Multi-objective Selection of Algorithm Portfolios: Experimental Validation

Daniel Horn[1(✉)], Karin Schork[1], and Tobias Wagner[2]

[1] Computational Statistics, Technische Universität Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany
{daniel.horn,karin.schork}@tu-dortmund.de
[2] Institute of Machining Technology (ISF), Technische Universität Dortmund, Baroper Str. 303, 44227 Dortmund, Germany
wagner@isf.de

Abstract. The selection of algorithms to build portfolios represents a multi-objective problem. From a possibly large pool of algorithm candidates, a portfolio of limited size but good quality over a wide range of problems is desired. Possible applications can be found in the context of machine learning, where the accuracy and runtime of different learning techniques must be weighed. Each algorithm is represented by its Pareto front, which has been approximated in an a priori parameter tuning. Our approach for multi-objective selection of algorithm portfolios (MOSAP) is capable to trade-off the number of algorithm candidates and the respective quality of the portfolio. The quality of the portfolio is defined as the distance to the joint Pareto front of all algorithm candidates. By means of a decision tree, also the selection of the right algorithm is possible based on the characteristics of the problem.

In this paper, we propose a validation framework to analyze the performance of our MOSAP approach. This framework is based on a parametrized generator of the algorithm candidate's Pareto front shapes. We discuss how to sample a landscape of multiple Pareto fronts with predefined intersections. The validation is performed by calculating discrete approximations for different landscapes and assessing the effect of the landscape parameters on the MOSAP approach.

Keywords: Multi-objective optimization · Algorithm selection · Performance assessment · Benchmarking

1 Motivation

In algorithm selection tasks, it is still common practice to tune and compare competing algorithms or models with respect to a single performance measure. For instance, the mean error rate in classification or the best obtained function

D. Horn—We acknowledge partial support by the Mercator Research Center Ruhr under grant Pr-2013-0015 *Support-Vektor-Maschinen für extrem große Datenmengen.*

J. Handl et al. (Eds.): PPSN XIV 2016, LNCS 9921, pp. 421–430, 2016.
DOI: 10.1007/978-3-319-45823-6_39

value in optimization. The best algorithm can easily be selected based on the performance value. Often, however, additional performance measures are worth consideration. For instance, the budget of computation time or function evaluations could be considered as a second criterion. Since the performance measures are likely to be contradicting, both the tuning and the selection have to be adjusted. During the parameter tuning, the respective Pareto front has to be approximated for each algorithm. As a consequence, sets of solutions are compared in the selection step. As there is likely no single best candidate, the joint Pareto front is formed by a set of algorithms. In multi-objective selection of algorithm portfolios (MOSAP), we aim at approximating this subset of algorithms to allow selecting the best algorithm for a specific task a posteriori.

A possible application is the training of support vector machines (SVMs). Since the training of a single kernelized SVM scales at least quadratically with the number of observations, exact SVMs may be inapplicable for large datasets. Many approximative solvers have been introduced to compensate for this drawback. We conducted an exhaustive benchmark comparing the accuracy and the training time of some representative solvers in a multi-objective way [7].

To the best of our knowledge, there is no other work on the MOSAP topic. After the conceptual ideas of our MOSAP approach have been proposed and tested on the SVM application [6], we are now interested in benchmarking and validating MOSAP methods. In particular, we want to evaluate the performance of our own approach. To accomplish this, we propose a generator for constructing artificial data samples of candidate algorithms with known global Pareto fronts. Based on this generator, the performance of the resulting portfolios is evaluated for different properties of the generated data sets.

2 Multi-objective Selection of Algorithm Portfolios

In general, the performance of a set of r algorithms $\mathcal{A} = \{A_1, \ldots, A_r\}$ with respect to m objectives $(y_1, \ldots, y_m) \in Y^m$ shall be evaluated. Each algorithm $A_i\,(i = 1, \ldots, r)$ has its own set of parameters. We assume that a multi-objective parameter tuning has been performed in advance for each algorithm A_i. The resulting discrete approximation of the respective Pareto front is denoted as $PF(A_i)$. We focus on the common case of two objectives $(y_1, y_2) \in Y^2$.

