadily improving its solution until interrupted, eventually finding the optimal. Most metaheuristics and other optimisation algorithms satisfy this condition, and, hence, they are anytime optimisation algorithms. A concept of anytime behaviour more useful in the context of metaheuristics was introduced by Zilberstein [35], who highlights that *algorithms with good anytime behaviour* return as high-quality solutions as possible at any moment of their execution.

The anytime behaviour of a single run of an optimisation algorithm can be described by plotting the quality of the best solution achieved at each time step by the algorithm. These plots are called performance profiles [35] or solution-quality over time (SQT) curves [19], and they can be described as a set of points representing an improvement of solution quality at a particular time. Even until recently, the most common approach to compare optimisation algorithms in terms of anytime behaviour relies on visually inspecting such curves [26, 32]. Figure 1 gives an example of the SQT curves of three algorithms A, B and C on the same problem instance. These points can also be seen as solutions to a bi-objective problem, where both time and quality must be optimised.

In bi-objective optimisation problems, there are two conflicting objective functions that must be optimised at the same time. Without preference information about the importance of each objective, solutions are ordered according to Pareto-optimality. More formally, and assuming minimisation of all objectives without loss of generality, an objective vector $\vec{f}(s) = (f_1(s), f_2(s))$ *dominates* another objective vector $\vec{f}(s') = (f_1(s'), f_2(s'))$ iff $f_i(s) \leq f_i(s')$, $i = 1$ or 2, and $f_j(s) < f_j(s')$, $j \neq i$. When neither objective vector dominates the other, we say that they are *nondominated*. A set of mutually nondominated vectors is called a nondominated set. The goal of a multi-objective optimisation problem is to find the optimal nondominated set, that is, the set of solutions whose image in the objective space is not dominated by any other feasible solution.

As pointed out above, the anytime behaviour of a single-objective algorithm may be modelled as a bi-objective optimisation problem [8, 11], where the output of the algorithm is not the lowest cost found, but a nondominated set of pairs (*time*, *cost*). These pairs are obtained by recording the time and the solution quality whenever a new best-so-far solution is found during the run of the algorithm. Since these pairs are by definition mutually nondominated, the curve of solution quality over time is a nondominated set, and we can measure its quality

with respect to other nondominated sets produced by different algorithms. In particular, we say that a nondominated set is *better* than another one in the Pareto-optimality sense, if they are not equal and every solution of the latter is equal or dominated by at least one solution of the former. It is often the case, however, that neither nondominated set is better than the other, i.e., they are *incomparable*. In the example shown in Fig. 1, sets A and B are incomparable, since some points of A are not dominated by B and viceversa. In such cases, it is typical to measure the quality of nondominated sets with respect to a quality indicator. Unary quality indicators assign a value to each nondominated set, while binary indicators assign a value to a pair of nondominated sets. Unary quality indicators are less powerful than binary ones, but their results are much easier to analyse and interpret, since they transform a multi-objective problem into a single-objective one.

The hypervolume measures stands out as the only unary quality indicator that is always able to discriminate when a nondominated set is better than another in the Pareto-optimal sense [36]. In the bi-objective case, the hypervolume computes the area of the objective space that is dominated by a nondominated set and bounded by a reference point. This reference point should be dominated by all points in all possible nondominated sets. By definition, a nondominated set that is better in terms of Pareto-optimality than another will have a larger area. Moreover, when two nondominated sets are incomparable, the one that dominates a larger area of the objective space will be considered better by the hypervolume measure. Finally, another benefit of the hypervolume, which we will explore in this paper, is the possibility of defining preferences by means of the weighted hypervolume [3, 37].

3. Automatic configuration of anytime algorithms

Our proposal is to tackle the automatic configuration of anytime behaviour by integrating the unary hypervolume measure into an automatic configuration tool.

Automatic configuration [5], also known as offline tuning, aims at simplifying the task of properly configuring optimisation algorithms. Given a description of the parameter space of an algorithm, a set of training instances representative of the problem, and a tuning budget (e.g., a maximum number of runs of the algorithm), the automatic configuration problem is to find the best configuration of the algorithm for unseen instances of the same problem.

A notable example of an offline automatic configuration tool is I/F-Race [6, 25], which is based on the F-race procedure for selection of the best configuration over a sequence of training instances [5]. I/F-Race iteratively updates a probabilistic model of the parameter space that is used to sample new parameter configurations in subsequent iterations. I/F-Race is able to handle a large number of numerical and categorical parameters, and parameters subordinate to the settings of other parameters. These features make it ideal for the case studies tackled in this paper.

Existing automatic configuration tools, including I/F-Race, were designed for single-objective algorithms [4, 6, 20]. By means of unary quality measures, these tools have recently been applied to the configuration of multi-objective algorithms. Wessing et al. [33] automatically tune the variation operator of a multi-objective evolutionary algorithm applied to a single problem instance. Simultaneously, López-Ibáñez and Stützle [22, 24] automatically instantiate new designs of multi-objective ant colony optimisation (MOACO) algorithms for the bi-objective travelling salesman problem from a framework of MOACO algorithmic components. More recently, Dubois-Lacoste et al. [15] applied this latter approach to outperform the state of the art in several bi-objective permutation flow-shop problems. We follow in

this paper the same approach used in these two latter works, that is, the integration of the hypervolume measure into the iterated F-race (I/F-Race) automatic configuration tool. This approach allows us to apply automatic configuration tools to improve the anytime behaviour of an algorithm.

The automatic configuration of the anytime behaviour of an algorithm should not be confused with parameter tuning as a multi-objective problem [14], where the aim is to produce a set of parameter configurations that are mutually nondominated with respect to multiple criteria. By contrast, in this paper, our aim is to produce a single parameter configuration that generates an anytime behaviour that is as good as possible.

There have been some recent attempts at tackling this problem. The proposal closest to ours is by den Besten [11], who combined a racing procedure for selection of the best and a performance measure based on the binary ϵ-indicator. The use of a binary measure involves computing a matrix of ϵ-measure values, comparing each alternative with the rest, and transforming it into ranks. More recently, Branke and Elomari [7] combined a meta-level evolutionary algorithm and an ad-hoc ranking procedure for tuning the mutation rate of a lower-level algorithm for multiple termination criteria in a single tuner run. Their ranking method is not based on a multi-objective quality measure. Instead, it ranks each configuration with respect to the number of discrete time steps in which the configuration was better than other configurations. The proposal was tested by tuning a single parameter of one algorithm on a single problem instance.

In the next sections, we evaluate two applications of the proposed technique. The first case study is the design of parameter variation strategies to improve the anytime behaviour of MMAS in the TSP. This case study arises from our own practical experience, and it illustrates how automatic configuration may save a significant amount of human effort, while leading to improved results. The second case study evaluates the improvement in anytime behaviour that may be achieved when automatically configuring an off-the-shelf optimisation software with a very large number of parameters, namely the open-source mixed integer programming solver SCIP [1]. In this case, we attempt to configure 207 parameters of SCIP, which were suggested by the SCIP developers as being the most relevant. This number of parameters is three times larger than the largest attempt we are aware of, i.e., the automatic configuration of CPLEX [20].

4. Case Study: Design of parameter variation strategies for MAX-MIN Ant System on the TSP

Parameter adaptation [2, 16], that is, the variation of parameter settings while solving a problem instance, enables metaheuristics to adapt the parameters to different phases of the search, and to balance exploration of the search space and exploitation of the best solutions found. Hence, parameter adaptation may make algorithms more robust to different termination criteria. Designing and comparing parameter adaptation schemes is, however, an arduous and complex task. Here, we show that our proposed method for the automatic configuration of the anytime behaviour can help to implement parameter adaptation strategies.

In previous work [27, 32], we studied parameter adaptation strategies for ant colony optimisation (ACO) algorithms. In particular, we studied the anytime behaviour of MAX-MIN Ant System (MMAS) by experimenting with various static parameter settings and parameter

variation strategies. The experiments carried out in these preliminary works involved testing various settings that were deemed interesting, and visually comparing the resulting SQT curves. Needless to say, this was a human intensive task that required many iterations of experimentation and analysis. We roughly estimate that the overall effort for obtaining the best configurations was close to one person-year. The result of this effort is that the best configurations are significantly better in terms of anytime behaviour than the default settings of MMAS [27, 32]. In the present paper, we use this scenario as a case study for the automatic configuration of anytime behaviour. First, we briefly introduce MMAS. Then, we present several parameter variation strategies. Finally, we compare the configurations obtained after automatic configuration with the ones obtained by manual configuration.

