J Glob Optim manuscript No.
(will be inserted by the editor)

Dynamic Algorithm Selection for Pareto Optimal Set Approximation

Ingrida Steponavičė · Rob J Hyndman · Kate Smith-Miles · Laura Villanova

Received: date / Accepted: date

Abstract This paper presents a meta-algorithm for approximating the Pareto optimal set of costly black-box multiobjective optimization problems given a limited number of objective function evaluations. The key idea is to switch among different algorithms during the optimization search based on the predicted performance of each algorithm at the time. Algorithm performance is modeled using a machine learning technique based on the available information. The predicted best algorithm is then selected to run for a limited number of evaluations. The proposed approach is tested on several benchmark problems and the results are compared against those obtained using any one of the candidate algorithms alone.

Keywords multiobjective optimization · expensive black-box function · machine learning · classification · algorithm selection · hypervolume metric · features

1 Introduction

Many optimization problems involve multiple conflicting objectives, where there is no single optimal solution optimizing all objective functions simultaneously, but a set

Ingrida Steponavičė
School of Mathematical Sciences, Monash University, Clayton, Australia
Tel.: +61 3 9905 8511
E-mail: ingrida.steponavice@monash.edu

Rob J Hyndman
Department of Econometrics & Business Statistics, Monash University, Clayton, Australia

Laura Villanova
School of Mathematical Sciences, Monash University, Clayton, Australia

Kate Smith-Miles
School of Mathematical Sciences, Monash University, Clayton, Australia

of solutions representing the best possible trade-offs among the objectives. Therefore, multiobjective optimization is a very important research area due to the multiobjective nature of most real-life problems, with many challenging issues to tackle.

The development of multiobjective optimization techniques has been an active area of research for many years, resulting in a wide variety of approaches [4,23,24]. Besides the challenge caused by multiple objectives, practical problems arising in engineering often require the solution of optimization problems where analytical expressions of the objective functions are unavailable and the evaluation of the objective functions are very expensive. Such problems might involve computationally expensive black-box simulation, or require costly experiments to be conducted in order to obtain the objective function values. One simulation or experiment may take several hours, days or even weeks. In addition to time restrictions, there can be other limitations such as financial and physical constraints. Therefore, in order to keep the cost affordable, it is important to find approximate solutions of the optimization problem within a very restricted number of function evaluations (often only a few hundred evaluations can be made).

Methods have been developed to solve expensive black-box optimization (BBO) problems by building a surrogate model that approximates the objective function and predicts promising new solutions at a smaller evaluation cost [14,31]. One of the state-of-art methods for expensive multiobjective optimization problems, named ParEGO, was developed by Knowles [16]. It is essentially a multiobjective translation of the efficient global optimization (EGO) method [14], where multiple objectives are converted to a single objective using a scalarization function with different parameter values at each step. The idea of modelling challenging functions by statistical models has a very long history, and was popularized for optimization problems in [25, 34]. Other EGO modifications to address costly multiobjective optimization problems are also available, including SMS-EGO [29], ϵ-EGO [37], MOEA/D-EGO [41], and EGO-MO [7].

In addition to the EGO family of algorithms, we have previously proposed the EPIC (Efficient Pareto Iterative Classification) algorithm [32]. In this approach, the Pareto optimal set is identified by classifying regions of the decision space as likely to be part of the Pareto set or not. A support-vector-machine (SVM) is applied in order to capture nonlinear relationships between class labels and features (i.e., decision variable values in this case). The advantage of this approach is that it does not depend on the dimensionality of the objective space and so is suitable for high-dimensional multiobjective problems.

Other approaches are also possible. It is an open question how to best select the optimization algorithm for the particular problem of interest. Comparisons of various methods for expensive multiobjective black-box optimization are prospering in the literature, but they are usually limited and highlight the advantages of some proposed modification to an existing method over its predecessor. There is a need for a deeper analysis of the characteristics of the available algorithms and how well-suited they are to specific problems.

To our knowledge, the methods developed so far each have some strengths and weaknesses, and despite advances made in recent years, they are still far from being able to solve a variety of real-life problems efficiently. Often, they are better suited to some restricted problem classes. Moreover, for the same problem, some algorithms can perform very well at the beginning and then lose their power, while other algorithms can perform badly at the beginning but later demonstrate their superiority. Such an example is demonstrated in Figure 1 which presents the percentage of times when one of the considered algorithms (ParEGO, EPIC and a hybrid of EPIC and Nelder-Mead) was outperforming others with respect to the hypervolume (HV) metric (see Section 3.3) on a benchmark problem ZDT3 over 100 runs; the horizontal axis represents the number of objective function evaluations (or iterations). This figure clearly shows that there is no single algorithm (at least among those considered) performing better than the rest for *all runs* and at *all points in time*. For example, if one can afford more than 70 evaluations, one should use ParEGO; in case of fewer than 20 affordable evaluations, one should run EPIC-NM. As this figure suggests, one might think that we can obtain good results by running EPIC-NM for the first 20 iterations, then EPIC for the next 50 iterations, and then ParEGO for the remaining iterations. However, algorithm performance depends on the decision space already explored. If we switch algorithms, then we would also have a different historical exploration of the decision space, so the performance may not match that presented in Figure 1. Thus, there is no guarantee that the results obtained using this simple idea will outperform the algorithms running separately.

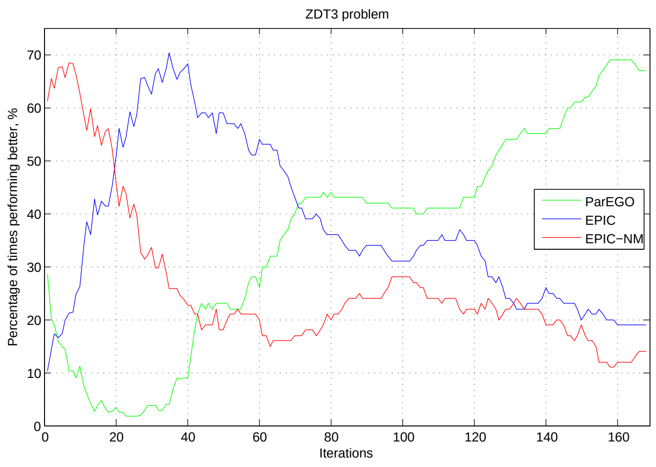

Fig. 1 Algorithms performance on ZDT3 problem

Therefore, we are interested in learning how to select the right algorithm at each stage of the optimization process when very little is known about the multiobjective optimization problem in advance. In particular, we focus on expensive black-box problems where we wish to limit the number of function evaluations.