Usually, there is stochasticity in the tuning results (e.g., random start designs in the optimization [8]). We assume that each tuning has been replicated $n > 1$ times, resulting in n independent approximations of $PF_j(A_i)\,(j = 1, \ldots, n)$ for each algorithm. From these replications, we can compute the empirical attainment function [4]. In this paper, we use the median front (50 %-EAF) as representative of the outcome of each algorithm.

Our MOSAP approach is divided into three independent steps. In the first step, unnecessary candidate algorithms producing so-called interfering fronts are detected. Interfering fronts are completely dominated by the fronts of the other candidates and therefore do not contribute to the joint Pareto front. In our approach, we remove algorithms that are completely dominated in η of

the replications. In the second step, we build a subset of algorithms with a reasonable trade-off between the size and the quality with regard to the joint Pareto front. This selection is a bi-objective decision making problem, as we aim to minimize the size of the subset and to maximize its quality. We define the quality of a given subset as the negative gap between its representative Pareto front and the joint Pareto front of all algorithms. The gap can be measured by any binary performance indicator, for example the hypervolume [12]. The decision making is implemented by optimizing the augmented Tschebyscheff norm [9] with a predefined weight vector w. In the third step, we aim at defining a decision rule for selecting the candidate algorithm for a specific problem. As we assume a bi-objective problem, we know that for non-dominated points the value of the second objective will decrease while the value of the first one increases. Hence, the solutions of the Pareto front can be indexed with regard to the first objective y_1. The domain of this objective is partitioned into intervals $[x_1, x_2], [x_2, x_3], \ldots, [x_{t-1}, x_t]$. Each interval is assigned to a specific algorithm A_i. For approximating this mapping, we calculate the joint non-dominated 50 %-EAF of all remaining algorithms and learn a decision tree [1] with input parameter y_1. To avoid that x_{k-1} and x_k $(k = 2, ..., t)$ are too close to each other, the decision tree is pruned with complexity control parameter *cp*.

In this paper, we aim at selecting an almost comprehensive portfolio of algorithms, therefore we parametrize our method as follows: $\eta = 0.5$, $cp = 0.1$ and $w = (0.01, 0.99)$ (it is more desirable to have a small gap than a small portfolio).

3 Test Case Generator

Our goal is the automatic construction of artificial test cases that resemble the real data we observed in the SVM benchmark [7], but are also able to take rather different shapes. In Fig. 1 an example of real data is displayed.

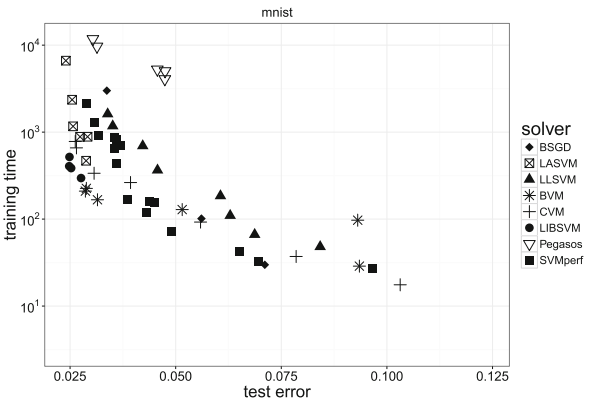

Fig. 1. Result of a biobjective parameter tuning (test error versus training time) of different approximative SVM solvers on the mnist dataset [7].

