Automatically Improving the Anytime Behaviour of Multiobjective Evolutionary Algorithms

Andreea Radulescu[1], Manuel López-Ibáñez[2], and Thomas Stützle[2]

[1] LINA, UMR CNRS 6241, Université de Nantes, Nantes, France
andreea.radulescu@etu.univ-nantes.fr
[2] IRIDIA, CoDE, Université Libre de Bruxelles (ULB), Brussels, Belgium
{manuel.lopez-ibanez,stuetzle}@ulb.ac.be

Abstract. An algorithm that returns as low-cost solutions as possible at any moment of its execution is said to have a good anytime behaviour. The problem of optimising anytime behaviour can be modelled as a bi-objective non-dominated front, where the goal is to minimise both time and cost. Using a unary quality measure such as the hypervolume indicator, the analysis of the anytime behaviour can be converted into a single-objective problem. In this manner, available automatic configuration tools can be applied to improve the anytime behaviour of an algorithm. If we want to optimise the anytime behaviour of multi-objective algorithms, we may apply again unary quality measures to obtain a scalar value for measuring the obtained approximation to the Pareto front. Thus, for multi-objective algorithms, the anytime behaviour may be described in terms of the curve of the hypervolume over time, and the quality of this bi-objective tradeoff curve be evaluated according to its hypervolume. Using this approach, we can automatically improve the anytime behaviour of multi-objective evolutionary algorithms (MOEAs). In this article, we first introduce this approach and then experimentally study the improvements obtained considering three MOEAs, namely, IBEA, NSGA-II and SPEA2.

1 Introduction

In many real world problems, the quality of solutions is evaluated according to multiple objective functions. The goal of algorithmic approaches to their solution typically is to provide an as good as possible approximation to the unknown Pareto front of tradeoff solutions. Among the most successful such algorithmic approaches are multi-objective evolutionary algorithms (MOEAs).

In practical settings, the user will have a limited time to run an MOEA, and the available amount of time is not always known in advance when deciding for the MOEA's parameter settings. Thus, a goal in the design of MOEAs and other multi-objective optimizers is to find algorithm parameter settings that allow finding the best possible Pareto front approximations for any stopping criterion. More in general, algorithms that provide as good solutions as possible independent of a specific termination criterion are referred to as having good anytime behaviour [18].

In this paper, our goal is to examine and, in particular, to improve the anytime behaviour of MOEAs by automatic algorithm configuration techniques [2, 6, 9, 11, 13]. To this aim, we need to model the anytime behaviour of MOEAs. For single objective algorithms, the anytime behaviour may be modelled as a bi-objective non-dominated front, where both solutions quality and computation time must be considered [15, 16]. By using a unary quality measure of the obtained non-dominated front, the analysis of the anytime behaviour is converted into a single-objective problem [16]. Here, we follow [16] and use the hypervolume indicator for this task [22].

In addition, we need to determine the quality of the set of solutions returned by the MOEA. For this task, again the hypervolume indicator can be used and each pair of (time, hypervolume) represents an improvement of the best solutions found at a particular time since the algorithm start. The objective of optimising the anytime behaviour reduces then to find MOEA configurations that produce the best possible set of sequences of points (time, hypervolume). Hence, by making this double usage of the hypervolume, we can apply standard algorithm configuration techniques for the automatic configuration of MOEAs.

We analyse the impact of automatic configuration on the anytime behaviour of MOEAs. In particular, we selected three classical MOEAs: the indicator-based evolutionary algorithm (IBEA) [19], the nondominated sorting genetic algorithm (NSGA-II) [4], and the strength Pareto evolutionary algorithm (SPEA2) [20]. These algorithms are among the best-known MOEAs and they have been thoroughly studied in the literature.

This paper is organized as follows. Section 2 introduces basic notions of multi-objective optimization and MOEAs. In Section 3, we describe the method used for tuning anytime behaviour of MOEAs. The experimental setup and the results are described in Sections 4 and 5, respectively. We conclude in Section 6.

2 Multi-objective optimization

A multiobjective optimization problem (MOP) can be formulated as

$$\text{minimize} \quad \boldsymbol{f}(x) = (f_1(x), \ldots, f_m(x))^T \quad \text{subject to} \quad x \in \Omega$$

where Ω is the search space, $\boldsymbol{f} \colon \Omega \to \mathbb{R}^m$ consists of m real-valued objective functions and $\mathbb{R}^m$ is called objective space. A continuous MOP is an MOP where each of the D variables is a continuous variable with $x_i \in \mathbb{R}\ \forall x_i, i = 1, \ldots, D$ and possible constraints restrict the set of feasible solutions.

Typically, the objectives of an MOP are conflicting and there is no solution $x \in \Omega$ that minimizes all objectives simultaneously. The solutions representing the best compromise between the objectives can be defined in terms of Pareto optimality. Let u, v be two vectors in $\mathbb{R}^m$; u is said to dominate v if and only if $u_i \leq v_i$ for every $i \in 1, \ldots, m$ and $u_j < v_j$ for at least one index $j \in 1, \ldots, m$. This definition applies without loss of generality to minimization problems. A point $x^* \in \Omega$ is Pareto optimal if there is no point $x \in \Omega$ such that $\boldsymbol{f}(x)$ dominates $\boldsymbol{f}(x^*)$. A Pareto set is the set of all Pareto optimal points and the Pareto front is the set of the objective vectors of all Pareto optimal points.

