A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms

Tinkle Chugh * [1], Karthik Sindhya[1], Jussi Hakanen[1], and Kaisa Miettinen[1]

[1]University of Jyvaskyla, Faculty of Information Technology, P.O. Box 35 (Agora), FI-40014 University of Jyvaskyla, Finland

Abstract

Evolutionary algorithms are widely used for solving multiobjective optimization problems but are often criticized because of a large number of function evaluations needed. Approximations, especially function approximations, also referred to as surrogates or metamodels are commonly used in the literature to reduce the computation time. This paper presents a survey of 45 different recent algorithms proposed in the literature between 2008-2016 to handle computationally expensive multiobjective optimization problems. Several algorithms are discussed based on what kind of an approximation such as problem, function or fitness approximation they use. Most emphasis is given to function approximation based algorithms. We also compare these algorithms based on different criteria such as metamodelling technique and evolutionary algorithm used, type and dimensions of the problem solved, handling constraints, training time and the type of evolution control. Furthermore, we identify and discuss some promising elements and major issues among algorithms in the literature related to using an approximation and numerical settings used. In addition, we discuss selecting an algorithm to solve a given computationally expensive mmultiobjective optimization problem based on the dimensions in both objective and decision spaces and the computation budget available.

Keywords: surrogate, metamodel, machine learning, multicriteria optimization, computational cost, response surface approximation, Pareto optimality

1 Introduction

Many engineering problems have multiple objectives to be optimized and these objectives are typically conflicting in nature, i.e. improvement in one objective is possible only by allowing deterioration of at least one of the other objectives. These kinds of problems are known as multiobjective optimization problems (MOPs). Because of the conflicting nature, there typically does not exist one optimal solution, but multiple so-called Pareto optimal solutions. The set of all Pareto optimal solutions in the objective space is called a Pareto front. In many problems, explicit formulations of objective or constraint functions are not known and such functions are called black box functions. Usually, problems involving such functions need a long time to be solved e.g. problems involving

*Corresponding author (tinkle.chugh@jyu.fi)

computational fluid dynamics simulations utilizing finite element algorithms take substantial time to obtain one solution. These are examples of problems that we refer to as computationally expensive multiobjective optimization problems.

In the last few decades, evolutionary algorithms (EAs) [20, 28] have been widely used for solving MOPs because of their advantages such as obtaining a set of nondominated solutions in one solution process, ability to handle problems with multiple local and nonconvex Pareto fronts, ability to easily deal with different kinds of variables (such as binary, real, integer or mixed) and no assumptions set on convexity and differentiability of objectives and constraints involved. Despite of these advantages, EAs do not guarantee convergence to optimal solutions. Moreover, they are often criticized as they consume many function evaluations which increases the computation time. This concern is particularly relevant when dealing with computationally expensive problems. It is therefore required to adapt EAs in a way that they can be used to obtain solutions in less computation time without too much reduction in the quality of solutions. In this paper, a survey of different algorithms proposed in the literature to handle MOPs with computationally expensive objective functions is presented. For other challenges related to solving MOPs, see [20, 28, 93].

Several algorithms have been proposed reduce the computational cost while solving MOPs using EAs. Different surveys exist in the literature [61, 62, 119, 74] on this topic. One of the popular approaches mentioned in these surveys is the use of approximation, especially, function approximation or metamodelling to handle computationally expensive problems. The potential of metamodelling techniques has been widely studied in the machine learning community [41]. Many algorithms exist in the literature which use a combination of these techniques and evolutionary algorithms.

In [61], different ways of using approximation such as problem, function and fitness approximation were discussed based on their use in the literature. Additionally, various issues such as how evolutionary algorithms can benefit from these approximations and what kind of approximation to be selected etc. were also discussed. It is to be noted that fitness approximation is also used as evolutionary approximation in the literature [61]. In [119], the main focus was to discuss various algorithms proposed in the literature before the year 2008 based on the classification of [61]. Some real-world applications were also mentioned, where different algorithms had been applied to reduce the computation time. In [62], the main focus was to discuss the recent developments for using function approximation in reducing the computation time. In that survey, different issues such as strategies for managing approximation, selection of individuals to be evaluated using approximation functions etc. were discussed. In addition, the author also mentioned the potential of using function approximation in dynamic, robust and constrained optimization problems. A taxonomy of metamodel based optimization algorithms was provided in [49]. In [74], several algorithms based on the management of using function approximation (e.g. Kriging, radial basis function etc.) were discussed. In addition, the importance of interactive algorithms with EAs was detailed. Two real-world applications were also mentioned, which were solved by two different algorithms, ParEGO [71] and the algorithm proposed in [98].