Our framework for creating the test cases consists of four steps. In the first step, we propose a flexible parametrized class of convex Pareto front shapes. By adjusting the parameters we are able to generate different Pareto fronts with predefined locations and shapes. In the second step, the sampling is extended to

sets containing multiple Pareto fronts corresponding to r different algorithms. In this set, we differentiate between two types of Pareto fronts: active fronts that do have a contribution to the joint Pareto front and interfering fronts without contribution. The active fronts are constructed under consideration of predefined intersection points. In the third step, we describe how to generate discrete approximations from the continuously defined fronts in order to simulate the outcome of each algorithm. We propose four methods with different types of distributions and approximation error. In the fourth step, we discuss how to create n noisy replications of these discrete approximations.

Class of Functions. We define a parametrized function family $y = e^{-ax} - bx$ for $a, b > 0$ to construct convex functions which differ in the location of the knee (controlled by parameter b) and the curvature (controlled by a). We restrict the generator to convex functions based on our experiences with real-world data [6,7,10]. With this general formulation, we can only define Pareto fronts with a knee point skewed to lower values of the first objective y_1. For skewing the knee point towards lower values of y_2, the function is reflected on the angle bisector. To accomplish this, we utilize the Lambert W-function [2], which is the inverse function of xe^x. We assign this inverse function to negative values of b. A value of $b = 0$ results in a knee in the center of the front. The parameter a defines the curvature of the Pareto front, higher values of a result in a stronger severity of the knee. The effects of a and b are shown in Fig. 2. We normalize our fronts by subtracting $e^{-a} - b$ and dividing by $1 - e^{-a} + b$. After normalization, all functions of the function family intersect with the extreme points $(0, 1)$ and $(1, 0)$.

For preparing the next step of building defined sets of Pareto fronts, the parameters c and d are added to the function family. These parameters allow the Pareto fronts to be moved horizontally (c) and vertically (d). In addition, the parameter s is introduced for scaling the Pareto fronts.

The final class of functions is defined as

$$y = \begin{cases} \frac{1}{s}\left(\frac{e^{-a(x+c)} - b(x+c) - e^{-a} + b}{1 - e^{-a} + b} + d\right) & \text{if } b \geq 0 \\ \frac{1}{s}\left(\frac{1}{a|b|}\left[|b| W(u) + a(x + c - 1)(e^{-a} - |b|) - a(x + c)\right] + d\right) & \text{if } b < 0 \end{cases}$$

with W the Lambert W-function [2] and

$$u = \frac{a}{|b|} \exp\left(\frac{a}{|b|}\left[e^{-a} - b + x(b - e^{-a} + 1)\right]\right).$$

Sets of Pareto Fronts. For generating a set of Pareto fronts, we want to sample N active and M interfering functions of the function family. The N active fronts are arranged according to predefined cut points $\{t_0 = 0, t_1, \ldots, t_{N-1}, t_N = 1\}$ of the joint Pareto front. Again, we want the joint Pareto front to lie in $[0, 1]^2$ with extreme points $(0, 1)$ and $(1, 0)$.[1] The M interfering fronts benchmark the ability of the algorithm selection approach to sort out unnecessary algorithms.

[1] If desired, an a posteriori scaling to arbitrary intervals is possible.

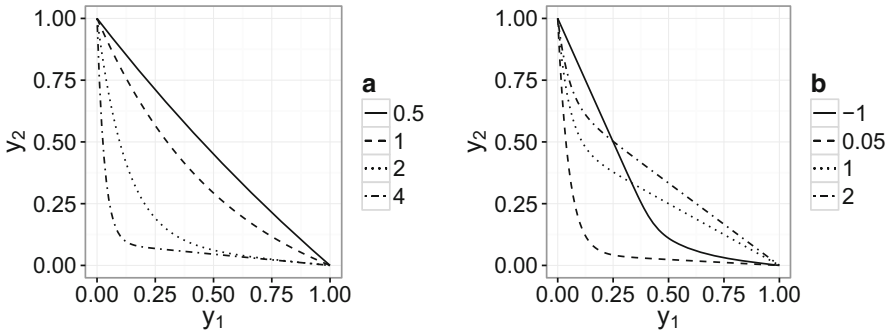

Fig. 2. Influence of the parameters a and b on the function family. In the left plot b is set to 0.1, in the right plot a is set to 20.