In order to measure the quality of a Pareto front, we can use the hypervolume indicator [22]. The unary hypervolume indicator measures the quality of a set P of n non-dominated objective vectors produced in a run of a multiobjective optimizer. For a minimization problem involving m objectives, this indicator measures the region that is simultaneously dominated by P and bounded above by a reference point $r \in \mathbb{R}^m$ such that $r > (\max_p p_1, \ldots, \max_p p_m)$, where $p = (p_1, \ldots, p_m) \in P \subset \mathbb{R}^m$.

3 Anytime optimization

An anytime algorithm returns as high quality solutions as possible at any moment of its execution [18]. One characteristic of anytime algorithms is that, independently of the termination criterion, the best solution found so far is steadily improved, eventually finding the optimal. This implies that anytime algorithms should keep exploring the search space and avoid getting trapped in local optima. Moreover, good solutions should also be discovered as early as possible. This implies that the algorithm should converge to good solutions as fast as possible.

Normally, there is a trade-off between the quality of the solution and the runtime of the algorithm. There are two classical views when analyzing this trade-off. One view defines a number of termination criteria and analyzes the quality achieved by the algorithm at each termination criterion. In this quality-versus-time view, the anytime behaviour is often analyzied as a plot of the (average) solution quality over time, also called SQT curve (e.g., Fig. 1). A different view defines a number of target quality values and analyzes the time required by the algorithm to reach each target. In this time-versus-quality view, algorithms are often analyzed in terms of their runtime distribution [10, Chapter 4].

We consider here a third view that does not favor time over quality or viceversa, but models the anytime behaviour as a bi-objective problem [16] where the first objective, the solution quality, has to be maximized, while the second, the time, has to be minimized. If we consider the points (*quality*, *time*) that describe at which time the quality of the best solution has been improved by the algorithm, then the set of points that describe a run of an algorithm is, by definition, a nondominated set of solutions. Moreover, we can definitely compare the anytime behaviour of two algorithms by comparing their respective nondominated sets of (*quality*, *time*) points.[3] If the nondominated set of one algorithm dominates (in the Pareto-optimality sense) the nondominated set of another algorithm, we say that the anytime behaviour of the former is better than the

[3] In our approach, we do not give more importance to either *time* or *quality*, and, hence, the order in which they are plotted is irrelevant. However, we visualize later the results in terms of SQT curves, and, hence, the order (*quality*, *time*) is more natural for such purpose.

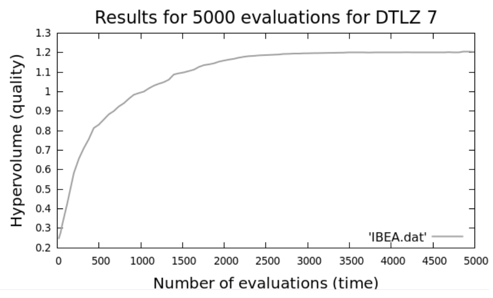


Fig. 1. An example of SQT curve.

anytime behaviour of the latter. More importantly, this model allows us to apply the same unary quality measures used in multi-objective optimization to evaluate the anytime behaviour of algorithms. In particular, we have identified the hypervolume measure [22] as being the most suitable for this purpose [16].

In the case of single-objective optimisers, the output of an algorithm is a single solution, and its quality is a unique scalar value. In the proposal described above, their anytime behaviour is evaluated as a bi-objective problem with two objectives (*quality*, *time*). In the case of multi-objective optimisers such as MOEAs, the output of the algorithm is a nondominated set of points, and, hence, we need an additional step that assigns a unique scalar quality value to each nondominated set by means of a unary quality measure. Once each nondominated set is assigned a scalar quality value, we can proceed as described above and evaluate the anytime behaviour of MOEAs as the hypervolume of the nondominated set of points (*quality*,*time*) that describes a run of an algorithm.

In summary, our proposal for assessing the anytime behaviour of a run of a MOEA consists of two main steps:

- *Compute the quality of the best nondominated set found by a run of an algorithm at each moment of its execution.* This entails recording every improvement of the best nondominated set. Since this may turn out to be computationally expensive for long runs, a good approximation is to record improvements of the best nondominated set only at specific time intervals. Then, we may compute the quality of these nondominated sets by means of any unary quality measure. For simplicity, we have chosen here the hypervolume measure. In this way, we obtain a nondominated set of (*quality*, *time*) points that describe the anytime behaviour of the run of the algorithm.
- *Compute the quality of the anytime behaviour curve.* At the end of the run of an algorithm, its anytime behaviour is evaluated by computing the hypervolume measure of the nondominated set of (*quality*, *time*) points obtained in the previous step. In the case of a fixed frequency of time steps, the hypervolume computation in this step can be simplified as the sum of the qualities over each time step. However, the use of the hypervolume is more general

and it allows the introduction of preference information [16], although we do not examine this possibility in this paper.

Although one may use a different quality measure in the first step than the hypervolume measure used in the second step, we have chosen to use the hypervolume in both steps for simplicity. In this case, the above steps are equivalent to extending the original multi-objective problem with an additional objective (*time*) and replacing the two steps above by simply computing the hypervolume of the extended problem. We prefer the two-step approach described above because it clearly separates between the computation of the quality of a solution (set) to the multi-objective problem at hand, and the evaluation of the anytime behaviour of the algorithm.