The paper has the following structure. Section 2 introduces the main concepts involved in multiobjective optimization, and our proposed approach is described in Section 3. In Section 4, we outline our experimental setup and the selected algorithms, and present and analyse the results we have obtained on some test problems.

Section 5 draws some conclusions and briefly discusses some future research directions.

This is an initial exploration of an approach to this problem, describing how it can be implemented, and highlighting and discussing the results obtained on a small number of optimization problems. Much larger computational experiments involving many more optimization problems would be required in order to draw general conclusions, and validate the proposed approach. This would take a vast amount of time and so is left for future research.

2 Expensive Multiobjective Optimization Problems

The problems considered here are formulated as follows:

$$\min \boldsymbol{f}(\boldsymbol{x}) = \big(f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})\big)^T \qquad \text{subject to } \boldsymbol{x} \in S, \tag{1}$$

where $S \subset \mathbb{R}^n$ is the feasible set and $f_i :\to \mathbb{R}$, $i = 1, \ldots, m$ ($m \geq 2$), are expensive black-box functions that are to be minimized simultaneously. All objective functions are represented by the vector-valued function $\boldsymbol{f} : S \to \mathbb{R}^m$. A vector $\boldsymbol{x} \in S$ is called a *decision vector* and a vector $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^m$ an *objective vector*.

In multiobjective optimization, the objective functions $f_1, \ldots, f_m$ in (1) are typically conflicting. In that case, there does not exist a decision vector $\bar{\boldsymbol{x}} \in S$ such that $\bar{\boldsymbol{x}}$ minimizes f_i in S for all $i = 1, \ldots, m$, but there exists a number (possibly infinite) of Pareto optimal solutions. In mathematical terms, a decision vector $\bar{\boldsymbol{x}} \in S$ and its image $\bar{\boldsymbol{z}} = \boldsymbol{f}(\bar{\boldsymbol{x}})$ are said to be *Pareto optimal* or *non-dominated* if there does not exist a decision vector $\boldsymbol{x} \in S$ such that $f_i(\boldsymbol{x}) \leq f_i(\bar{\boldsymbol{x}})$ for all $i = 1, \ldots, m$ and $f_j(\boldsymbol{x}) < f_j(\bar{\boldsymbol{x}})$ for some $j = 1, \ldots, m$. If such a decision $\boldsymbol{x} \in S$ does exist, $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{z}}$ are said to be *dominated* by $\boldsymbol{x}$ and its image $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x})$, respectively.

3 Dynamic Algorithm Selection

3.1 General Framework

There are many different algorithms that perform well on some problem classes and struggle on others, and it is difficult to predict the accurate performance of an algorithm on a particular problem. In practice, the number of function evaluations required by the candidate algorithm to solve some particular problem can be vast. Having a particular problem to be solved, especially in a limited number of function evaluations, one must select an algorithm without being sure of making the most appropriate choice. Bad decisions may lead to an unacceptable number of function evaluations and poor approximation of the true Pareto set. Algorithm selection is a learning problem where we use a model to predict the expected performance of each algorithm on a given problem; the model is trained on a set of performance data for a number of problems [30]. For each new problem, the model is used to select the algorithm that is expected to give the best results. In addition to static approaches where the selection is performed before running the algorithms, there have been proposed a number of dynamic algorithm selection approaches where the selection process is adapted during the actual execution of the algorithms (e.g., see [2,21,15]).

Here, we suggest switching among different algorithms during the search, based on the information collected in the objective and decision spaces. For this purpose, we use a model that predicts which algorithm will perform the best in a given situation according to a selected performance metric. In multiobjective optimization, algorithm performance can be assessed taking into account different qualities of the estimated Pareto optimal set such as spread, convergence, distribution, etc. The choice of metrics to use in evaluating algorithm performance is somewhat subjective.

The basic idea of dynamic algorithm selection tends to circumvent the following challenges which are associated with expensive multiobjective black-box optimization problems: (i) selecting the 'right' algorithm to solve the problem with very little (or no) knowledge about it; and (ii) obtaining a high quality approximation of the Pareto optimal set within a limited number of evaluations.

Algorithm performance prediction can be modelled as a classification problem in which each new instance is assigned to the algorithm class that would lead to the greatest improvement in the performance metric, based on models trained on instances whose class membership is known. Therefore, the proposed approach can be divided into two stages.

In Stage A, we collect the training data and use them to build the performance prediction model(s). That is, a classification algorithm is used to learn the relationship between the descriptive metrics of a current situation and subsequent algorithm performance over the next few evaluations. This is based on a large dataset where all considered algorithms have been applied to a large number of problems at various points in time and their performance has been monitored.

Stage B employs the prediction model to approximate the Pareto optimal set and can be decomposed into the following steps:

Step B1 Generate an initial set in the decision space and evaluate the objective functions;
Step B2 Given some evaluated vectors, calculate *descriptive metrics*;
Step B3 Ask the *prediction model* to predict the best algorithm based on the calculated metrics;
Step B4 Run the *suggested algorithm* for a limited number of evaluations;
Step B5 Stop if the maximum number of function evaluations is reached. Otherwise, go to B2.

Stage B is represented in Figure 2. The most important elements are the prediction model and descriptive metrics which at the very beginning are calculated from the initial set of evaluated solutions. After running a suggested algorithm for a small number of iterations, the solution set is updated. Metrics are updated and the same steps are repeated until the maximum number of evaluations is reached.

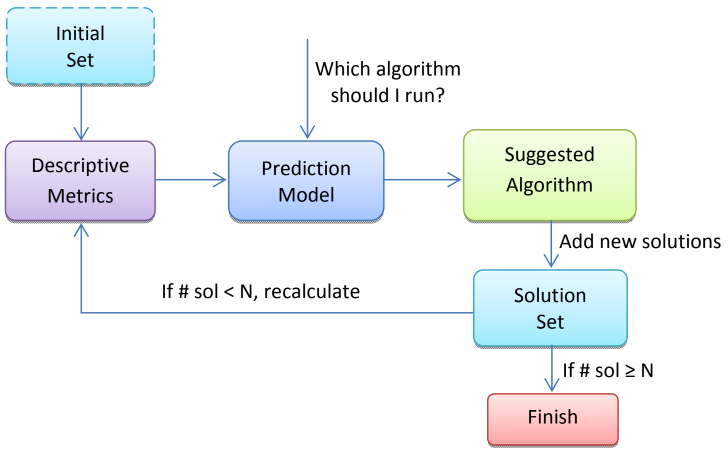

Fig. 2 Stage B representation

3.2 Performance prediction model

The purpose of a performance prediction model is to help select an algorithm with the best performance in a given situation. Therefore, it has to give an answer to the question "Which algorithm will perform best in the current situation: X, Y or Z?"