First the N fronts contributing to the joint Pareto front are sampled. The parameters a, b and c are drawn randomly as shown in Table 1. Due to numerical reasons the interval $[-0.05, 0.05]$ for parameter b is excluded. The value of c is slightly perturbed around the desired left cut point. For the first Pareto front, c is drawn with $\mu = 0$. This ensures that the knee of the front still lies in $(0, 1)^2$. The parameter d is automatically calculated based on the sampled values of a, b and c. It guarantees that the Pareto front intersects with the previous front or the extreme points in the predefined cut point.

Table 1. Types and parameters of the sampling distributions used in the generator.

Parameter	Distribution	Distr. parameters
$\log_2 a$	Uniform	$[-1, 5]$
b	Uniform	$[-5, -0.05] \cup [0.05, 5]$
c	Absolute	$\mu =$ left cut point t_{i-1}
	Normal	$sd = 0.05$

This procedure for generating the active Pareto fronts does not guarantee that a suitable joint Pareto front can always be constructed. To avoid infeasible data sets D, several checks are performed after each Pareto front part PF_j has been sampled. The front PF_j has to be dominated by the remaining fronts for $x \in [0, t_{j-1})$ and $x \in (t_j, 1]$. For $x \in [t_{j-1}, t_j]$, it has to be non-dominated. Furthermore, the front must not be quite similar to one of the other fronts. If one of this criteria is violated, a new value of the parameter c of the front is sampled. If no suitable front can be found after 10 samples, all its parameters are sampled again. This is done up to 100 times. If still no suitable front has been found, the next to last front is resampled, too.

In the next step, the M interfering fronts are generated. The parameters a and b are sampled according to Table 1. The parameters c and d are copied randomly from one of the active fronts and a positive noise value is added to make the function dominated by the corresponding active front. In addition, it is checked that the interfering front has no intersections with the joint front. If necessary, the parameters of the interfering front are sampled again. Up to

now, only y_1 is scaled to $[0, 1]$. In the last step, this also done for y_2 by setting $d = d - \min y_2$ and $s = |\max y_2 - \min y_2|$.

Sampling Discrete Approximations. In reality, we have to deal with discrete approximations of the true Pareto front of an algorithm A_i. We propose four methods to construct those discrete approximations from the continuously defined Pareto fronts. In the first method called *deterministic* approximation, k points are distributed with a regular spacing along the front. To accomplish this, vectors $v_i = (v_{i,1}, v_{i,2})$ are generated with $v_{i,1} = \frac{i-1}{k-1}$ and $v_{i,2} = 1 - v_{i,1}$ $(i = 1, \ldots, k)$. The respective points on the front with $v_1 y_1 = v_2 y_2$ are calculated. The second method samples the weight vectors $v_{i,1} \in [0, 1]$ randomly from a uniform distribution (*random* approximation). The last two methods are based on actual approximations of the NSGA-II [3], where we use a population size of k. To construct the respective multi-objective problem, the continuous Pareto front is plugged as shape function h into the ZDT-concept [11] $\mathrm{ZDT}(x_1, \ldots, x_l) = (x_1, g \cdot h(x)))$, where g is a function encoding the distance to the front with minimum 1. We consider an *NSGA-II* approximation, where g is fixed to its minimum value, and a *NSGA-II_g* approximation, where $g(x_2, \ldots, x_l) = 1 + \frac{9}{l-1} \sum_{i=2}^{l} x_i$ [11] is optimized for a few iterations to add a small approximation error to the front. In our experiments, we set $l = 10$ and fix the budget of the NSGA-II to 400 evaluations ($\frac{400}{k}$ generations).