In this survey, we extend these surveys and discuss 45 algorithms based on the classification of [61] from the year 2008 to 2016 published in different journals and conference proceedings in English. We also found that most of the algorithms use Kriging when compared to other function approximation techniques and therefore, classify different function approximation based algorithms into Kriging and non-Kriging based algorithms. This survey is also different from other surveys in following ways. 1. Algorithms using function approximation or surrogates are emphasized as they are more widely used. 2. Classification of function approximation based algorithms further into Kriging and non-Kriging based algorithms shows the wide applicability of Kriging. 3. Algorithms are described in reference to a general function approximation framework which will provide the reader an understanding for using any function approximation in EAs. 4. The efficiency of different

algorithms in terms of reducing computational cost or number of function evaluations is emphasized. 5. Various shortcomings in algorithms are observed and discussed e.g. handling constraints, dimensions (both in decision and objective spaces) of the problem solved, training time etc. 6. Some promising elements are also identified which can be helpful in overcoming the limitations of several issues e.g. efficient management of function approximation technique to handle more than three objectives. 7. Some guidelines are also provided in selecting a particular algorithm based on the characteristic of the problem to be solved, dimensions in both objective and decision spaces and the budget available.

In the rest of this paper, in Section 2, some relevant concepts are described which are frequently used when reducing the computational burden. In Section 3, different algorithms are discussed based on the steps of general approximation framework. In addition, different algorithms are compared based on the type of approximation, evolutionary algorithm, evolution control and characteristics of the optimization problem solved. A brief discussion on various issues and using promising elements related to the use of different algorithms is presented in Section 4. Finally, conclusions are drawn in Section 5 along with future research directions.

2 Basic concepts and terminology

We consider multiobjective optimization problems of the form [93]:

$$\begin{aligned} &\text{minimize } \{f_1(x), \ldots, f_k(x)\} \\ &\text{subject to } x \in S \end{aligned} \tag{1}$$

with $k(\geq 2)$ objective functions $f_i(x) : S \to \Re$. The vector of objective function values is denoted by $f(x) = (f_1(x), \ldots, f_k(x))^T$. For the simplicity of presentation, we assume that all the objective functions are to be minimized. If some objective function f_i is to be maximized, it is equivalent to minimize $-f_i$. The (nonempty) feasible region S is a subset of the decision variable space $\Re^n$ and consists of decision variable vectors $x = (x_1, \ldots, x_n)^T$ that satisfy all the constraints. The image of the feasible space S in the objective space $\Re^k$ is called the feasible objective set denoted by Z. The elements of Z are called feasible objective vectors denoted by $f(x)$ or $z = (z_1, \ldots, z_k)^T$, where $z_i = f_i(x), i = 1, \ldots, k$, are the objective function values.

As discussed in the introduction, objective functions in a MOP are typically conflicting in nature, and, thus, there is no single well-defined optimal solution but a set of so-called Pareto optimal solutions exists. A decision vector $x^* \in S$ is Pareto optimal if there does not exist another decision vector $x \in S$ such that $f_i(x) \leq f_i(x^*)$ for all i=1,...,k and $f_j(x) < f_j(x^*)$ for at least one index j. An objective vector is Pareto optimal if the corresponding decision vector is Pareto optimal. A Pareto optimal set consists of all Pareto optimal solutions in the decision space and a Pareto front consists of all Pareto optimal solutions in the objective space. A solution $x^1 \in S$ is said to dominate another solution $x^2 \in S$, denoted by $x^1 \preceq x^2$ if $f_i(x^1) \leq f_i(x^2)$ for all $i = 1, \ldots, k$ and $f_i(x^1) < f_i(x^2)$ for at least one $i \in \{1, \ldots, k\}$.