In Stage A, a set of instances with features and the class to which each of the instances belongs (i.e., the name of the algorithm which give the best performance for each instance) is given to the machine learning algorithm (e.g., a random forest or support vector machine) to learn the relationship between features and class labels. The learning algorithm utilizes the label information as well as the data itself to learn a mapping function g (or a classifier) from features to class labels: $g(features) \rightarrow labels$. To build a performance prediction model we need to have metrics describing different situations used as features associated with a class label, and the name of the algorithm whose performance was the best under given circumstances.

In Stage B, instances are represented by the same features as used in training and then the performance prediction model assigns labels; i.e., it predicts which of the algorithms will perform best in each instance.

Building a reliable performance prediction model that is applicable to various expensive multiobjective optimization problems requires as much data as possible resulting in a time-consuming process of running algorithms on a large and diverse set of test problems. However, collecting data is a very important step to ensure an efficient performance of the proposed approach. It is also very important to select the right features or descriptive metrics; we discuss these in the following subsection.

3.3 Descriptive metrics

To build a classifier, we need to have knowledge of what features make good predictors of class membership for the algorithms we are considering. In real-world situations, we often have little knowledge about relevant features. Therefore, we have to find a set of features that separates classes as cleanly as possible.

It is clear that the selected features must have some relationship with the performance of the algorithms. Usually, in the published comparisons of different multi-objective optimization algorithms, we can see only numerical experiments and discussion about the algorithm performance without deeper analysis about why some algorithms work well on the problems, why some algorithms have some difficulties, and what are the characteristics that make them succeed or fail.

Intuitively, the metrics characterizing the observed Pareto set and the progress made in searching for non-dominated solutions should be important. There exist many metrics used to assess the quality of the obtained solution set in multiobjective optimization that can be categorized into cardinality (or capacity), convergence, diversity and hybrid measures [12]. Most of these metrics were developed to compare an obtained solution set with the true Pareto optimal set, and include generational distance [5], inverted generational distance [35], ϵ-indicator [45] and hypervolume difference [38] among others. However, these are not suitable for our purpose as in practice the true Pareto set is not known a priori. Hence, this significantly reduces our choice.

We now describe the quality metrics we have considered.

Ratio of non-dominated solutions. This metric is a capacity measure quantifying the ratio of non-dominated solutions in the obtained solution set:

$$\mathrm{RON}(S,P) = \frac{|P|}{|S|}, \tag{2}$$

where $|S|$ is number of the solutions in the observed solution set and $|P|$ is the number of non-dominated solutions in the observed Pareto set P.

Generalized Spread metric [6]. This diversity metric indicates the distribution of solutions in the observed Pareto set P:

$$\Delta^*(P,T) = \frac{\sum_{i=1}^{m} d(e_i,P) + \sum_{X\in P} |d(X,P) - \bar{d}|}{\sum_{i=1}^{m} d(e_i,P) + |P| * \bar{d}}, \tag{3}$$

where $(e_1,\dots,e_m)$ are m extreme solutions in T, the *true* Pareto optimal set, and

$$d(X,P) = \min_{Y\in P, Y\neq X} ||\boldsymbol{f}(X) - \boldsymbol{f}(Y)||^2,$$

$$\bar{d} = \frac{1}{|P|} \sum_{X\in P} d(X,P).$$

Smaller values are preferable. This metric requires knowledge of extreme values of T. When solving real-word problems, this information is not available beforehand. Therefore, one can use some estimates of the extreme values of the objective space.

Number of distinct choices [38]. This metric divides the objective space into a grid of $(1/\mu)^m$ m-dimensional hypercubes ($\mu \in [0,1]$) and calculates the number of hypercubes containing solutions; i.e., it indicates the number of distinct solutions that exists in an observed Pareto solution set P:

$$\mathrm{NDC}_\mu(P) = \sum_{\ell_m=0}^{\nu-1} \cdots \sum_{\ell_2=0}^{\nu-1} \sum_{\ell_1=0}^{\nu-1} N_\mu(q,P) \tag{4}$$

where $q = (q_1, q_2, \ldots, q_m)$ with $q_i = \frac{\ell_i}{\nu}$, $\nu = \frac{1}{\mu}$ and $N_\mu(q,S)$ is equal to 1 provided there is at least one non-dominated solution falling into the hypercube $T_\mu(q)$; otherwise, it is equal to 0. Solutions falling into the same hypercube are considered similar to one another. The idea is to count only those generated non-dominated points that are sufficiently distinct from each other. The higher value is preferred over the smaller for diversity.

Dispersion. This metric was introduced in [38] where it was named 'cluster'; it describes the average size of clusters composed of similar non-dominated solutions in P, and is calculated as follows:

$$\mathrm{CL}_\mu(P) = \frac{|P|}{\mathrm{NDC}_\mu(P)}, \tag{5}$$

In the ideal case where every non-dominated solution obtained is distinct, then the value of metric $\mathrm{CL}_\mu(P)$ is equal to 1. Also, the higher the value of metric $\mathrm{CL}_\mu(P)$, the more clustered the non-dominated solution set P, and hence the less preferred. In our opinion, the term 'cluster' is misleading, therefore, we have named it 'dispersion'.

Correct classification. This is the percentage of correctly classified non-dominated and dominated solutions obtained by a support vector machine (SVM). This metric was selected because some of the considered algorithms use an SVM. Thus, the quality of an SVM at a given point in the search is a useful descriptor of the current situation and how an algorithm relying on SVM modelling is likely to perform.

Correct classification of non-dominated class. This is the percentage of correctly classified non-dominated solutions. As the non-dominated class is usually (significantly) smaller than the dominated one, total correct classification may still be good while all examples from non-dominated class can be misclassified.

Correct classification of dominated class. This is the percentage of dominated solutions correctly classified by an SVM.