A practical application of the above proposal is the automatic configuration of the anytime behaviour of MOEAs. In all parametrized algorithms, such as MOEAs, the search behaviour is heavily influenced by their parameter setting. The goal of automatic algorithm configuration is to determine the settings of both numerical and categorical parameters before the algorithm is actually deployed in order to have an algorithm that is as high performing as possible. Automatic algorithm configuration is crucial in the design phase of parametrized algorithms. It is also relevant in practical applications when known algorithms are applied to specific classes of problems, in order to find the parameter settings that optimise performance for such problems.

Automatic algorithm configuration consists of two main phases:

– *tuning phase*: the algorithm is tuned on a representative set of problem instances;
– *production (or testing) phase*: a chosen algorithm configuration is used to solve unseen problem instances.

In recent years, a number of automatic configuration methods have been developed and recent overviews are available in the literature [2, 6, 8, 9]. The method proposed above to automatically improve the anytime behaviour of MOEAs is mostly independent of the automatic configuration method used. In this paper, we use as automatic configuration method the implementation of I/F-Race [2] provided by the **irace** software package [13]. We combine this method with a publicly available implementation of the hypervolume measure [7] to tune the anytime behaviour of MOEAs.

4 Experimental setup

In this section, we first present the three analysed MOEAs, explain the main parameters, and introduce the benchmark problems used in the experiments.

NSGA-II [4] uses nondominated sorting and a density estimator to rank the generated solutions and to construct a fixed-size elite population.

SPEA2 [20] keeps the best solutions in an fixed-size elite archive. After each generation, the archive is either truncated with an operator based on k-th nearest neighbour or completed with dominated solutions from the current population.

Table 1. Parameter space for tuning of IBEA, NSGA-II and SPEA2.

Parameter	**Role**	**Type**	**Range**	**Default**
p_{c}	probability of mating two solutions	real	(0.0,1.0)	1.0
$p_{\mathrm{m}}^{\mathrm{ext}}$	probability of mutating a solution	real	(0.0,1.0)	1.0
$p_{\mathrm{m}}^{\mathrm{int}}$	probability of mutating a variable in a solution	real	(0.0,1.0)	0.0833
N	number of solutions	integer	$[10, 1000]$	20
DI_{c}	distance between children and their parents	integer	$[0, 100]$	15
DI_{m}	distribution of the mutated values	integer	$[0, 100]$	20
l	scaling factor (IBEA)	real	(0.0, 1.0)	0.05
$\overline{N}$	archive size (SPEA2)	integer	$[10, 1000]$	100
k	k-th nearest neighbour (SPEA2)	integer	$[1, 50]$	10

IBEA [19] uses a binary quality indicator, in particular, the binary additive ϵ-indicator, in order to assign a fitness value to each solution and to keep a fixed-size elite population.

In this paper, we use the implementation of these three algorithms available in *ParadisEO* [12], a software framework dedicated to the flexible design of metaheuristics. For the experiments in this paper, all algorithms use the simulated binary crossover (SBX) operator [3] and polynomial mutation.

MOEAs parameters. Table 1 summarises the parameters of the three MOEAs tested (IBEA, NSGAII and SPEA2), their default values and the range considered for tuning. The default values are the ones used in the ParadisEO framework [12], and mostly correspond to the values suggested in the literature [4,5,19,20,21]. There are six common parameters: the population size (N); the probability of crossover (p_{c}); the probability of external mutation ($p_{\mathrm{m}}^{\mathrm{ext}}$), which determines whether a solution will be mutated; the probability of internal mutation ($p_{\mathrm{m}}^{\mathrm{int}}$), which determines which variables of a solution will be mutated; the crossover distribution index (DI_{c}), which determines the amount of exploration outside the parents, and the mutation distribution index (DI_{m}), which determines the distance between the original and the mutated value of a variable. Besides these common parameters, IBEA has a parameter l called the fitness scaling factor, which is used for computing the fitness values and it depends on the indicator used in the algorithm. SPEA2 has two additional parameters: the archive size $\bar{N}$, and the k-th nearest neighbour, which affects the density estimation operator.

Benchmark problem instances. As benchmark instances, we consider real-valued functions from two well-known benchmark sets: ZDT [21] and DTLZ [5]. The original ZDT set contains six bi-objective functions, but ZDT5 was not included in our setup. From the DTLZ set we used the seven functions with three objectives. Note that these benchmark sets are scalable to any number of decision variables (D) and they pose different difficulties to multi-objective algorithms such as non-convex fronts or discontinuous fronts.

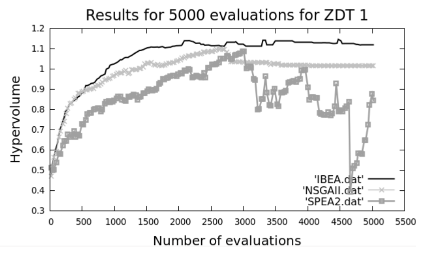
Monotonicity of hypervolume in MOEAs. MOEAs store the best non-dominated set found in an elite population that acts as an archive of solutions. Ideally, the quality of this archive should monotonically increase over time. However, MOEAs often limit the size of this archive. Most archiving algorithms, e.g. the one used by SPEA2 and NSGA-II, are not monotonic with respect to dominance [14], and even if the archiving algorithm is monotone with respect to dominance, such as the one of IBEA, the hypervolume of the archive does not need to be strictly monotonic (see below).