4.1. MAX-MIN Ant System

MMAS is an ACO algorithm that incorporates an aggressive pheromone update procedure and mechanisms to avoid search stagnation. When applying MMAS to the TSP, each ant starts at a randomly chosen initial city, and constructs a tour by randomly choosing at each step the city to visit next according to a probability defined by pheromone trails and heuristic information. In particular, the probability that ant k chooses a successor city j when being at city i is given by

$$p_{ij} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{h \in N^k} [\tau_{ih}]^{\alpha} \cdot [\eta_{ih}]^{\beta}} & \text{if } j \in N^k \\ \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where τ_{ij} is the pheromone trail strength associated to edge (i, j), η_{ij} is the corresponding heuristic information; α and β are two parameters that influence the weight given to pheromone and heuristic information, respectively; N^k is the feasible neighbourhood, that is, a candidate list of cities not yet visited in the partial tour of ant k.

Departing from the original MMAS but following previous work [28, 30], we also incorporate the *pseudo-random action choice rule* of ACS [12], which allows for a greedier solution construction. With a probability q_0 an ant chooses next a city $j \in N^k$ such that

$$j = \arg\max_{h \in N^k} \{[\tau_{ih}]^{\alpha} \cdot [\eta_{ih}]^{\beta}\}; \tag{2}$$

otherwise, the ant performs the probabilistic selection based on Eq. 1. A value of $q_0 = 0$ reverts back to the original MMAS.

The pheromone update of MMAS updates all pheromone trails as

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

where ρ, $0 < \rho \leq 1$, is a parameter called evaporation rate and

$$\Delta\tau_{ij}^{\text{best}} = \begin{cases} 1/f(s^{\text{best}}) & \text{if edge } (i,j) \in s^{\text{best}}, \\ \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $f(s)$ is the tour length of solution s, and s^{best} is either the iteration-best solution, the best-so-far solution or the best solution since a re-initialisation of the pheromone trails (restart-best). In MMAS, these solutions are chosen alternately [31].

Finally, solutions constructed by the ants may be further improved by the application of a local search algorithm. In this paper, we will use MMAS with 2-opt local search, as was done in previous work [27, 32].

Table 1: Default settings of the parameters under study for MMAS with 2-opt local search.

Algorithm	Time_{CPU}	α	β	ρ	m	q_0
MMAS	500 s	1.0	2.0	0.2	25	0.0

4.2. Parameter variation strategies in MMAS

As a first case study, we apply our proposal to automatically configure parameter variation strategies in order to improve the anytime behaviour of MMAS.

Following our previous work [32], we focus on two basic schemes for parameter variation in MMAS, which we call henceforth *delta* and *switch* strategies. The *delta* strategy applied, for example, to parameter β increases the value of β at each iteration of the algorithm by a certain amount $\Delta\beta$, starting from β_{start} and stopping at β_{end}. If $\beta_{\text{start}} > \beta_{\text{end}}$, then the parameter value is decreased by $\Delta\beta$ instead of increased. Conversely, the *switch* strategy switches the value of parameter β from β_{start} to β_{end} at iteration β_{switch}. An additional parameter β_{var} controls the variation strategy, which is either *delta*, *switch* or *none*, where *none* means that the parameter value stays constant throughout the run of the algorithm. As a result, we add to MMAS five additional parameters: β_{start}, β_{end}, $\Delta\beta$, β_{switch} and β_{var}.

We apply these parameter variation strategies to parameters β, ρ, the number of ants (m), and q_0. Hence, we add five additional parameters for each parameter that is dynamically varied. Other parameters are set to default values [31], in particular, $\alpha = 1$. Table 2 describes the domains used in the tuning of these new parameters.

The automatic configuration tool is the implementation of I/F-Race provided by the `irace` software package [25]. As explained above, we incorporate the hypervolume measure to the `irace` software in order to evaluate the anytime behaviour of a single run of MMAS. We use a publicly available implementation of the hypervolume measure [17], and the MMAS implementation provided by the ACOTSP software [29].

We consider MMAS with 2-opt local search on instances of 3 000 cities. We divide each set of instances into a training (tuning) set and a test set of 50 instances each. In a first step, we tune separately the variation strategy of one parameter at a time. That is, we perform one run of I/F-Race for each parameter $\{m, \beta, q_0, \rho\}$. In each run, the variation strategy is fixed to *none* for all parameters except one, and, hence, there are six parameters to tune (see Table 2). We give each run of I/F-Race a budget of 1 000 runs of MMAS. Each run of MMAS is stopped after Time_{CPU} seconds (Table 1). In a second step, we automatically configure all parameter variation strategies at the same time, that is, we configure 24 parameters instead of six. Since the parameter space is much larger now, we assign a larger tuning budget to this run of I/F-Race, specifically 10 000 runs of MMAS.

After each run of I/F-Race finishes, we apply the resulting parameter configurations (Table 4) to the test instances. In addition, we also run on the test instances the *default* parameter configuration of MMAS (Table 1) without any variation strategy and several variation strategies previously found by manual ad-hoc experimentation (Table 3) [32]. We present these results in the following sections.

4.3. Analysis of the results

4.3.1. Automatic configuration vs. manual configuration

Our goal is to improve the anytime behaviour of MMAS over the whole set of test instances. Hence, we analyse the overall results by plotting the average solution quality over time for

Table 2: Parameter space for variation strategies of MMAS.

Parameter	Domain	Constraint
m_{var}, β_{var}, ρ_{var}, $q_{0\text{var}}$	{ *delta*, *switch*, *none* }	
m	[1, 100]	if var = *none*
β	[0, 20]	
ρ	[0.01, 1.0]	
q_0	[0.0, 1.0]	
Δm	$\{0.01, 0.05, 0.1, 0.25, 0.5, 1, 2, 5\}$	if var = *delta*
$\Delta\beta$	$\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0\}$	
$\Delta\rho$	$\{0.001, 0.002, 0.005, 0.01\}$	
Δq_0	$\{0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005\}$	
m_{switch}, β_{switch}, ρ_{switch}, $q_{0\text{switch}}$	[1, 500]	if var = *switch*
m_{start}	1	if var $\in \{$*delta*, *switch*$\}$
m_{end}	[1, 500]	
β_{start}	[0, 20]	
β_{end}	[0, 5]	
ρ_{start}	[0.001, 1.0]	
ρ_{end}	[0.001, 1.0]	
$q_{0\text{start}}$	[0.0, 1.0]	
$q_{0\text{end}}$	[0.0, 1.0]	

Table 3: Parameter configurations found by a human expert when varying one parameter at a time in MMAS [32].

Configuration	Parameter settings
manual var ants	$m_{\text{var}} =$ *delta*, $m_{\text{start}} = 1$, $m_{\text{end}} = 25$, $\Delta m = 0.1$
manual var beta	$\beta_{\text{var}} =$ *switch*, $\beta_{\text{start}} = 20$, $\beta_{\text{end}} = 3$, $\beta_{\text{switch}} = 50$
manual var rho	$\rho_{\text{var}} =$ *none*, $\rho = 0.9$
manual var q0	$q_{0\text{var}} =$ *delta*, $q_{0\text{start}} = 0.99$, $q_{0\text{end}} = 0$, $\Delta q_0 = 0.0005$

Table 4: Parameter configurations found by I/F-Race for MMAS.