Hypervolume metric [44]. This metric has attracted a lot of interest in recent years as it describes both the convergence towards the Pareto optimal set and the distribution along it. Basically it calculates the volume covered by non-dominated solutions (see Figure 3). Mathematically, for each solution $i \in P$, a hypercube v_i is constructed with a reference point W and the solution i as the diagonal corners of the hypercube. The reference point can simply be obtained by composing a vector of the worst objective function values. Then, a union of all hypercubes is found and its hypervolume is calculated:

$$\text{HV}(P, W) = \text{volume}(\bigcup_{i=1}^{|P|} v_i). \tag{6}$$

One of the main reasons for the popularity of HV is that it not only reflects dominance, but also promotes diverse sets. Moreover, it is the only indicator known to be strictly monotonic with respect to Pareto dominance and thereby guaranteeing that the Pareto optimal set achieves the maximum hypervolume possible, while any worse set will be assigned a worse indicator value [1].

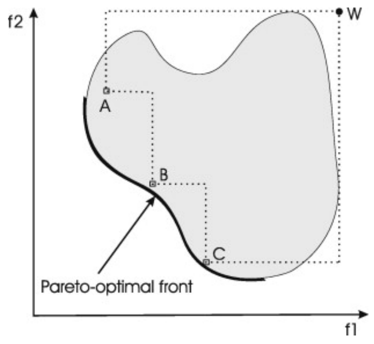

Fig. 3 HV metric calculation [12]

Despite the attractive features of HV, it has a few major issues. First, it is computationally intensive, especially for high dimensional problems. Second, the metric varies with the choice of the reference point [42]. Finally, if the scales of the objective functions are very different, it can be biased in favour of objectives with a larger scale. To eliminate the bias of different scales, it is suggested [4] to calculate the HV metric using normalized objective function values.

We use the HV metric for two purposes: first, as one of the features to characterize the current situation; and second, to assess algorithm performance or superiority to derive class membership.

All the descriptive metrics that depend on the scale of the objective functions should be calculated using normalized (scaled) objective function values in order to eliminate any bias in the metric values. Therefore, we estimated the extreme values of the objective space and used this information for normalization.

4 Experimental Analysis

4.1 Selected algorithms

To test the proposed approach, we switch among three algorithms: ParEGO, EPIC, and Nelder-Mead (NM). The performance of our dynamic switching algorithm was compared with ParEGO, EPIC and EPIC-NM algorithms running separately. They are shortly discussed below.

ParEGO This method employs a Gaussian process (GP) model to predict objective function values. It converts the multiobjective optimization problem into a single objective problem using the augmented Tchebycheff function:

$$f_\lambda(x) = \max_{j=1,\ldots,m} \big(\lambda_j f_j(x)\big) \pm \rho \sum_{j=1}^{m} \lambda_j f_j(x), \tag{7}$$

where $\rho > 0$ is a small positive number and λ is a weight vector. At each iteration of the algorithm, a different weight vector is drawn uniformly at random from the set of evenly distributed vectors allowing the model to gradually build up an approximation to the true Pareto set. Before scalarization, the objective functions are normalized with respect to the known (or estimated) limits of the objective space to the range $[0, 1]$. At each iteration, the method uses a genetic algorithm to search for the solutions that maximizes the expected improvement criterion with respect to a surrogate model. After evaluation of the selected solution on the real expensive function, ParEGO updates the GP surrogate model of the landscape and repeats the same steps.

The main disadvantage of employing a GP is that model construction can be a very time-consuming process [13], where the time increases with the number of evaluated vectors used to model the GP. To overcome this issue, when the iteration number is greater or equal to 25, ParEGO uses a subset of the evaluated vectors to build the GP model, thus attempting to balance model accuracy and computation time. Moreover, using a GP becomes increasingly problematic in high dimensional spaces [8], so these methods do not scale well as the dimension of the problem increases.

EPIC The EPIC algorithm approximates the Pareto optimal set with a limited number of objective function evaluations. Its main idea is to learn about the evaluated non-dominated and dominated vectors in the decision space and to predict which unevaluated vectors are likely to be non-dominated, thus gradually building an approximation of the Pareto optimal set by evaluating the most promising decision vectors. A discussion of how to select vectors for evaluation can be found in [32].

A major advantage of this method is that it does not use any statistical model of the objective function, such as GP, and so it involves more modest computational requirements, and scales easily to handle high dimensional spaces. Moreover, it is simple to implement, has no limitations on high dimensional problems, and multiple decision vectors can be selected at each iteration [32]. However, its weakness is that it does not generate new decision vectors but rather selects a decision vector from a given set representing the decision space, whose quality has an impact on the method performance. To overcome this issue, the EPIC method can be modified by introducing a mechanism for generating new decision vectors in the most promising areas of the decision space.

Nelder-Mead algorithm for multiobjective optimization The NM method was developed for single objective optimization problems as a very efficient derivative-free local search procedure [26]. A disadvantage of NM is that its convergence is sensitive to the selected starting point and can fail because the search direction becomes increasingly orthogonal to the direction of steepest descent [33]. Also, it can become inefficient for large dimensional problems [11,28]. Despite lacking a satisfactory convergence theory, the NM method generally works well on small dimensional real-life problems and remains one of the most popular direct search methods [20,18]. Recently, numerous improvements to the Nelder–Mead simplex algorithm have been posed to address these issues [22,39,28,9]. For multiobjective optimization problems, several NM modifications or hybridization have been proposed as well [17, 40].

To apply NM for multiobjective optimization, we scalarize a multiobjective problem to a single objective by using the augmented Tchebycheff function (7). An initial simplex is constructed from the $n + 1$ vertices having the best scalarized objective function values. Then the main simplex transformation operations (reflexion, expansion, contraction and shrinking) are performed as in the original algorithm for single objective problems. At every iteration a new starting simplex is selected using a different weighting vector to encourage exploration of the Pareto front.

EPIC-NM algorithm We have modified the EPIC algorithm to enable generation of the decision vectors by hybridizing it with NM for a local search in the promising areas. Here, NM is called when there is no significant improvement measured by the HV metric using the EPIC algorithm and run for a limited number of evaluations to look for improvement locally. Different weighting vectors are used to convert the original problem to a scalarized one for the NM algorithm. The starting simplex is composed of the best vectors evaluated so far in an appropriate scalarized objective space. Thereafter, NM continues with regular simplex iterations for a few evaluations. Then, the search is taken over by EPIC fed with an updated set of evaluated solutions. For switching, EPIC-NM does not use model-based decisions but a rule determined by the metric of the current state.