Figure 2 shows SQT curves of the three MOEAs considered here, in terms of hypervolume development over the number of function evaluations. In the left column, we plot the SQT curve corresponding to the elite population, whereas the right column corresponds to the SQT curve of an external, unbounded archive that stores all the dominated solutions found so far within a single run. The plots show that an unbounded archive results in a monotonic increase of the hypervolume over time. Moreover, the quality of the unbounded archive is significantly better than the quality of the elite population. Therefore, in the following, we always make use of an external unbounded archive.

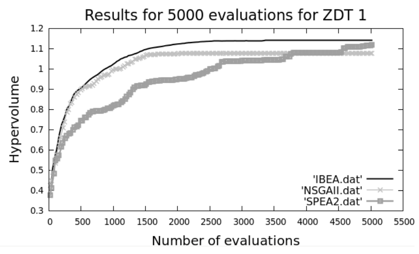
Tuning setup. We use Iterated F-race [2], as implemented by the `irace` software [13], to tune the parameters of the MOEAs. Each tuned parameter configuration is obtained by running `irace` with a budget of 2 000 runs of the MOEA being tuned. Each MOEA run is stopped after $100D$ function evaluations, where D is the number of variables of the problem.

In order to increase the effectiveness of the tuning, each run of `irace` was repeated 10 times with different random seed, and the ten resulting MOEA configurations were compared using F-race [2] in order to select the best one.

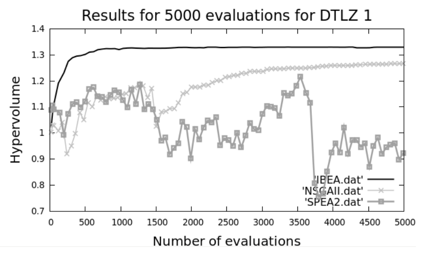
Each MOEA is tuned separately on each set of benchmark instances, namely, one run of `irace` uses DTLZ instances and another ZDT instances. Due to the diversity of the benchmark sets, the training instances are setup in a special way. We split each benchmark set into training instances used for tuning and testing instances used for comparing the configurations obtained after tuning. For tuning, the input of `irace` is a stream of instances that is structured in blocks, each block containing one function of each type (5 functions for ZDT and 7 functions for DTLZ) and random $D \in [10, 100]$. We setup `irace` in such a way that configurations are run at least on two blocks of instances before discarding any configuration, and we only discard configurations after evaluating each surviving configuration on a whole block of instances. For testing, we select each function with $D \in \{20, 35, 50, 65, 80, 95\}$ (these values for D are excluded from the training set).



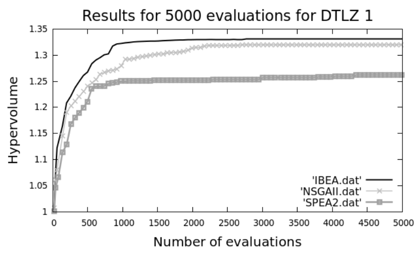
(a) ZDT1 with a fixed-sized archive



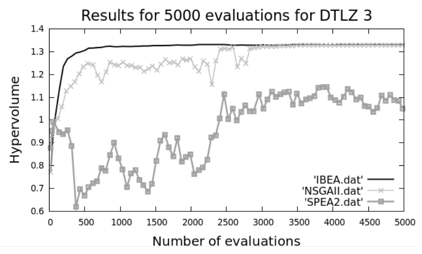
(b) ZDT1 with an unbounded archive



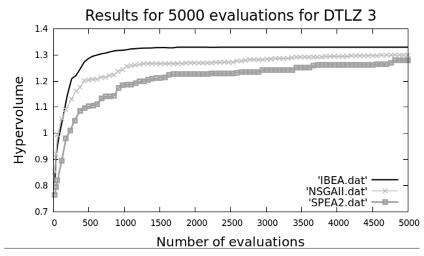
(c) DTLZ1 with a fixed-sized archive



(d) DTLZ1 with an unbounded archive



(e) DTLZ3 with a fixed-sized archive



(f) DTLZ3 with an unbounded archive

Fig. 2. Monotonicity preservation of the hypervolume through an unbounded external archive.

5 Experimental results

Tuning for anytime behaviour. Figure 3 illustrates the difference between the results obtained with the default values and with the best configuration tuned for the anytime behaviour. The values plotted represent the hypervolume value of the SQT curves for each test problem (that is, for each function in each of $D \in \{20, 35, 50, 65, 80, 95\}$). Each point represents the mean quality value obtained after 15 runs of a particular configuration. A larger hypervolume value indicates a better anytime behaviour. The plots clearly show that, as ex-