Configuration	**Parameter settings**
auto var ants	$m_{\text{var}} = \textit{delta}$, $m_{\text{delta}} = 0.05$, $m_{\text{start}} = 1$, $m_{\text{end}} = 417$
auto var beta	$\beta_{\text{var}} = \textit{delta}$, $\beta_{\text{delta}} = 0.05$, $\beta_{\text{start}} = 9$, $\beta_{\text{end}} = 4$
auto var q0	$q_{0\text{var}} = \textit{switch}$, $q_{0\text{switch}} = 200$, $q_{0\text{start}} = 0.96$, $q_{0\text{end}} = 0.30$
auto var rho	$\rho_{\text{var}} = \textit{delta}$, $\rho_{\text{delta}} = 0.001$, $\rho_{\text{start}} = 0.82$, $\rho_{\text{end}} = 0.84$
auto var ALL	$m_{\text{var}} = \textit{delta}$, $m_{\text{delta}} = 1$, $m_{\text{start}} = 1$, $m_{\text{end}} = 384$, $\beta_{\text{var}} = \textit{switch}$, $\beta_{\text{switch}} = 79$, $\beta_{\text{start}} = 5$, $\beta_{\text{end}} = 0$, $q_{0\text{var}} = \textit{delta}$, $q_{0\text{delta}} = 0.002$, $q_{0\text{start}} = 0.87$, $q_{0\text{end}} = 0.57$, $\rho_{\text{var}} = \textit{none}$, $\rho = 0.68$
auto fix	$\beta = 5.9$, $\rho = 0.62$, $m = 84$, $q_0 = 0.099$
auto var no anytime	$m_{\text{var}} = \textit{switch}$, $m_{\text{switch}} = 50$, $m_{\text{start}} = 1$, $m_{\text{end}} = 317$, $\beta_{\text{var}} = \textit{delta}$, $\beta_{\text{delta}} = 0.5$, $\beta_{\text{start}} = 8$, $\beta_{\text{end}} = 2$, $q_{0\text{var}} = \textit{switch}$, $q_{0\text{switch}} = 139$, $q_{0\text{start}} = 0.6241$, $q_{0\text{end}} = 0.2725$, $\rho_{\text{var}} = \textit{switch}$, $\rho_{\text{switch}} = 493$, $\rho_{\text{start}} = 0.338$, $\rho_{\text{end}} = 0.7495$

all test instances at once. More concretely, for each algorithm configuration, we have the best solution quality found f_{rit} on run r on instance i at time t. We compute the relative percentage deviation (RPD) from the optimal solution for each instance as $RPD_{rit} = 100 \cdot f_{rit}/f_i^{\text{opt}}$, where f_i^{opt} is the optimal tour length of instance i. Then, we compute the mean RPD over all 50 instances and over all 15 independent runs of each algorithm as $RPD_t = \frac{1}{50 \cdot 15} \cdot \sum_{i=1}^{50} \sum_{r=1}^{15} RPD_{rit}$. Each line in the plots in Fig. 2 corresponds to the RPD_t of one algorithm configuration. Finally, we compute the hypervolume of each run on each instance by normalising both time and solution quality to the interval $[1, 2]$ and using $(2.1, 2.1)$ as the reference point. Then, for each algorithm configuration we compute its mean hypervolume over all its runs on all test instances, and we give this value in the legend of each plot.

The first important observation is how the value of the hypervolume matches the quality of the anytime behaviour of the different configurations: A larger hypervolume value indicates a better anytime behaviour. The first four plots (a,b,c,d) in Fig. 2 show a large improvement in the anytime behaviour of the manually tuned configurations with respect to the default configuration of MMAS. Nonetheless, the automatically found configurations are able to match, and in most cases surpass the manually tuned configurations in terms of hypervolume, despite the fact that the manually tuned configurations were found by extensive experimentation under the guidance of human expertise.

Fig. 3 compares the configuration obtained after automatically configuring all parameter variation strategies at once versus the best configurations obtained after automatically configuring the variation strategy of one parameter at a time. In our previous study, the manual tuning and analysis of all parameter strategies at once was ruled out as infeasible, given the extremely large number of potential configurations and interactions among different parameters. Here, we see that automatically configuring all parameters at once leads to an additional improvement in anytime behaviour.

The above plots show the mean hypervolume over all runs and all test instances. To assess whether the observed differences are statistically significant, we perform a statistical analysis of the results over the whole set of test instances. The analysis is based on the Friedman test for analysing non-parametric unreplicated complete block designs, and its associated post-

Table 5: Various configurations of MMAS ordered according to the sum of ranks with respect to the hypervolume obtained over all test instances. The numbers in parenthesis are the differences of ranks relative to the best ranked configuration. All configurations are statistically significantly worse than the best one (auto var ALL).

ΔR_α	**Configurations (ΔR)**
9.87	**auto var ALL (0)**, auto var q0 (34), manual q0 (94), auto var rho (125), manual rho (189), auto var ants (236), auto var beta (294), manual ants (345), manual beta (380), default (433)

test for multiple comparisons [9]. First, we calculate the mean hypervolume of the 15 runs of each algorithm for each instance. Then, we perform a Friedman test using the instances as the blocking factor, and the different configurations of MMAS as the treatment factor. The null hypothesis is that the configurations have identical effect on the ranking according to the hypervolume within each instance. If the Friedman test rejects the null hypothesis given a significance level of $\alpha = 0.05$, we proceed to calculate the minimum difference between the sum of ranks of two configurations that is statistically significant (ΔR_α). In this manner, we identify which configurations are significantly different from the best ranked one, i.e., the one with the lowest sum of ranks.

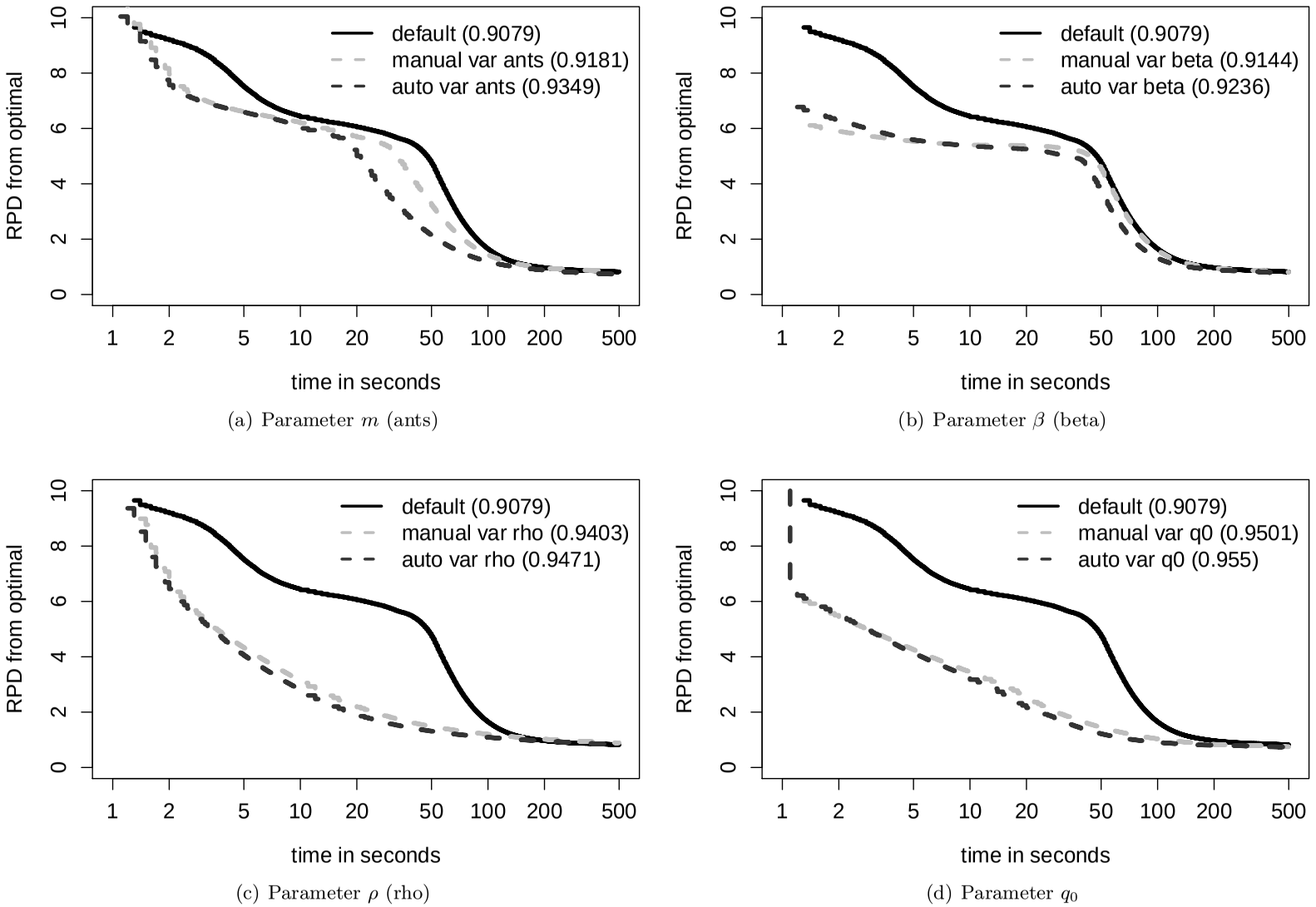
Table 5 summarises the results of the statistical analysis. It shows the value of ΔR_α for $\alpha = 0.05$, the different configurations of MMAS sorted by increasing sum of ranks, and the difference between the sum of ranks of each configuration and the best configuration (ΔR).