4.2 Experimental setup

All algorithms were implemented in Matlab and their parameters were set to default values. In particular, our ParEGO implementation was based on the C code by Knowles, which can be downloaded from `www.cs.bham.ac.uk/~jdk/parego/`; this implementation corresponds to the algorithm described in [16]. The default values for the ParEGO implementation were used, namely (i) population size equal to 20, (ii) number of restarts when optimising the likelihood function is equal to 30, and (iii) crossover is equal to 0.2. The implementation of EPIC is described in [32]. In both EPIC and EPIC-NM, we used SVM with a radial basis function kernel; SVM kernel parameters were obtained through cross-validation performed at each iteration. In EPIC-NM, an initial simplex was composed of the vertices having the best scalarized problem values. A local search was called after EPIC could not make progress (i.e., no change occurred in HV metric values for the last four iterations), and run for five evaluations. All algorithms started with the same initial set consisting of $11n - 1$ decision vectors, where n is the dimension of the decision space, as suggested in [14]. The Latin hypercube technique was used to sample the decision space. In addition, for EPIC and EPIC-NM, we sampled a design space representation consisting of 500 vectors although the objective function values for these points were not evaluated unless selected by an algorithm. The maximum number of evaluations was restricted to 200 including the initial sampling. The algorithms were run 100 times with different initial sets (as their performance is influenced by the initial set), and the average values of the HV metric were calculated. The performance of the algorithms was measured at every iteration to assess the progress obtained after each objective function evaluation.

4.3 Test problems

Our training set consisted of more than 12000 instances (snapshots in time) from solving the following four benchmark problems: ZDT3 [43], OKA2 [27], Kursawe [19] and Viennet [36]. These presented different challenges for approximating the true Pareto optimal set.

ZDT3. This problem has two objective functions and three decision variables. The Pareto optimal set comprises several discontinuous convex parts in the objective space.

Kursawe. This problem has two objective functions and a scalable number of decision variables. In our experiment, three decision variables were used. Its Pareto optimal set is disconnected and symmetric in the decision space, and disconnected and concave in the objective space.

OKA2. This problem has two objective functions and three decision variables. Its true Pareto optimal set is a spiral shaped curve in the objective space, and the density of the Pareto optimal solutions in the objective space is low.

Viennet. This problem consists of three objective functions and two decision variables. Its true Pareto optimal set is convex in the objective space.

4.4 Building the performance prediction model

There are many classification methods available including linear classifiers, support vector machines, decision trees and neural networks. To model our algorithm performance we have used a random forest [3], which is an ensemble of randomly trained decision trees. Algorithm performance was assessed with the HV metric calculated using normalized objective function values. Each instance was associated with the name of the algorithm that had the largest value of the HV metric. The distribution of instances among the three classes was as follows: 4599 instances in the ParEGO class, 5890 in the EPIC class and 2930 in the NM class. Hence, the largest class consists of instances where EPIC was best, while NM's was the smallest.

Table 1 Confusion matrix of algorithm performance model

	predicted EPIC	predicted NM	predicted ParEGO
actual EPIC	5014	230	646
actual NM	374	1618	938
actual ParEGO	1108	715	2776

When building a model for a classification problem, it is always important to look at the classification accuracy; i.e., the number of correct predictions from all predictions made. Out-of-bag accuracy of the trained model was 70.11%. A clear way to present the prediction results of a classifier is to use a confusion matrix, which shows how the predictions are made by the model. The diagonal elements show the number of correct classifications made for each class, and the off-diagonal elements show the errors made. The confusion matrix of the prediction model is presented in Table 1. It shows that the model tends to predict the EPIC algorithm will be best more often and misclassifies more instances from the NM and ParEGO classes.

The feature space of the prediction model gives some insight about the values of the metrics which are favorable for each algorithm. For example, Figure 4 shows that the EPIC algorithm was best in more than 90% of instances where the HV values were between 0.6 and 0.65. When the ratio of non-dominated solutions is small, ParEGO or NM are preferred algorithms as seen in Figure 5. This can be simply explained by the fact that the SVM used in the EPIC algorithm has limited information about the non-dominated solution class when there are few non-dominated solutions, resulting in poor EPIC performance. Figure 6 shows that EPIC prefers higher NDC values over smaller ones; i.e., it needs more distinct evaluated solutions to work well.

The random forest model we used to select the preferred algorithm revealed that the most important feature for classification prediction is the hypervolume metric followed by generalized spread, dispersion, correct classification of dominated points, overall classification accuracy and the ratio of nondominated solutions. On the other hand, the least important features for algorithm prediction are the number of distinct choices and the correct classification of nondominated points.

The accuracy of the prediction model can be affected by several factors. First, a model may be based on a training set where classes were unbalanced. Therefore, the model assigned more instances to the majority class and made more errors for smaller classes. This can be overcome by collecting more examples for smaller classes, or by using different accuracy measures such as precision and recall. Second, selected descriptive metrics may not completely describe the situation at that time. Looking for more and/or better descriptive metrics can be a solution.

A further consideration is where one or more algorithms may be equally good, or approximately the same, in their performance. In that case, finding the best algorithm is not as important as avoiding a poor algorithm. For that reason, misclassifications will not necessarily lead to poor performance in the main task of estimating the Pareto front. Table 2 shows that our prediction model suggested the best algorithm in 70.11% of all cases, while the worst one in only 9.78% of all cases. Thus, our proposed approach can minimize the risk of selecting the poorest algorithm.