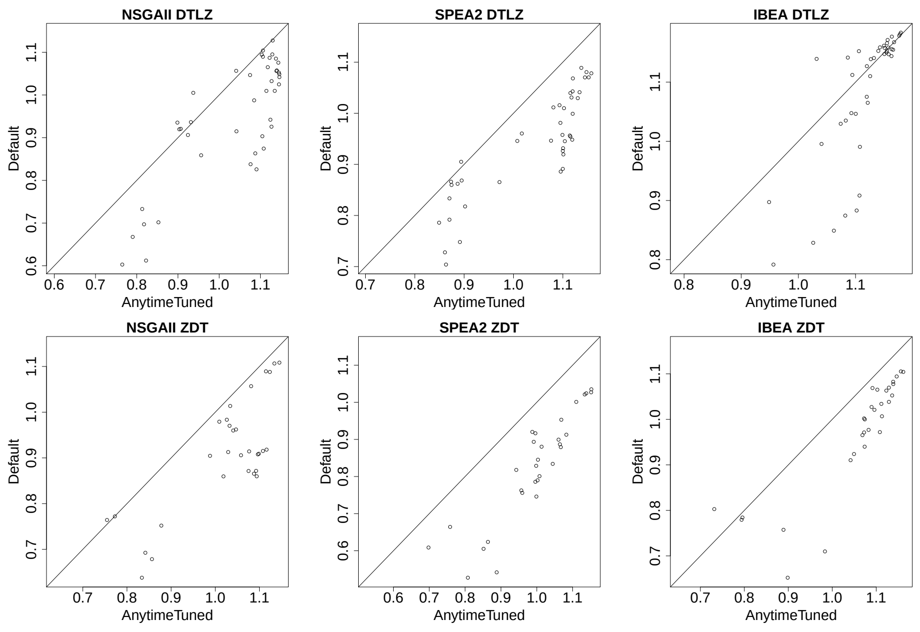


Fig. 3. Anytime behaviour quality for the default configuration versus configurations tuned for anytime behaviour. Each point represents an "unseen" instance from the ZDT set or the DTLZ set and gives the mean hypervolume of the SQT curves (measured across 15 runs) for the default configuration (y-axis) and the anytime tuned configuration (x-axis). A point below the diagonal indicates better results by the anytime tuned configuration.

pected, the hypervolume of the SQT curves obtained by the MOEAs improve significantly after tuning.

In order to assess whether improving the hypervolume of the SQT curve results in a visible improvement of the anytime behaviour of the MOEAs, we plot in Fig. 4 the mean SQT curve over 15 runs of each MOEA on individual benchmark instances. Moreover, we plot for each curve the 95% confidence interval around the mean as a grey shadow to give an idea of the variation over multiple runs. In all cases, the anytime behaviour of the MOEAs visibly improves after tuning over default settings. The improvement is very strong for NSGA-II and SPEA2, which shows that their default parameters are far from ideal.

Next, we examine whether the final quality of the non-dominated set obtained at the maximum termination criterion ($100 \cdot D$) is improved or not by tuning for anytime behaviour. Figure 5 compares the hypervolume of this final non-dominated set when generated by the default configuration versus the one generated by the configuration tuned for anytime behaviour. In the case of NSGA-II and SPEA2, the final quality is improved in most cases. In the case of IBEA, the final quality is clearly improved in a few cases, but the differences are often rather small.

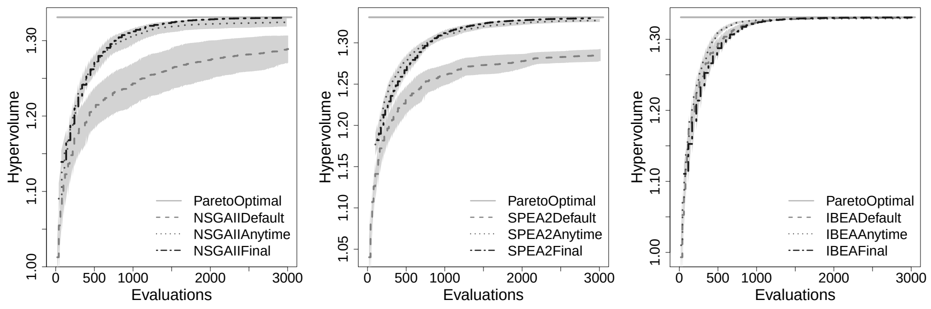


Fig. 4. Variation of the quality of the Pareto front obtained for the three different configurations: the default parameter set, the best parameter set tuned for the anytime behaviour and the best parameter set tuned for the quality of the final Pareto front. The instance DTLZ1 with 30 variables was executed 15 times for each MOEAs

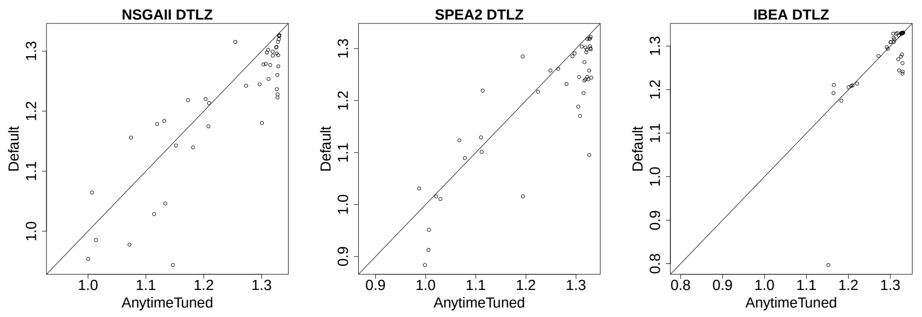


Fig. 5. Final Pareto front quality for the default configuration versus configurations tuned for anytime behaviour. Each point represents an "unseen" instance from the DTLZ set and gives the mean hypervolume of the final Pareto front approximation (measured across 15 runs) for the default configuration (y-axis) and the anytime tuned configuration (x-axis). A point below the diagonal indicates better results by the anytime tuned configuration.