For each parameter considered, the ranking shown in Table 5 always ranks higher the configurations found automatically than their counterparts found by ad-hoc experimentation (auto vs. manual, respectively). More importantly, it shows that the best ranked configuration is the one that automatically configured all parameters at once, and that the difference in ranks between this configuration and the rest is statistically significant.

4.3.2. Hypervolume vs. final quality

Here, we show that the use of the hypervolume as the tuning criterion is the key factor for improving the anytime behaviour. Fig. 4 shows four configurations of MMAS: the default configuration (default); the one resulting from automatically tuning all variation parameters (auto var ALL); a configuration obtained by tuning all variation parameters with respect to final quality, that is, the solution quality obtained at 500 seconds (auto var no anytime); and a configuration obtained by tuning the classical MMAS parameters, without any variation, with respect to final quality (auto fix). The plot clearly shows that, independently of whether MMAS uses parameter variation or not, the results not tuned with respect to the hypervolume have worse anytime behaviour.

A possible concern of tuning for anytime behaviour is a significant loss of final quality. Hence, we examine the final quality achieved by these four variants of MMAS in Fig. 5. According to the boxplots, there is an important improvement in the final quality achieved in comparison with the default configuration of MMAS, even for the configuration tuned for anytime behaviour. The boxplot does not show a large difference between the three automatically configured variants. Nonetheless, the Friedman test indicates that the final quality obtained by the variant tuned for anytime behaviour is statistically worse than the variants tuned for final quality (Table 6). In order to bound the amount of loss, we perform a



(a) Parameter m (ants)

(b) Parameter β (beta)

(c) Parameter ρ (rho)

(d) Parameter q_0

Figure 2: Anytime behaviour of manually tuned vs. automatically configured variants of MMAS.

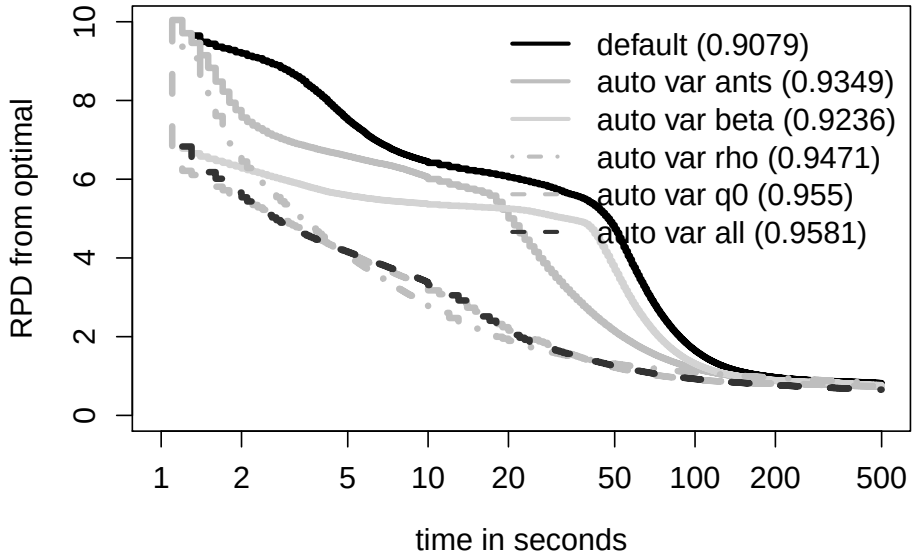


Figure 3: Anytime behaviour of automatically configured variants of MMAS vs. the default configuration.

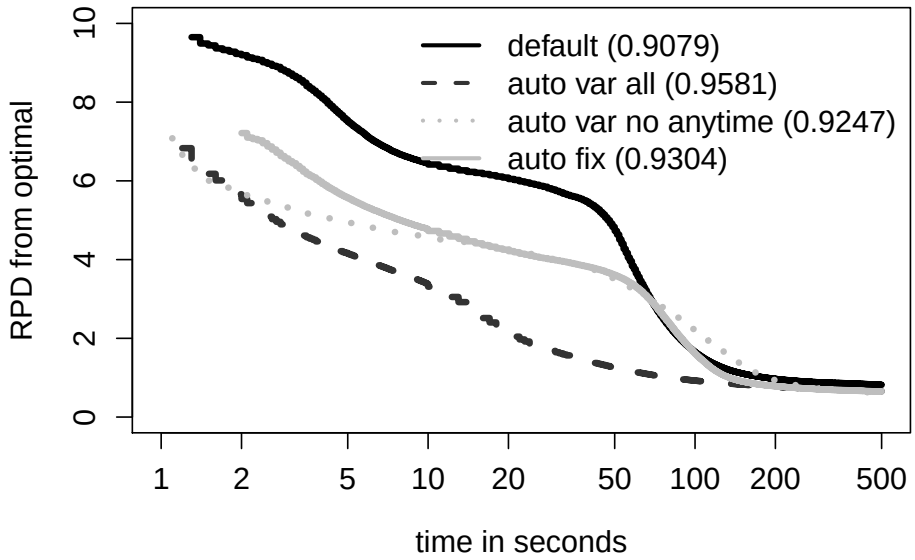


Figure 4: Anytime behaviour of automatically configured variants of MMAS vs. the default configuration.

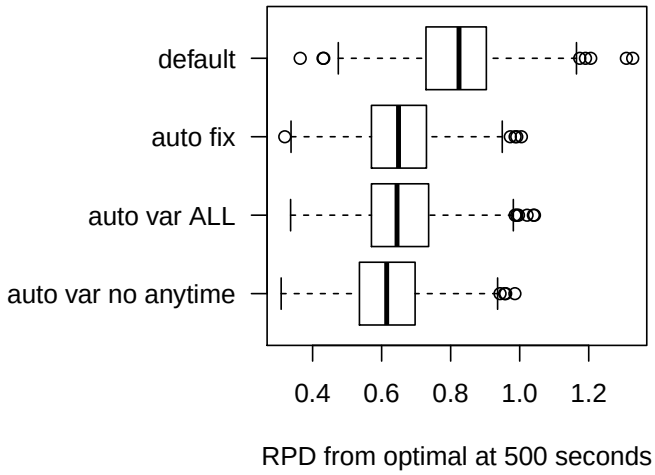


Figure 5: Final quality achieved by several variants of MMAS.

Table 6: Configurations of MMAS ordered according to the sum of ranks with respect to the final solution quality obtained. The numbers in parenthesis are the difference of ranks relative to the best configuration. Configurations that are not significantly different from the best one are indicated in bold face.

ΔR_α	**Configurations (ΔR)**
13.68	**auto var no anytime (0)**, auto var ALL (48), auto fix (52), default (132)

Student's t-test and compute the associated 95% confidence interval on the mean difference in final quality between the configuration tuned for anytime behaviour and the best ranked configuration, which is $[0.0244, 0.0480]$ measured in RPD. Although the loss in final quality when tuning for anytime behaviour is small in this case, an anytime algorithm should aim to match the best possible final quality in the ideal case.

5. Articulation of preferences in automatic configuration of anytime algorithms

The use of the hypervolume for automatic tuning of anytime algorithms has an additional advantage compared to other unary measures, that is, the possibility of specifying the decision-maker's preferences. A recent proposal extends the hypervolume indicator by a weight function over the objective space [3, 37]. A weight function that assigns a higher value to a certain region of the objective space will bias the hypervolume indicator to favour nondominated sets that dominate that region. We show here that this formulation can straightforwardly be used to introduce a bias in the anytime behaviour produced by automatic configuration.