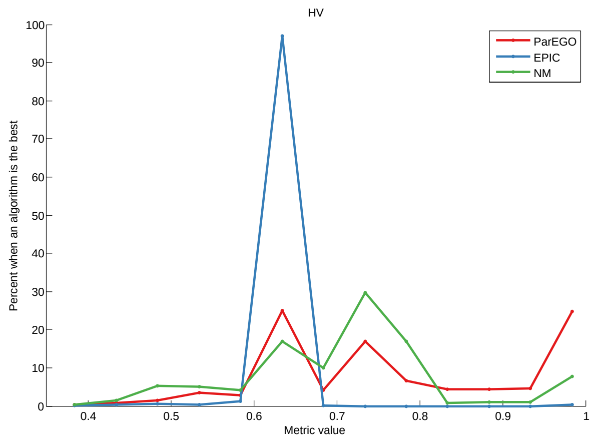

Fig. 4 Percentage distribution of HV values when each algorithm was the best

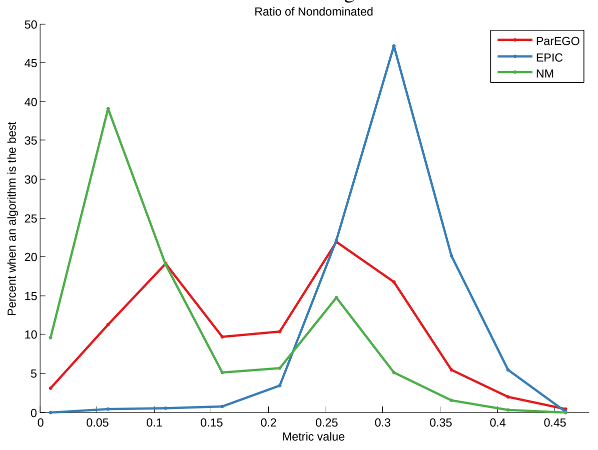

Fig. 5 Percentage distribution of RoN values when each algorithm was the best

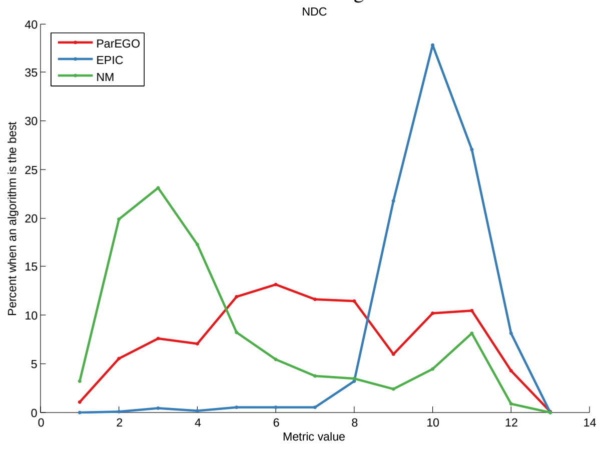

Fig. 6 Percentage distribution of NDC values when each algorithm was the best

Table 2 Distribution of suggested algorithms by the performance model

Best algorithm	Second best algorithm	Worst algorithm
70.11%	20.11%	9.78%

4.5 Numerical results

A number of computational experiments were carried out to test the ability of the proposed approach to approximate the Pareto optimal set with a limited number of evaluations. The performance of our proposed dynamic switching algorithm is demonstrated in Figure 7. Here, it dynamically switches among three algorithms at every fifth evaluation using model predictions to decide which algorithm to use and these moments are marked by dots. Also, its performance is compared with the performance of three algorithms run separately.

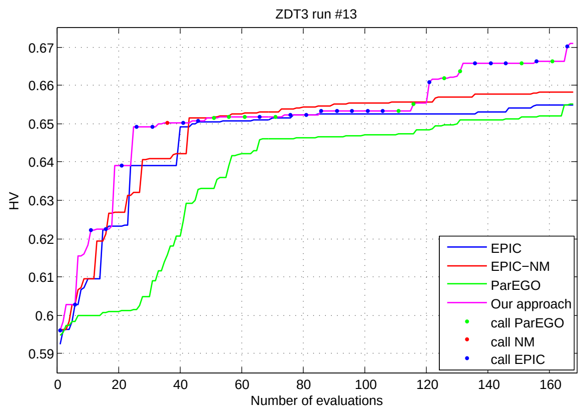

Fig. 7 Performance on a single run of ZDT3 problem

The comparison, based on the average HV metric over 100 runs and calculated using normalized and original values of ZDT3 and Kursawe, is presented in Figures 8–13. The figures depict the average HV measured after the initial sampling (i.e., starting from the $11n$th function evaluation). The initial sampling does not provide relevant information for algorithm comparisons because, for each of the 100 runs, all the algorithms have been evaluated on the same initial sample. Results similar to those reported in Figures 8–13 were obtained for OKA2 and Viennet problems. They also show that the proposed approach is competitive. It can be noted that for the ZDT3 problem, algorithm superiority depends on how the HV metric is calculated. For example, Figure 10 demonstrates that the proposed approach is the most efficient with respect to the HV metric calculated using the original scale while Figure 8 does not provide a clear winner.

This raises the question of which metric should be used to judge the algorithm performance. If the objectives have different scales and we aim to find a uniformly

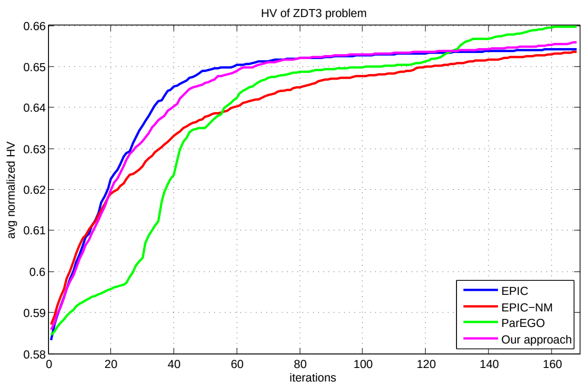

Fig. 8 ZDT3: average HV (normalized scale)

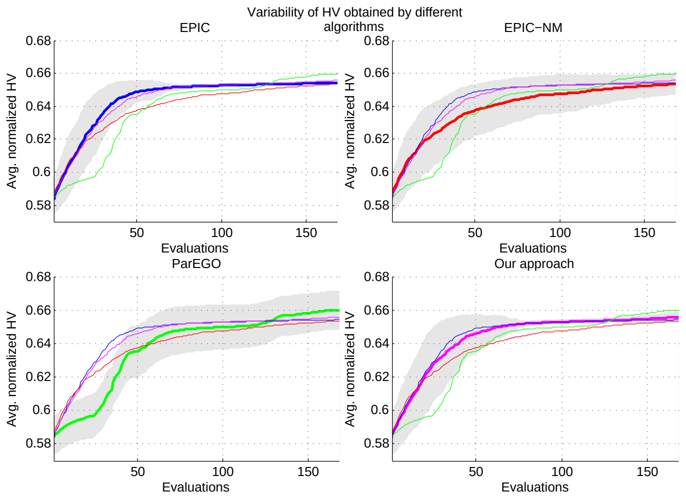

Fig. 9 ZDT3: average and standard deviations of HV (normalized scale)

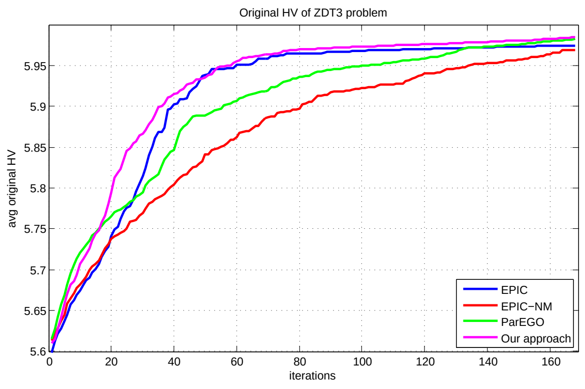

Fig. 10 ZDT3: average HV (original scale)