Tuning for final quality. We also evaluate the MOEAs tuned for anytime behaviour relative to how much improvement could be reached by tuning for final quality, the latter being a more traditional approach. When tuning for final quality, the tuning procedure ignores the SQT curve and only takes into account the quality of the non-dominated set obtained after $100 \cdot D$ function evaluations. We measure the quality of this final non-dominated set according to the hypervolume. Otherwise, we follow the same tuning setup as described above. After tuning, we obtain a parameter configuration for each MOEA and each benchmark set and run the configurations 15 times on each test instance.

Figure 6 compares the configurations tuned for anytime behaviour and the configurations tuned for final quality when evaluated with respect to their anytime behavior for the three MOEAs and the two benchmark function sets. (The configurations tuned for anytime behaviour and final quality turned out to be

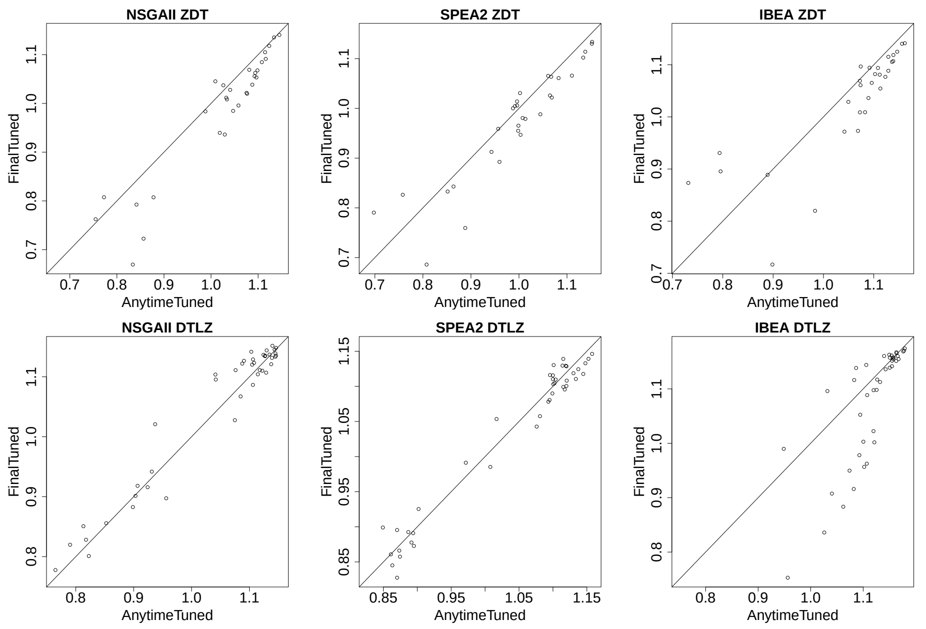


Fig. 6. Difference for **anytime behaviour quality** obtained for the best parameter set tuned for anytime behaviour and the best parameter set tuned for quality of the final Pareto front. Each point represents an "unseen" instance from the ZDT and the DTLZ benchmark sets.

far better than the default configurations and therefore the plots comparing to the default configurations are omitted here.) As the plots show, the configurations tuned for anytime behaviour obtain generally better results for the anytime behaviour than those tuned for the final quality.

Figure 7 shows that, in terms of final quality, the differences between the configurations tuned for the anytime behaviour and the configurations tuned for the final quality are only slightly in favor of the latter with the exception of few outliers in the cases of NSGA-II on the DTLZ set and IBEA and SPEA2 on the ZDT sets. Thus, the improved anytime behaviour does not necessarily incur a strong loss with respect to the final quality reached.

By comparing the three MOEAs on the same function (here as example DTLZ1, $D = 30$), we can observe that IBEA is typically the best performing algorithm for the default settings, and the configurations tuned for anytime behaviour or final quality (Fig. 8). However, while for the default settings the advantage of IBEA is often substantial, after tuning NSGAII and SPEA2 are typically strongly improved and can become competitive.

The differences in the behaviour of the algorithms before and after tuning may be explained by the different parameter settings. Table 2 shows the default configuration, and the parameter configurations found when tuning for anytime behaviour and when tuning for final quality. The most notable differences