As an example, let us assume that the decision maker's preference is to obtain as good final solution quality as possible, while still giving some minor importance to achieving a good anytime behaviour. In other words, the decision maker prefers configurations that generate solution-quality curves that are better towards minimising the solution quality (in our case, the second objective). Zitzler et al. [37] suggest to model this preference by considering the following weight function (adapted here to minimisation):

$$w^{\text{qual}}(z) = e^{20\cdot(1-z_2)}/e^{20} \tag{5}$$

where $z = (z_1, z_2) \in Z$ is an objective vector, with z_1 representing time and z_2 representing solution quality, and $Z = [0, 1] \times [0, 1]$ represents the normalised bi-objective space of time×quality.

The weighted hypervolume is computed as the integral of the weight function over the region dominated by a set of nondominated points and bounded above by a reference point. To give a rough idea of this integral when using the weighted function w^{qual}, Figure 6 (left) shows the value of the weighted hypervolume for each individual vector in the normalised objective space Z and with reference point $(1, 1)$. The plot shows that vectors with very small values of z_2 are assigned a high hypervolume, but vectors with values of z_2 higher than 0.2 are assigned a hypervolume close to zero. By comparison, Figure 6 (left) shows the value of the non-weighted hypervolume, which is symmetric around the diagonal, that is, without a preference for either objective.

We can assign a positive hypervolume value to more regions of the objective space by weighting also the z_1 component (corresponding to time), but then we have to increase the exponent associated to z_2 in order to keep a strong preference for low solution quality. This is done with the following weight function:

$$w^{\text{xqual}} = e^{10 \cdot z_1}/e^{10} + e^{100 \cdot (1 - z_2)}/e^{100} \tag{6}$$

The weighted hypervolume using this weight function for each individual vector in the objective space Z is shown in Fig. 6 (right). In this case, more regions of the objective space have a positive hypervolume. Yet, the value of the hypervolume increases exponentially in the direction of decreasing z_2 (solution quality), while it stays roughly constant along z_1 (except for very high values of z_1).

We illustrate the differences between the original hypervolume and the two weighted variants above with an example. Figure 7 shows five SQT curves in the normalised objective space Z. The plot shows the region $z_2 \in [0.0, 0.15]$, where we can see that the curves are ordered according to the final quality achieved, with curve a being the best and curve e being the worst. The legend provides three numbers for each curve, which correspond to evaluating the curve with the classical hypervolume, the hypervolume weighted by w^{qual} and the hypervolume weighted by w^{xqual}, respectively. Table 7 gives the curves in increasing order of preference according to each measure.

In this example, the classical hypervolume ranks curve a, which is the curve with the best final quality, worse than other three curves. The weighted hypervolume functions increase the preference for curve a, and our proposed variant w^{xqual} gives it the highest rank.

Next, we test the effect of these two weighted hypervolume functions on the automatic configuration procedure. In particular, we carry out additional runs of I/F-Race using the weighted hypervolume variants described above, i.e., w^{qual} (Eq. 5) and w^{xqual} (Eq. 6). I/F-Race is run with the same setup as for tuning all parameter variations at once in Section 4.2, in particular, with a budget of 10 000 runs of ACOTSP. These additional tuning runs produce two new configurations of MMAS, which we ran 25 times with different random seed on each test instance.

Figure 8 plots the mean RPD over all runs of the resulting four configurations of MMAS: the default configuration (default); the one resulting from automatically tuning all variation parameters at once using the classical hypervolume (auto var ALL); the configuration obtained with the same tuning setup but using the weighted hypervolume with w^{qual} (whv qual); and the configuration obtained using the weighted hypervolume with w^{xqual} (whv xqual). In addition,

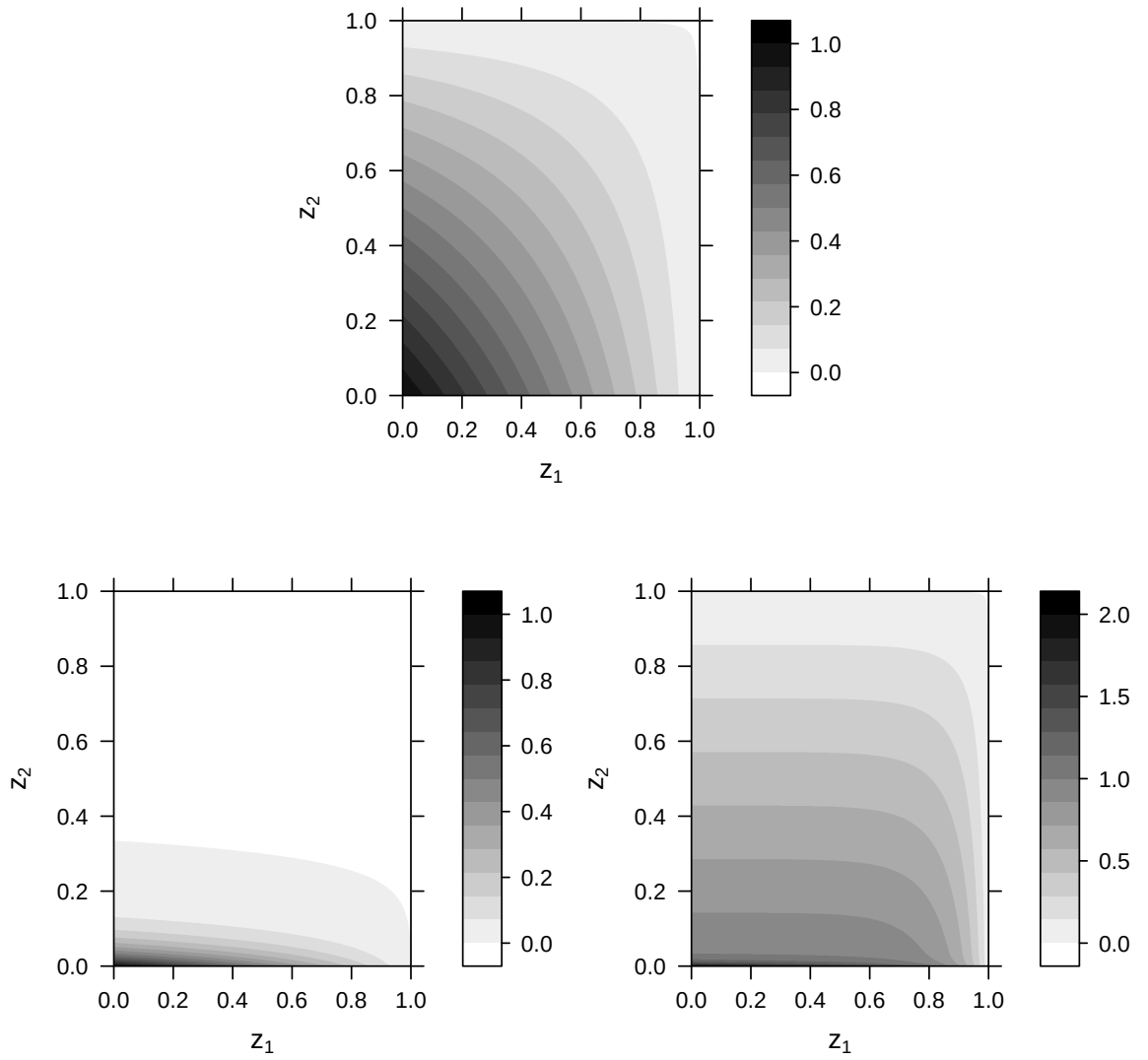


Figure 6: Value of the classical hypervolume (top) and the weighted hypervolume when using weight functions w^{qual} (left) and w^{xqual} (right) on the normalised objective space. The grey level at each point gives the (weighted) hypervolume of that individual point, which is computed as the integral of the weight function over the region dominated by the point and bounded above by the reference point $(1, 1)$.