Stochastic Replications. In the last step, we simulate n replications of potential tuning runs for a given joint Pareto front. We consider two practically motivated situations. In the first situation, the experiment is repeated under exactly the same circumstances. Hence, the only source of variation is the approximation quality of the tuning algorithm. This variation is simulated by adding noise to each point of the discrete approximation (*point-noise*). Following the idea of approximation error, we use absolute normal random variables with $\mu = 0$ and $\sigma = 0.02$. In the second situation, some details of the experiments change in the replications. In the context of machine learning, a different subset of learning instances may be considered during tuning [6], whereas a different rotation of the test function could be used for optimization [5]. For this situation, we create different instances by adding noise to the parameters of the Pareto fronts (*parameter-noise*). We use normally distributed random numbers with $\mu = 0$ and standard deviations according to Table 2. As a baseline method, we also consider a noiseless variant (*without-noise*), that simply replicates the discrete approximation n times. Examples of the possible combinations of approximation and replication methods are shown in Fig. 3. For reproducing our results, the generation of test data has been implemented in our MOSAP R-package[2].

Table 2. Standard deviations in the *parameter-noise* approach.

Parameter	sd
a	0.030
b	0.004
c	0.020
d	0.020

[2] https://github.com/danielhorn/multicrit_result_test.

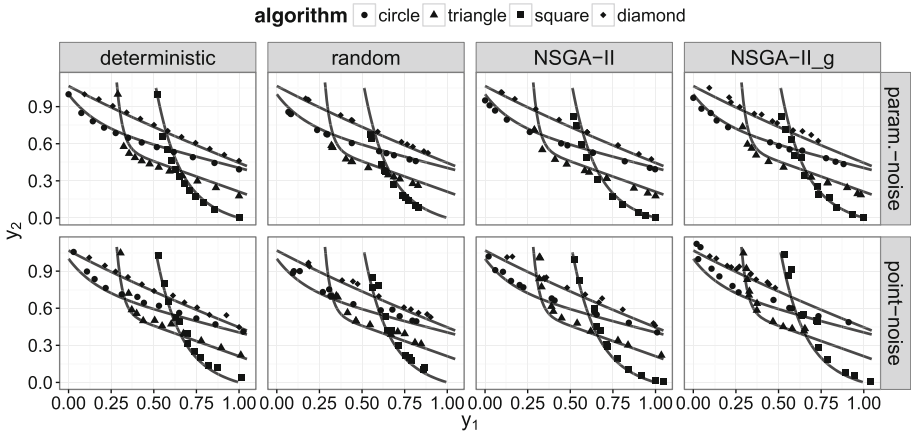

Fig. 3. Different types of generating discrete approximations and noisy replications.

4 Experimental Validation

A MOSAP method can make two types of errors: In the first type, it fails in predicting the correct subset of algorithms. A type 1 error occurs if active Pareto fronts are not selected, interfering fronts are selected, or the sequence of active fronts is swapped. The second type of error (type 2 error) is related to the accuracy of approximating the split points in the algorithm mapping.

Performance Measure. We propose an error measure that simultaneously takes both types of errors into account. Due to the construction principles of our test generator, there exists an oracle $f : [0, 1] \to \mathcal{A}$ which assigns the best algorithm A for a given value x of the performance measure y_1. Furthermore we define $\hat{f}$ as an estimator for f obtained by the MOSAP approach. The performance of $\hat{f}$ can be measured by

$$z(f, \hat{f}) = \int_0^1 \mathbb{1}(x)_{f(x)=\hat{f}(x)} dx.$$

z can be interpreted as the ratio of correct predictions of $\hat{f}$ over y_1. The optimum value is 1. In case of type 1 error, the z-value decreases by the length of the interval assigned to the wrong algorithm. In case of type 2 error, the z-value decreases by the approximation error of the split point. The integral can be easily computed in closed form because $\mathbb{1}(x)_{f(x)=\hat{f}(x)}$ is a piecewise constant function with known split points.

Benchmark. For benchmarking our MOSAP approach, we consider situations motivated by our practical applications in SVM tuning [7]. We consider $N \in \{2, 3, 5\}$ active Pareto fronts. The split points between these optimal fronts are

Table 3. Split points of the considered joint Pareto fronts.