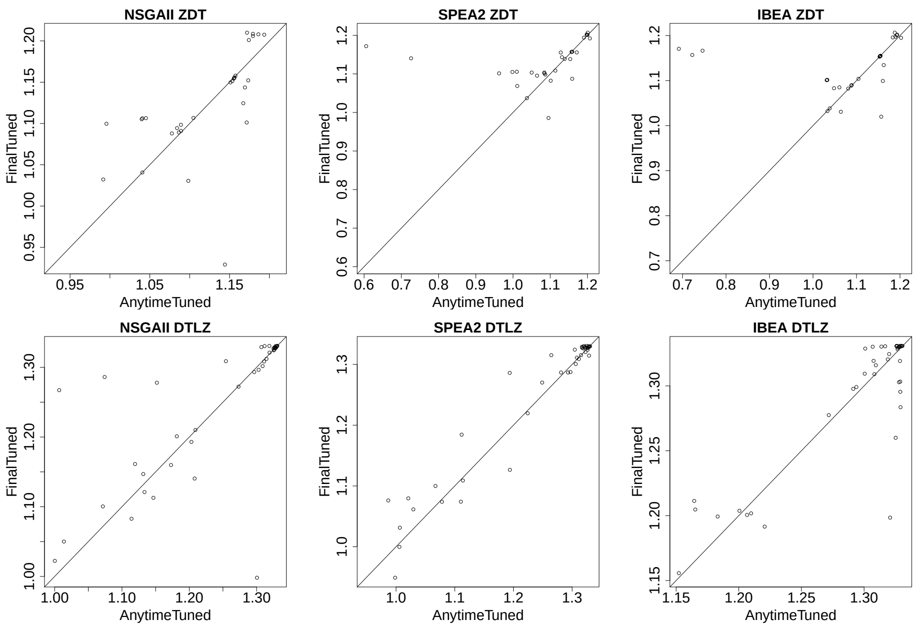


Fig. 7. Difference for the quality of the **final Pareto front** obtained the best parameter set tuned for anytime behaviour and the best parameter set tuned for quality of the final Pareto front. Each point represents an "unseen" instance from the ZDT and the DTLZ benchmark sets.

with respect to the default parameters is the larger population size for the configurations optimising anytime behaviour and final quality. Interestingly, when optimising performance for final quality, the population sizes are in all cases larger than when optimising for anytime behaviour. This suggests to us that an incremental population approach could probably help to improve the anytime behaviour of these MOEAs. The effect of other parameters is more difficult to interpret, but the lack of a clear trend suggests that simple time-varying parameter adaptation strategies may not be successful. A more profound analysis, for example, by tuning the parameters for various termination criteria, could help to reveal some exploration versus exploitation trade-offs. A more direct approach would be to implement a large number of parameter variation strategies and find the best strategy by means of the technique demonstrated in this paper, that is, by automatic configuration with respect to anytime behaviour.

6 Conclusions

In this article, we have shown that the quality of the anytime behaviour of MOEAs can be improved significantly by using the combination of Iterated F-race [2,13] and the hypervolume quality measure. For this task, the hypervolume

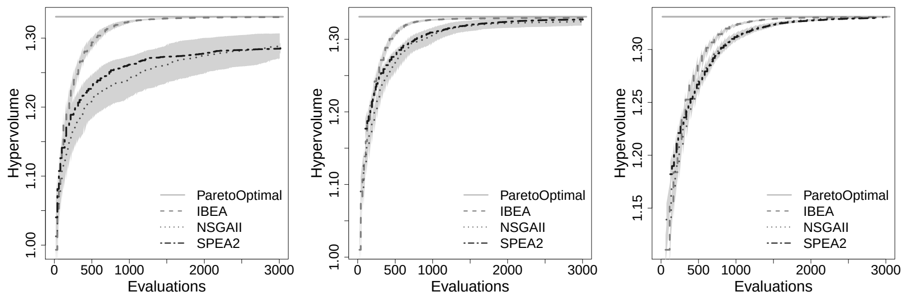


Fig. 8. Variation of the quality of the Pareto front approximation obtained by the three MOEAs with default parameter sets (left plot), parameter sets tuned for anytime behaviour (middle plot) and parameter sets tuned for final quality (right plot).

Table 2. Best values of the tuned parameters for IBEA, NSGAII and SPEA2. Fo each algorithm and benchmark set the tuned values for optimising the anytime behavior (any) or the final quality (final) are given together with the default values.

		IBEA				NSGAII				SPEA2			
Para-	default	DTLZ		ZDT		DTLZ		ZDT		DTLZ		ZDT	
meter	values	any	final	any	final	any	final	any	final	any	final	any	final
p_c	1.0	0.824	0.844	0.935	0.915	0.928	0.755	0.398	0.843	0.172	0.190	0.823	0.790
$p_\mathrm{m}^\mathrm{ext}$	1.0	0.468	0.371	0.799	0.796	0.131	0.066	0.290	0.921	0.027	0.099	0.823	0.743
$p_\mathrm{m}^\mathrm{int}$	0.083	0.848	0.756	0.915	0.779	0.578	0.894	0.807	0.966	0.099	0.130	0.876	0.857
N	20	23	60	32	60	39	72	46	79	101	123	49	71
DI_c	15	81	71	96	69	94	99	65	31	30	87	69	37
DI_m	20	18	69	0	1	15	11	1	0	20	36	0	0
l	0.05	0.136	0.065	0.317	0.148	-	-	-	-	-	-	-	-
$\overline{N}$	100	-	-	-	-	-	-	-	-	184	482	209	340
k	10	-	-	-	-	-	-	-	-	11	16	20	41

measure is used in two places. First, to measure the quality of the Pareto-front approximations generated by the MOEAs; second, to measure the quality of the anytime behaviour as defined by the trade-off curve of the solution quality over time. We have applied the resulting methodology to three different MOEAs: IBEA, NSGA-II and SPEA2.