Table 7: Ranking of the SQT curves in Fig. 7 according to various preferences

Preference	Ranking (best to worst)
final quality	a b c d e
hypervolume	c b d a e
w^{qual}	c b a d e
w^{xqual}	a c b d e

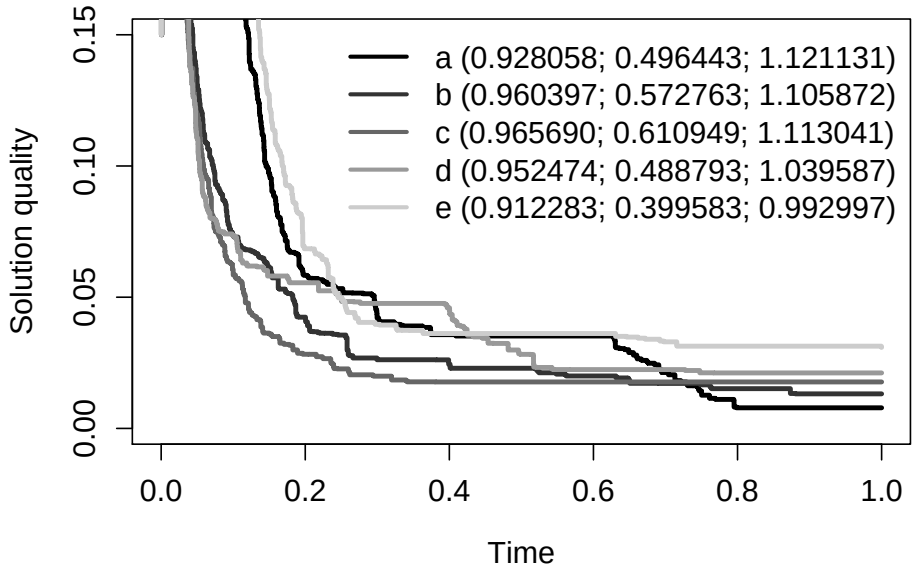


Figure 7: For each SQT curve, the legend shows the classical hypervolume, and the weighted hypervolume variants w^{qual} and w^{xqual}.

the legend provides three numbers for each curve, which correspond to evaluating the results with the classical hypervolume, the hypervolume weighted by w^{qual} and the hypervolume weighted by w^{xqual}, each of them averaged over all runs.

In terms of final quality, the two configurations tuned with the weight functions (w^{qual} and w^{xqual}) are slightly better than the one tuned with the classical hypervolume, as indicated by the boxplots given in Fig. 9. Moreover, the configurations tuned with the weight functions obtain the lowest final quality in most instances. In fact, according to the Friedman test, these configurations are significantly better than configurations obtained by tuning for the classical hypervolume and for final quality (Table 8). The main conclusion of these experiments is that the weighted hypervolume allows us to set preferences on the trade-off between quality and time. For example, the weighted function w^{xqual} imposes a strong preference for good final quality.

6. Case Study: Automatic configuration of an anytime MIP solver

6.1. Experimental setup

In this second scenario, we apply our proposed approach to a very different problem with a large number of parameters. In particular, we tune 207 parameters of SCIP [1], a mixed

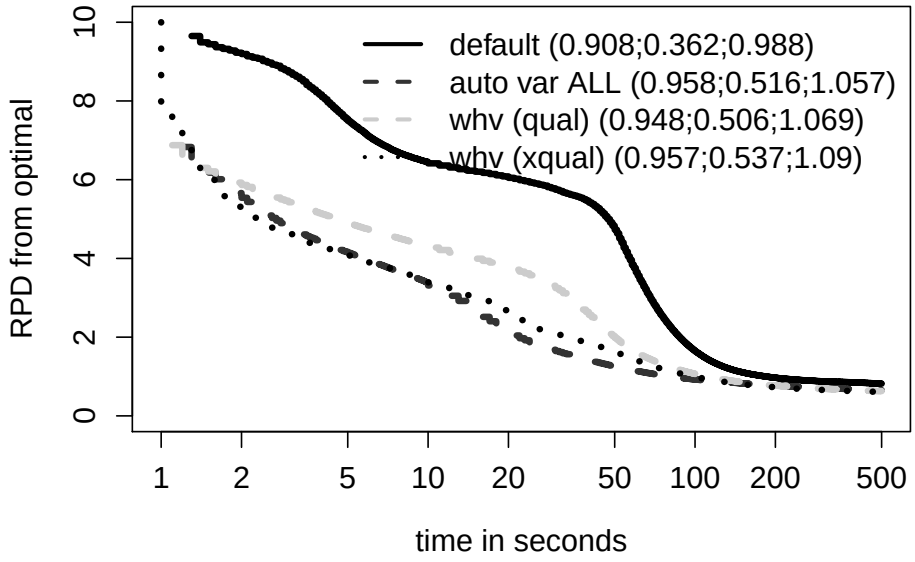


Figure 8: Automatically tuned variants of MMAS vs. the default configuration.

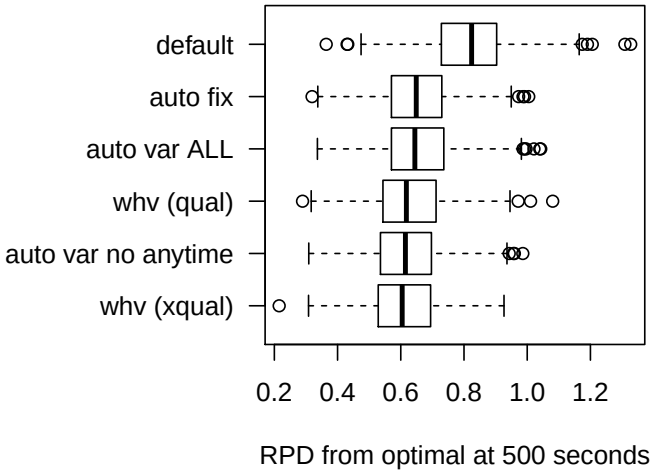


Figure 9: Final quality achieved by several variants of MMAS.

Table 8: Configurations of MMAS ordered according to the sum of ranks with respect to the final solution quality obtained. The numbers in parenthesis are the difference of ranks relative to the best configuration. Configurations that are not significantly different from the best one are indicated in bold face.

ΔR_α	**Configurations (ΔR)**
23.16	**whv (xqual) (0), auto var no anytime (11)**, whv (qual) (47), auto var ALL (94), auto fix (99), default (199)

integer programming (MIP) solver. This number of parameters is three times larger than the configuration scenario that included the largest number of parameters that we are aware of [20]. The number of parameters is too large to be detailed here, but details can be found in the supplementary page [23].

The benchmark set is composed of 2 000 MIP-encoded instances (200 goods, 1000 bids) of the NP-hard winner determination problem for combinatorial auctions [21] previously used by Hutter et al. [20]. The benchmark set is split in two equal sets for training and testing. In a combinatorial auction, bids are placed for subsets of goods. The goal in the winner determination problem is to find an assignment of goods to bids that maximises the total value of the winning bids.

For our experiments here, we use SCIP version 2.0.2 linked with the linear programming solver SoPlex 1.5.0. We set the maximum memory limit of SCIP to 350 MB. During our experiments, we discovered that some parameter configurations produced an incorrect behaviour of SCIP, and we assign those configurations the worst possible hypervolume. We give SCIP a time limit of 300 seconds, and we allow 5 000 runs of SCIP for each run of `irace`. We carry out the tuning as before, that is, we combine `irace` with the hypervolume measure in order to improve the anytime behaviour of SCIP. We seed the automatic configuration procedure with the default configuration of SCIP.

For the purposes of comparison, we perform two additional tuning runs with two different objectives: (1) minimising the run time to find the optimal solution, and (2) maximising the final objective value obtained after 300 seconds. This way we obtained two additional configurations of SCIP, which we label as auto time and auto quality, respectively. We use these configurations to asses the potential loss of either run time or final solution quality, when tuning for improving the anytime behaviour.

Finally, we run all configurations of SCIP obtained from the various tuning setups plus the default configuration one time on each instance from the test set.

6.2. Analysis of SCIP configurations

As a first step in our analysis, we graphically examine the solution quality over time. For each configuration, we compute the mean RPD over the 1 000 test instances at each time step. Next, we plot the mean RPD over time in Fig. 10. We also give in the legend the mean hypervolume value corresponding to each configuration of SCIP. The plot uses a logarithmic scale for the x-axis (time), since the largest differences appear on the first half of the computation time limit.