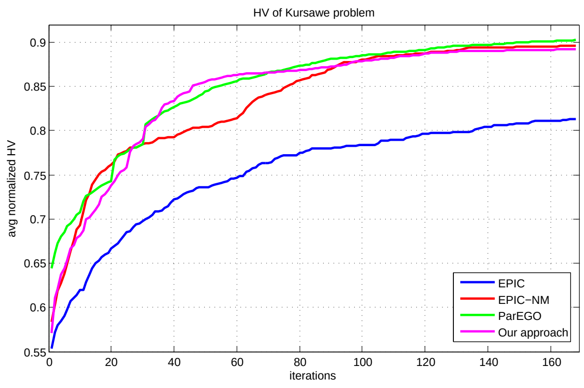

Fig. 11 Kursawe: average HV (normalized scale)

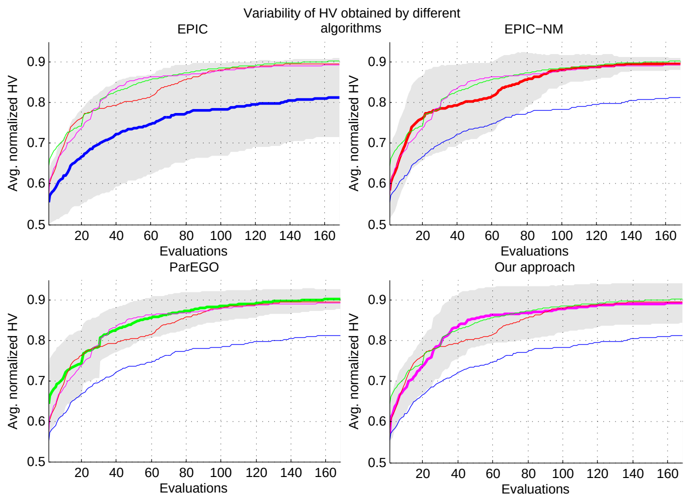

Fig. 12 Kursawe: average and standard deviations of HV (normalized scale)

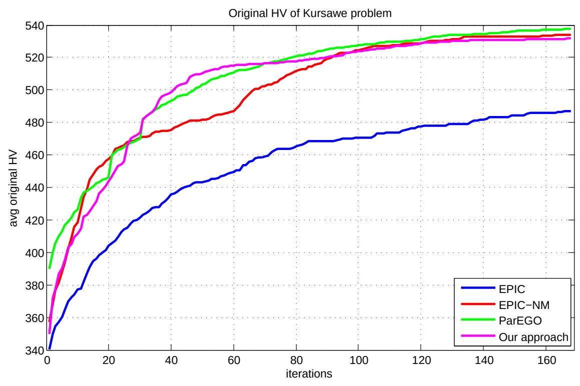

Fig. 13 Kursawe: average HV (original scale)

distributed representation of the Pareto optimal set, HV should be calculated using a normalized objective function axis. However, if the priority is given to the objective function with the larger scale, we might be willing to use the original scales in order to have favorable bias to the larger objective.

Moreover, Figures 9 and 12 demonstrate the standard deviation of HV as a shaded area. It can be seen that for the ZDT3 problem, the EPIC algorithm and the proposed approach do not vary significantly which means that for all runs their HV metric values converge to the average. The HV variance for the Kursawe problem is a little larger indicating slower convergence.

In summary, taking into account that there is nothing or very little known about the real-world optimization problem at hand, the proposed approach can be very promising in generating the approximation of the true Pareto set.

5 Conclusions and Future Work

5.1 Conclusions

Summarizing, we have introduced an approach to approximating the Pareto optimal set for expensive black-box problems by switching among different algorithms. In the proposed approach, the algorithms are selected based on the prediction of their performance using the information available at that time. A classification technique is used to build a performance prediction model that is used to predict which of the algorithms is likely to outperform the others in the current situation. The initial results of our proposed dynamic switching approach are very encouraging and deserve further investigation.

Although the proposed approach is applicable to any costly multiobjective black-box optimization problem, its performance depends on the prediction model which is strongly affected by the training set data. Therefore, to get a reliable prediction model, we have to perform extensive calculations where all the algorithms are provided with the same initial conditions. This is a very time consuming process, as each algorithm has to be tested with many situations: problems with different characteristics, an approximated optimal Pareto set at different moments in time, etc.

However, if one works in a specific application area with a well-defined set of similar optimization problems, it can be beneficial to train our approach only for that class of problems.

5.2 Discussion on future research directions

We plan to build a performance model with a higher classification accuracy by collecting more data over a larger set of test problems. Future work includes searching for new descriptive metrics that better characterize the situation at each instance. One such metric is fitness distance correlation [10], a landscape metric for multiobjective optimization providing useful information concerning the relative difficulty of moving “along” the Pareto optimal set in the objective space. Moreover, we plan to investigate the strengths and weaknesses of the different algorithms as optimization results by different algorithms seem to depend on the characteristics of the problem. Finally, we will augment the set of algorithms to include the most diverse set of state-of-the-art algorithms devoted to multiobjective black-box optimization problems.