N	2	2	3	3	5	5
Split.type	Unif.	Non-unif.	Unif.	Non-unif.	Unif.	Non-unif.
Split.points	$\{0.5\}$	$\{0.2\}$	$\{\frac{1}{3}, \frac{2}{3}\}$	$\{0.3, 0.5\}$	$\{0.2, 0.4, 0.6, 0.8\}$	$\{0.18, 0.2, 0.55, 0.75\}$

given in Table 3 and are chosen either *uniformly* (unif.) or *non-uniformly* (non-unif.). In addition, we add $M \in \{0, 2, 5\}$ interfering fronts and consider all four types of noise in creating the discrete approximation using $k \in \{4, 12, 40, 80\}$ points and all three types of stochastic replications resulting in 864 different setups with 100 replications each. For each experiment, we store the corresponding z–value. A higher error corresponds to decreasing z–values.

Hypotheses.

1. Even if a MOSAP method does not make any type 1 error, there will always be a type 2 error. This error should increase with the number of split points, the strength of the noise and decreasing quality of the coverage of the Pareto frontier (number of solutions k, spread and distribution).
2. The MOSAP-method should be able to eliminate the interfering fronts. Hence, M should not have a significant influence on the z–values.

All hypotheses are checked by means of a linear regression. The variables are coded as factors using dummy variables. The reference classes are set to: $N = 2$, $M = 0$, $k = 80$, split.type = *uniform*, discretize.type = *deterministic* and replications.type = *without-noise*. We do not report significance tests, as most observable results became significant due to the large number of replications.

5 Results

Due to space restriction we only provide the results of the linear regression in this paper, its coefficients are summarized in Table 4. The full results are available in the data-section of our MOSAP R-package.

The intercept of our model is slightly greater than 1. Hence, in the most easiest setting our method is able to reach a perfect result. With only $N = 2$ active and $M = 0$ inference fronts and no noise from both the discrete approximation and the replication, this case essentially measures how accurate the decision tree estimates the single split point.

The number of active fronts N has the largest effect. As expected, a higher number of active fronts or split points results in an increase of the error. For $N = 5$ we observe an effect size greater than 0.2. Hence, it is likely that some type 1 errors occur. This effect can be explained by the trade-off between the gap of the hypervolume and the number of algorithms which has to be found for deciding on the subset. Seemingly the gap is too small to be traded-off against the inclusion of another algorithm. This decision can be manipulated be using more extreme values of w in our method. Another option might be a human-in-the-loop, who reconsiders the parameter settings after looking at the results.

Both types of adding noise to the n replications do result in significant decreases of the z-value. This decrease is of the same level for both approaches. Compared to the noise added by the *NSGA-II_g* approach, however, the decrease of the z-values for both replication types is rather low.

Table 4. Results of the linear regression.

Variable	Value	Estimator
Intercept		1.02
N	3	−9.47e-2
N	5	−2.35e-1
Split.type	Non-uniform	2.71e-2
M	2	−3.74e-3
M	5	−7.74e-3
k	40	9.48e-4
k	12	−1.43e-2
k	4	−1.26e-1
Replications.type	Point-noise	−2.90e-2
Replications.type	Parameter-noise	−2.89e-2
Discretize.type	NSGA-II	−1.58e-2
Discretize.type	Random	−7.69e-3
Discretize.type	NSGA-II_g	−1.12e-1

The results of the discretization types are nearly as expected. The deterministic type is the easiest for the MOSAP approach. As expected, *NSGA-II_g* combining both approximation error and a non-uniform distribution results in a significant loss of performance. Surprisingly, *random* point sets result in better z-values than the ones of *NSGA-II* without approximation error. In fact, the *random* point sets are only slightly inferior to the deterministic ones. Hence, there is no need for perfectly spaced Pareto front approximations. This observation is confirmed by the effect of the approximation size. The reduction from $k = 80$ to $k = 40$ points even has a very small, positive effect, actually it is the only effect without a significant influence. Nevertheless, a further reduction to $k = 12$ or $k = 4$ points results in a either a slight or strong decrease of the z-value. Hence, k should not be set too small, but it is unnecessary to use very large discrete approximations. In conclusion, we can confirm our first hypothesis.