The plot shows that the configuration tuned with the hypervolume (auto anytime) obtains a better anytime behaviour (and a higher hypervolume) than the rest. Moreover, both the configuration tuned for final quality and the one tuned for solving time show worse anytime

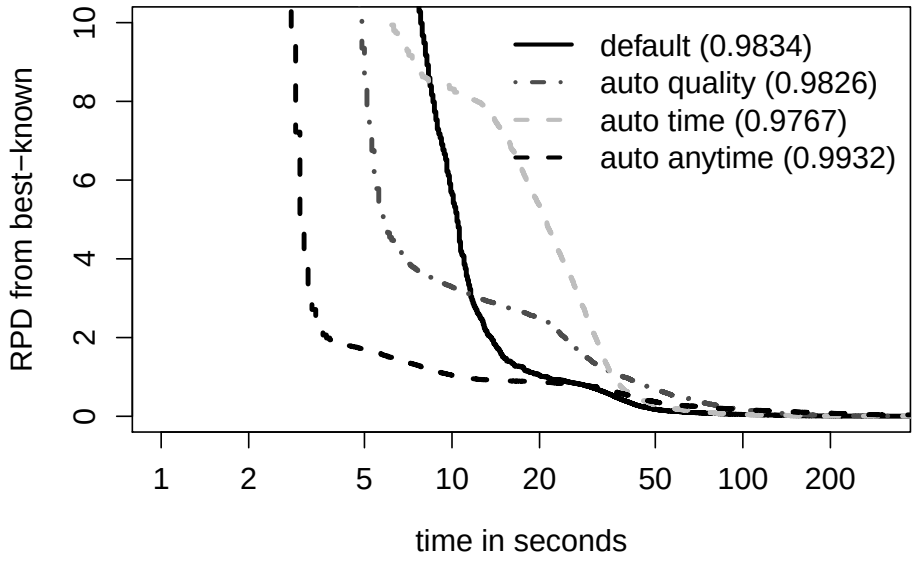


Figure 10: Mean RPD over all test instances for different configurations of SCIP. The number in parentheses is the mean hypervolume corresponding to that configuration.

behaviour (and lower hypervolume) than the default configuration of SCIP. The differences observed in the hypervolume values (and, hence, in the anytime behaviour) of each SCIP configuration are more evident in Fig 11(a), which shows that the hypervolume values corresponding to auto anytime are much lower than those corresponding to the other configurations of SCIP.

Improving the anytime behaviour does not necessarily mean that instances are solved faster to optimality. Fig. 11(b) shows the time required by each configuration to solve each of the 1 000 test instances. The best configurations of SCIP according to this criterion are the default configuration and the configuration tuned specifically for this criterion (auto time). This result is not surprising, since this is the most popular evaluation criterion in mixed-integer programming, and, hence, we presume that SCIP is by default tuned for it.

We also examine the potential loss of final quality. Fig. 11(c) shows the RPD from the optimal at the cut-off time of 300 seconds. All configurations solve most of the instances to optimality (or very close to it). However, the configuration tuned for anytime (auto anytime) is the one that diverges most often from near-optimality. Hence, there is some loss of final quality when tuning using the hypervolume.

If we look at the solution quality up to a different cut-off time, the situation is certainly different. For example, if we consider solution quality up to 10 seconds (Fig. 11(d)), there is a large difference between the configurations. While the auto anytime configuration obtains an RPD value much lower than 10% in most cases, the RPD values of the default configuration are frequently larger than 10%.

The observations above are further confirmed by statistical analysis. We carry out four independent Friedman tests (as described in Sec. 4.3.1), one for each evaluation criterion shown in Fig. 11. The results of the four tests are reported in Table 9. As expected, the best configuration in terms of hypervolume is the one tuned for that criterion (auto anytime), which is significantly better than the rest by a large margin. The auto anytime configuration

Table 9: Configurations of SCIP ordered according to the sum of ranks with respect to four different evaluation criteria. The numbers in parenthesis are the difference of ranks relative to the best ranked configuration. Configurations that are not significantly different from the best one according to the Friedman test are indicated in bold face.

ΔR_α	**Configurations (ΔR)**	**Evaluation criterion**
		Hypervolume
68.1	**auto anytime** (0), default (1377), auto quality (1658), auto time (2481)	
		Time to best found
63.7	**auto time** (0), default (195.5), auto quality (1845.5), auto anytime (2009)	
		Quality after 10 seconds
76.8	**auto anytime** (0), default (766), auto quality (1302), auto time (2230)	
		Final quality (300 seconds)
∞	**default** (0), **auto time** (2), **auto anytime** (21), **auto quality** (21)	

is also the clear winner in terms of the solution quality obtained if stopped after 10 seconds. Moreover, in terms of final quality, the differences between the strategies are not statistically significant. In fact, the difference in the sum of ranks between auto anytime and default is only 21.

Finally, by using the weighted hypervolume as explained in Section 5, we are able to find a configuration of SCIP with good anytime behaviour and that ranks better than default according to final quality. However, the differences in ranks are still not statistically significant. Hence, for conciseness, we do not discuss the results of using the weighted hypervolume for tuning SCIP here, but we provide the results as supplementary material [23]. The results provided here are sufficient to conclude that the proposed method was able to find a configuration of SCIP that has better anytime behaviour than the default, without a significant loss of final quality.

7. Conclusions

In this paper, we have shown that the combination of I/F-Race and the hypervolume quality measure is effective at improving the anytime behaviour of optimisation algorithms. We have presented two representative and challenging scenarios. The first scenario compares the results obtained automatically against those obtained by a human expert. The conclusion from this scenario is that both approaches obtain similar algorithmic configurations, but the automatic method saves significant human effort. In the second scenario, we apply our approach to an off-the-shelf optimisation software, with a very large number of parameters. Despite the more challenging scenario, we also obtain here a significant improvement of the anytime behaviour.

The choice of the hypervolume measure also allows us to incorporate preference information into the automatic configuration process by means of the weighted hypervolume. We propose a weighted formulation that emphasises a good final quality but still takes into account the overall anytime behaviour of the algorithms.

An open question is how to extend the results to longer termination criteria than the ones that are feasible to test during automatic configuration. A problem that may arise is that configurations produce good results up until the tested termination criterion, but the