References

1. Bader, J., Zitzler, E.: Hype: An algorithm for fast hypervolume-based many-objective optimization. Evolutionary computation **19**(1), 45–76 (2011)
2. Borrett, J.E., Tsang, E.P.: Adaptive constraint satisfaction: the quickest first principle. In: Computational Intelligence, pp. 203–230. Springer (2009)
3. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
4. Deb, K.: Multi-objective optimization using evolutionary algorithms, vol. 16. John Wiley & Sons (2001)
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. Evolutionary Computation, IEEE Transactions on **6**(2), 182–197 (2002)
6. Durillo, J.J., Nebro, A.J.: jmetal: A java framework for multi-objective optimization. Advances in Engineering Software **42**(10), 760–771 (2011)
7. Feng, Z., Zhang, Q., Zhang, Q., Tang, Q., Yang, T., Ma, Y.: A multiobjective optimization based framework to balance the global exploration and local exploitation in expensive optimization. Journal of Global Optimization pp. 1–18 (2014)
8. Forrester, A.I., Keane, A.J.: Recent advances in surrogate-based optimization. Progress in Aerospace Sciences **45**(1–3), 50–79 (2009)
9. Gao, F., Han, L.: Implementing the nelder-mead simplex algorithm with adaptive parameters. Computational Optimization and Applications **51**(1), 259–277 (2012)
10. Garrett, D., Dasgupta, D.: Multiobjective landscape analysis and the generalized assignment problem. In: Learning and Intelligent Optimization, pp. 110–124. Springer (2008)
11. Han, L., Neumann, M.: Effect of dimensionality on the nelder–mead simplex method. Optimization Methods and Software **21**(1), 1–16 (2006)
12. Jiang, S., Ong, Y.S., Zhang, J., Feng, L.: Consistencies and contradictions of performance metrics in multiobjective optimization. Cybernetics, IEEE Transactions on **44**(12), 2391–2404 (2014)
13. Jin, R., Chen, W., Simpson, T.: Comparative studies of metamodelling techniques under multiple modelling criteria. Structural and Multidisciplinary Optimization **23**(1), 1–13 (2001)
14. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization **13**(4), 455–492 (1998)
15. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of artificial intelligence research pp. 237–285 (1996)
16. Knowles, J.: Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation **10**(1), 50–66 (2006)
17. Koduru, P., Dong, Z., Das, S., Welch, S.M., Roe, J.L., Charbit, E.: A multiobjective evolutionary-simplex hybrid approach for the optimization of differential equation models of gene networks. Evolutionary Computation, IEEE Transactions on **12**(5), 572–590 (2008)
18. Kolda, T.G., Lewis, R.M., Torczon, V.: Optimization by direct search: New perspectives on some classical and modern methods. SIAM review **45**(3), 385–482 (2003)
19. Kursawe, F.: A variant of evolution strategies for vector optimization. In: H.P. Schwefel, R. Mnner (eds.) Parallel Problem Solving from Nature, vol. 496, pp. 193–197. Springer Berlin Heidelberg (1991)
20. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence properties of the nelder–mead simplex method in low dimensions. SIAM Journal on optimization **9**(1), 112–147 (1998)
21. Lagoudakis, M.G., Littman, M.L.: Algorithm selection using reinforcement learning. In: ICML, pp. 511–518. Citeseer (2000)
22. Luersen, M.A., Le Riche, R.: Globalized nelder–mead method for engineering optimization. Computers & structures **82**(23), 2251–2260 (2004)
23. Marler, R.T., Arora, J.S.: Survey of multi-objective optimization methods for engineering. Structural and multidisciplinary optimization **26**(6), 369–395 (2004)

24. Miettinen, K.: Nonlinear multiobjective optimization, vol. 12. Springer Science & Business Media (1999)
25. Mockus, J.: Bayesian Approach to Global Optimization. Kluwer Academic Publishers, Dordrecht (1989)
26. Nelder, J.A., Mead, R.: A simplex method for function minimization. The computer journal **7**(4), 308–313 (1965)
27. Okabe, T., Jin, Y., Sendhoff, M.O.B.: On test functions for evolutionary multi-objective optimization. In: X. Yao, E. Burke, J. Lozano, J. Smith, J. Merelo-Guervs, J. Bullinaria, J. Rowe, P. Tio, A. Kabn, H.P. Schwefel (eds.) Parallel Problem Solving from Nature – PPSN VIII, vol. 3242, pp. 792–802. Springer Berlin Heidelberg (2011)
28. Pham, N., Wilamowski, B.M.: Improved nelder mead's simplex method and applications. Journal of Computing **3**(3), 55–63 (2011)
29. Ponweiser, W., Wagner, T., Biermann, D., Vincze, M.: Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathcal{S}$-metric selection. In: G. Rudolph, T. Jansen, N. Beume, S. Lucas, C. Poloni (eds.) Parallel Problem Solving from Nature – PPSN X, *Lecture Notes in Computer Science*, vol. 5199, pp. 784–794. Springer Berlin Heidelberg (2008)
30. Rice, J.R.: The algorithm selection problem (1975)
31. Santana-Quintero, L., Montaño, A., Coello, C.C.: A review of techniques for handling expensive functions in evolutionary multi-objective optimization. In: Y. Tenne, C.K. Goh (eds.) Computational Intelligence in Expensive Optimization Problems, vol. 2, pp. 29–59. Springer Berlin Heidelberg (2010)
32. Steponavičė, I., Hyndman, R.J., Smith-Miles, K., Villanova, L.: Efficient identification of the pareto optimal set. In: Learning and Intelligent Optimization, pp. 341–352. Springer International Publishing (2014)
33. Torczon, V.J.: Multi-directional search: a direct search algorithm for parallel machines. Ph.D. thesis, Citeseer (1989)
34. Törn, A., Žilinskas, A.: Global Optimization, *Lecture Notes in Computer Science*, vol. 350 (1989)
35. Van Veldhuizen, D.A., Lamont, G.B.: Multiobjective evolutionary algorithm test suites. In: Proceedings of the 1999 ACM symposium on Applied computing, pp. 351–357. ACM (1999)
36. Viennet, R., Fonteix, C., Marc, I.: New multicriteria optimization method based on the use of a diploid genetic algorithm: Example of an industrial problem. In: Selected Papers from the European conference on Artificial Evolution, pp. 120–127. Springer-Verlag, London, UK, (1996)
37. Wagner, T.: Planning and Multi-objective Optimization of Manufacturing Processes by Means of Empirical Surrogate Models. Vulkan (2013)
38. Wu, J., Azarm, S.: Metrics for quality assessment of a multiobjective design optimization solution set. Journal of Mechanical Design **123**(1), 18–25 (2001)
39. Zahara, E., Kao, Y.T.: Hybrid nelder–mead simplex search and particle swarm optimization for constrained engineering design problems. Expert Systems with Applications **36**(2), 3880–3886 (2009)
40. Zapotecas-Martínez, S., Coello, C.A.C.: Monss: A multi-objective nonlinear simplex search approach. Engineering Optimization (ahead-of-print), 1–23 (2015)
41. Zhang, Q., Liu, W., Tsang, E., Virginas, B.: Expensive multiobjective optimization by moea/d with gaussian process model. Evolutionary Computation, IEEE Transactions on **14**(3), 456–474 (2010)
42. Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In: Evolutionary multi-criterion optimization, pp. 862–876. Springer (2007)
43. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation **8**(2), 173–195 (2000)
44. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms – a comparative case study. In: Parallel Problem Solving from Nature - PPSN-V, pp. 292–301. Springer (1998)
45. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. Evolutionary Computation, IEEE Transactions on **7**(2), 117–132 (2003)