An increase of the number of interfering fronts M results in decreases of the z-value in the order of 10^{-3}. Compared to the effects of approximation error (*NSGA-II_g*) or additional active fronts ($N = 5$), these effects are small, but they indicate that an increasing amount of interfering fronts may have a negative effect on the result. Therefore, we can only partially confirm our second hypothesis.

As additional observation, the *non-uniform* cut points result in better z-values than the *uniform* ones. This seems meaningful, since some of the active fronts cover only a small portion of the joint front. Hence, type 1 errors will result in a smaller decrease of the z-value.

6 Conclusion and Outlook

In this paper, we present a validation framework for MOSAP methods. We applied the framework to evaluate to performance of our approach. As expected, the performance slightly decreases with an increasing number of active fronts and noisy approximation sets. Nevertheless, our method is capable of finding suitable portfolios and mappings even in the hardest cases considered.

In future work we are going to apply our MOSAP method to more practical test cases. One possibility would be to derive algorithm portfolios from the results of the Black-Box Optimization Benchmarking workshop (BBOB) [5]. In this

workshop, the objectives are the number of function evaluations and the ratio of target levels attained over a set of functions. In this context, it would also be interesting whether the Pareto fronts can be merged over different instances instead of only replications on the same instance.

References

1. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
2. Corless, R.M., Gonnet, G.H., Hare, D.E.G., Jeffrey, D.J., Knuth, D.E.: On the LambertW function. Adv. Comput. Math. **5**, 329–359 (1996)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
4. da Fonseca, V.G., Fonseca, C.M.: The attainment-function approach to stochastic multiobjective optimizer assessment and comparison. In: Bartz-Beielstein, T., et al. (eds.) Experimental Methods for the Analysis of Optimization Algorithms, pp. 103–130. Springer, Heidelberg (2010)
5. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2010: experimental setup. Technical report RR-7215, INRIA (2010)
6. Horn, D., Bischl, B., Demircioglu, A., Glasmachers, T., Wagner, T., Weihs, C.: Multi-objective selection of algorithm portfolios. Archives of Data Science (2016, under revision)
7. Horn, D., Demircioglu, A., Bischl, B., Glasmachers, T., Weihs, C.: A comparative study on large scale kernelized support vector machines. Advances in Data Analysis and Classification, S.I.: Science of Big Data: Theory, Methods and Applications (2016, under revision)
8. Hutter, F., Bartz-Beielstein, T., Hoos, H.H., Leyton-Brown, K., Murphy, K.: Sequential model-based parameter optimisation: an experimental investigation of automated and interactive approaches. In: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuß, M. (eds.) Empirical Methods for the Analysis of Optimization Algorithms, pp. 361–411. Springer, Heidelberg (2010)
9. Miettinen, K.: Nonlinear Multiobjective Optimization. International Series in Operations Research and Management Science, vol. 12, 4th edn. Kluwer Academic, Dordrecht (2004)
10. Weinert, K., Zabel, A., Kersting, P., Michelitsch, T., Wagner, T.: On the use of problem-specific candidate generators for the hybrid optimization of multiobjective production engineering problems. Evol. Comput. **17**(4), 527–544 (2009)
11. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. Evol. Comput. **8**(2), 173–195 (2000)
12. Zitzler, Eckart, Thiele, Lothar: Multiobjective optimization using evolutionary algorithms - a comparative case study. In: Eiben, Agoston E., Bäck, Thomas, Schoenauer, Marc, Schwefel, Hans-Paul (eds.) PPSN 1998. LNCS, vol. 1498, p. 292. Springer, Heidelberg (1998)