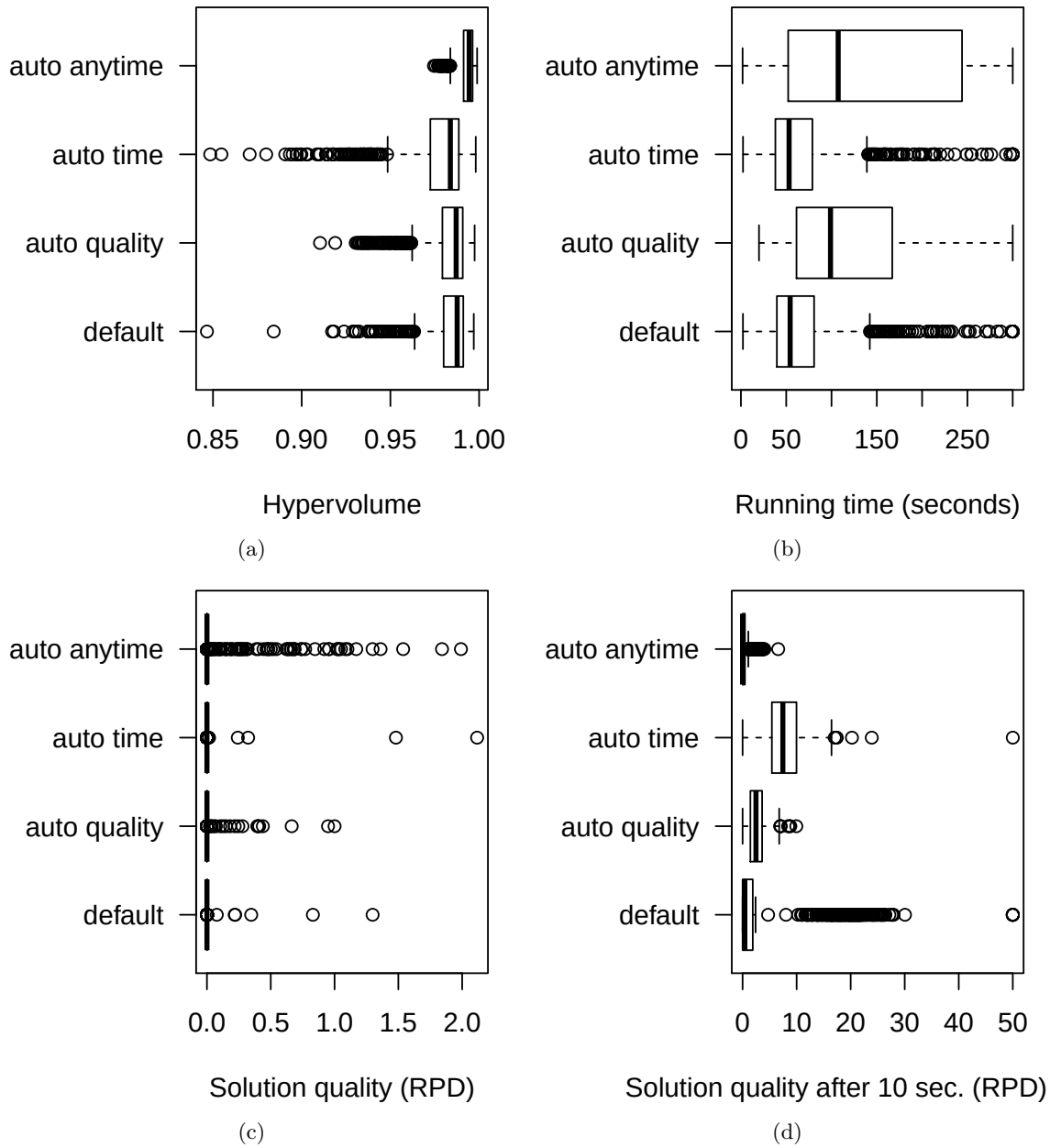


Figure 11: Boxplots of the results obtained by four different parameter configurations of SCIP according to various evaluation criteria.

performance becomes unsatisfactory for longer runs. Woodruff et al. [34] have studied how to predict the point of diminishing returns, which may be used to dynamically set a termination criterion. Survival analysis techniques may help to estimate the behaviour of the algorithms for longer runtime [18]. We are convinced that our approach contributes towards the final goal of designing algorithms that are more robust to different termination criteria and, hence, applicable to a wider range of scenarios, without sacrificing solution quality.

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 n° 246939. 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

[1] T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, July 2009.

[2] S. Aine, R. Kumar, and P. P. Chakrabarti. Adaptive parameter control of evolutionary algorithms to improve quality-time trade-off. *Applied Soft Computing*, 9(2):527–540, 2009.

[3] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Investigating and exploiting the bias of the weighted hypervolume to articulate user preferences. In F. Rothlauf, editor, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2009*, pages 563–570. ACM Press, New York, NY, 2009.

[4] T. Bartz-Beielstein. *Experimental Research in Evolutionary Computation: The New Experimentalism*. Springer, Berlin, Germany, 2006.

[5] M. Birattari. *Tuning Metaheuristics: A Machine Learning Perspective*, volume 197 of *Studies in Computational Intelligence*. Springer, Berlin/Heidelberg, Germany, 2009.

[6] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle. F-race and iterated F-race: An overview. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 311–336. Springer, Berlin, Germany, 2010.

[7] J. Branke and J. Elomari. Simultaneous tuning of metaheuristic parameters for various computing budgets. In N. Krasnogor and P. L. Lanzi, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011*, pages 263–264. ACM Press, New York, NY, 2011.

[8] M. Chiarandini. *Stochastic Local Search Methods for Highly Constrained Combinatorial Optimisation Problems*. PhD thesis, FG Intellektik, FB Informatik, TU Darmstadt, Germany, 2005.

[9] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, New York, NY, third edition, 1999.

[10] T. Dean and M. S. Boddy. An analysis of time-dependent planning. In *Proceedings of the 7th National Conference on Artificial Intelligence, AAAI-88*, pages 49–54. AAAI Press, 1988.

[11] M. den Besten. *Simple Metaheuristics for Scheduling*. PhD thesis, FG Intellektik, FB Informatik, TU Darmstadt, Germany, 2004. URL `http://tuprints.ulb.tu-darmstadt.de/516/`.

[12] M. Dorigo and L. M. Gambardella. Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1 (1):53–66, 1997.

[13] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[14] J. Dréo. Using performance fronts for parameter setting of stochastic metaheuristics. In F. Rothlauf, editor, *GECCO (Companion)*, pages 2197–2200. ACM Press, New York, NY, 2009.

[15] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle. Automatic configuration of state-of-the-art multi-objective optimizers using the TP+PLS framework. In N. Krasnogor and P. L. Lanzi, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011*, pages 2019–2026. ACM Press, New York, NY, 2011.

[16] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In F. Lobo, C. F. Lima, and Z. Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, pages 19–46. Springer, Berlin, Germany, 2007.

[17] C. M. Fonseca, L. Paquete, and M. López-Ibáñez. An improved dimension-sweep algorithm for the hypervolume indicator. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)*, pages 1157–1163. IEEE Press, Piscataway, NJ, July 2006.

[18] M. Gagliolo and C. Legrand. Algorithm survival analysis. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 161–184. Springer, Berlin, Germany, 2010.

[19] H. H. Hoos and T. Stützle. *Stochastic Local Search—Foundations and Applications*. Morgan Kaufmann Publishers, San Francisco, CA, 2005.

[20] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, Oct. 2009.

[21] K. Leyton-Brown, M. Pearson, and Y. Shoham. Towards a universal test suite for combinatorial auction algorithms. In A. Jhingran et al., editors, *ACM Conference on Electronic Commerce (EC-00)*, pages 66–76. ACM Press, New York, NY, 2000.

[22] M. López-Ibáñez and T. Stützle. Automatic configuration of multi-objective ACO algorithms. In M. Dorigo et al., editors, *Swarm Intelligence, 7th International Conference, ANTS 2010*, volume 6234 of *Lecture Notes in Computer Science*, pages 95–106. Springer, Heidelberg, Germany, 2010.

[23] M. López-Ibáñez and T. Stützle. Automatically Improving the Anytime Behaviour of Optimisation Algorithms: Supplementary material. `http://iridia.ulb.ac.be/supp/IridiaSupp2012-011/`, 2012.

[24] M. López-Ibáñez and T. Stützle. The automatic design of multi-objective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 2012. In press.

[25] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011. URL `http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf`.

[26] S. Loudni and P. Boizumault. Combining VNS with constraint programming for solving anytime optimization problems. *European Journal of Operational Research*, 191:705–735, 2008.

[27] M. Maur, M. López-Ibáñez, and T. Stützle. Pre-scheduled and adaptive parameter variation in MAX-MIN Ant System. In H. Ishibuchi et al., editors, *Proceedings of the 2010 Congress on Evolutionary Computation (CEC 2010)*, pages 3823–3830. IEEE Press, Piscataway, NJ, 2010.

[28] T. Stützle. MAX-MIN Ant System for the quadratic assignment problem. Technical Report AIDA–97–4, FG Intellektik, FB Informatik, TU Darmstadt, Germany, July 1997.

[29] T. Stützle. ACOTSP: A software package of various ant colony optimization algorithms applied to the symmetric traveling salesman problem, 2002. URL `http://www.aco-metaheuristic.org/aco-code/`.

[30] T. Stützle and H. H. Hoos. MAX-MIN Ant System and local search for combinatorial optimization problems. In S. Voss, S. Martello, I. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 137–154. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[31] T. Stützle and H. H. Hoos. MAX-MIN Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[32] T. Stützle, M. López-Ibáñez, P. Pellegrini, M. Maur, M. A. Montes de Oca, M. Birattari, and M. Dorigo. Parameter adaptation in ant colony optimization. In Y. Hamadi, E. Monfroy, and F. Saubion, editors, *Autonomous Search*, pages 191–215. Springer, Berlin, Germany, 2012.

[33] S. Wessing, N. Beume, G. Rudolph, and B. Naujoks. Parameter tuning boosts performance of variation operators in multiobjective optimization. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, volume 6238 of *Lecture Notes in Computer Science*, pages 728–737. Springer, Heidelberg, Germany, 2010.

[34] D. L. Woodruff, U. Ritzinger, and J. Oppen. Research note: the point of diminishing returns in heuristic search. *International Journal of Metaheuristics*, 1(3):222–231, 2011.

[35] S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3): 73–83, 1996.

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

[37] E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In S. Obayashi et al., editors, *Evolutionary Multi-criterion Optimization (EMO 2007)*, volume 4403 of *Lecture Notes in Computer Science*, pages 862–876. Springer, Heidelberg, Germany, 2007.