We have combined the above approach with an automatic configuration method (`irace`) in order to automatically improve the anytime behaviour of the MOEAs. However, the proposed method is not restricted to `irace`, and other automatic configuration methods could be used for this purpose. Future work should investigate whether some configuration methods are more suited for this task than others.

The experimental results presented in this paper showed that a considerable improvement of the anytime behaviour could be obtained for all three algorithms. Additionally, tuning for the anytime behaviour improves also (with respect to anytime behaviour) over a version of the three algorithms that is tuned for optimising the quality of the final Pareto front approximation. This can be seen as a confirmation that the optimisation goal of improving anytime behaviour actually leads to algorithms configurations that are more robust to different termination criteria without sacrificing much of the solution quality that may be obtained by the algorithm. Most importantly, these improvements are obtained by an automatic method, which is saving substantial human effort.

In initial research efforts, we have considered the automatic tuning and configuration of the anytime behaviour of single objective optimisation algorithms [15, 16]. This here is the first attempt to apply the automatic configuration of the anytime behaviour to multi-objective algorithms, especially MOEAs. There are several directions that can be taken to extend this work. A first one is to extend the analysis of the impact of automatic tuning to others MOEAs, such as SMS-EMOA [1] or MOEA/D [17]. The quality of the anytime behaviour can be tested also with others benchmark problems, with more complicated objective functions. A second direction is to consider more parameters, including also categorical parameters such as the choice of the cross-over and the mutation operators. In this paper, we only consider static parameters as done in the original MOEAs. A promising approach when dealing with anytime optimization is to consider parameter variation strategies, and automatically configure the parameters of such strategies w.r.t. anytime behaviour [16]. Moreover, we have limited ourselves to parameters independent of problem instance features. Nevertheless, it would be straightforward to define parameters as functions of instance features and tune the parameters that define such functions. A more challenging question is how to detect automatically which features should influence which parameters. We are convinced that further work in the directions indicated here, will lead to the design of multi-objective algorithms that are more robust with respect to the choice of specific termination criteria and, thus, will improve the practice of multi-objective evolutionary algorithms.

Acknowledgements. 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. Manuel López-Ibáñez and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are a postdoctoral researcher and a research associate, respectively. Andreea Radulescu acknowledges support through the Laboratoire d'Informatique de Nantes-Atlantique (LINA) and Nantes university. The authors also acknowledge support from the FRFC project "*Méthodes de recherche hybrides pour la résolution de problèmes complexes*".

References

1. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research 181(3), 1653–1669 (2007)
2. 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, pp. 311–336. Springer, Berlin, Germany (2010)
3. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search spaces. Complex Systems 9(2), 115–148 (1995)

4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), 181–197 (2002)
5. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. TR 112, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, Zürich, Switzerland (2001)
6. Eiben, A.E., Smit, S.K.: Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm and Evolutionary Computation 1(1), 19–31 (2011)
7. Fonseca, C.M., Paquete, L., López-Ibáñez, M.: An improved dimension-sweep algorithm for the hypervolume indicator. In: Congress on Evolutionary Computation (CEC 2006), pp. 1157–1163. IEEE Press, Piscataway, NJ (2006)
8. Hoos, H.H.: Automated algorithm configuration and parameter tuning. In: Hamadi, Y., Monfroy, E., Saubion, F. (eds.) Autonomous Search, pp. 37–71. Springer, Berlin, Germany (2012)
9. Hoos, H.H.: Programming by optimization. Communications of the ACM 55(2), 70–80 (Feb 2012)
10. Hoos, H.H., Stützle, T.: Stochastic Local Search—Foundations and Applications. Morgan Kaufmann Publishers, San Francisco, CA (2005)
11. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: an automatic algorithm configuration framework. Journal of Artificial Intelligence Research 36, 267–306 (2009)
12. Liefooghe, A., Jourdan, L., Talbi, E.G.: A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research 209(2), 104–112 (2011)
13. López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The irace package, iterated race for automatic algorithm configuration. Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium (2011),
14. López-Ibáñez, M., Knowles, J.D., Laumanns, M.: On sequential online archiving of objective vectors. In: Takahashi, R.H.C., et al. (eds.) EMO 2011, LNCS, vol. 6576, pp. 46–60. Springer, Heidelberg, Germany (2011)
15. López-Ibáñez, M., Liao, T., Stützle, T.: On the anytime behavior of IPOP-CMA-ES. In: Coello Coello, C.A., et al. (eds.) PPSN 2012, Part I, LNCS, vol. 7491, pp. 357–366. Springer, Heidelberg, Germany (2012)
16. López-Ibáñez, M., Stützle, T.: Automatically improving the anytime behaviour of optimisation algorithms. Tech. Rep. TR/IRIDIA/2012-012, IRIDIA, Université Libre de Bruxelles, Belgium (2012)
17. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. 11(6), 712–731 (2007)
18. Zilberstein, S.: Using anytime algorithms in intelligent systems. AI Magazine 17(3), 73–83 (1996)
19. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., et al. (eds.) PPSN VIII, LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg, Germany (2004)
20. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K., et al. (eds.) Evolutionary Methods for Design, Optimisation and Control. pp. 95–100. CIMNE, Barcelona, Spain (2002)
21. Zitzler, E., Thiele, L., Deb, K.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8(2), 173–195 (2000)
22. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. 7(2), 117–132 (2003)