Ideal and nadir objective vectors represent bounds for objective function values in the Pareto front. Components of an ideal objective vector $z^* \in \Re^k$ are determined by minimizing each objective function individually, that is, $z_i^* = \underset{x \in S}{\text{minimize}} f_i(x)$. A nadir objective vector consists of upper bounds of objective functions in the Pareto front. It is usually difficult to obtain for problems with more than two objectives. Several ways of approximating it have been proposed in the literature (see e.g. [29, 93]). In what follows, we define concepts relevant to the present survey.

A *metamodel* is an approximation of some computationally expensive element in a multiobjective optimization problem. A metamodel replaces the computationally expensive element by an element

which consumes less computation time. There can be different computationally expensive elements while solving a MOP. For example, original functions or constraints, hypervolume etc. A surrogate is often used as a synonym for a metamodel and the process to build a metamodel is known as metamodelling or function approximation. Therefore, what to approximate using a metamodel is an important concern in reducing the computation time. Neural networks [94, 145], radial basis functions [111], support vector regression [7] and Kriging [52] are some examples of commonly used metamodelling techniques.

An *ensemble of metamodels* means using more than one metamodel to reduce the computation time and usually there are two ways to use an ensemble of metamodels in the literature. In the first one, one metamodel having the highest accuracy among different metamodels is selected to evaluate individuals [127]. In the second one, different metamodels are used to evaluate individuals with a different weight coefficient (ω_j) assigned to them. For instance, in [84], the predicted fitness value from an ensemble of m metamodels was defined as:

$$F_{ens}(x) = \sum_{j=1}^{m} \omega_j \overline{F^j}(x) \text{ and } \sum_{j=1}^{m} \omega_j = 1, \tag{2}$$

where $\overline{F^j}(x)$ is the approximated fitness value of $F = \sum_{i=1}^{k} w_i f_i(x)$ using j^{th} metamodel and $w_i, i = 1, \ldots, k$ is the weight vector assigned to each objective function. A weighted sum method was used in [84], however, other scalarizing methods such as ϵ-constraint method, achievement scalarizing function [93] can also be used to convert a multiobjective optimization problem into a single objective optimization problem. Fitness values, $F_{ens}(x)$ is approximated by m metamodels and ω_j is the weight assigned to the fitness value of the j^{th} metamodel. The fitness value of a metamodel is assigned a larger weight if it is found to be more accurate than other metamodels and the accuracy can be obtained by statistical measurements such as root mean square error [132] etc.

An *evolution control* or *model management* [64] is a strategy to manage metamodels in an EA. There are two different ways to manage metamodels, fixed and adaptive evolution control. In a fixed evolution control, metamodels are used for a prefixed number of generations and updated afterward with a predefined criterion. On the other hand, in an adaptive evolution control, the frequency of using the metamodel is adjusted according to the accuracy of the metamodel. Therefore, the model management is also concerned with when to update the metamodel.

In *fitness inheritance* [121, 150], the fitness values of offspring are evaluated using fitness values of the parents. In *fitness imitation* [70], individuals are grouped into several clusters e.g. using K-means clustering [57] in the decision space and only those individuals are evaluated which represent the clusters (e.g. individuals closest to the centroid of each cluster). The fitness values of other individuals are estimated using the fitness values of the representative individuals.

Next, we discuss the general function approximation framework and describe various algorithms based on the steps of the framework.

3 Approximation based algorithms

In approximation based algorithms, the computationally expensive element of the problem is replaced with an approximation which consumes less computation time. As classified in [61], an approximation can be applied in three ways in multiobjective optimization problems: problem, function and fitness approximation. In problem approximation, the original problem is replaced with a simplified problem which is faster to solve. In function approximation, an approximation of a computationally expensive function is formed which is faster to evaluate. On the other hand, in fitness approximation, the fitness value referring typically to the function value of an individual is derived from

the fitness values of the existing evaluated individuals in its vicinity. The function approximation is more widely used than other approximations and algorithms applying it are discussed next in Section 3.1. Approaches using problem and fitness approximation are discussed in Section 3.2.

3.1 Function approximation

As said, function approximation is the most commonly used approach among approximation based algorithms and there, an explicit or an implicit approximation of a computationally expensive, function is formed, which is faster to evaluate. In what follows, we refer to the functions of the original, computationally expensive problem as original functions. In the literature, metamodel is often used for all objective functions, therefore all objective functions are assumed as computationally expensive. Neural networks [76, 94, 8, 84], Kriging [71, 59, 109, 83] and polynomial regression [52, 127] are examples of algorithms used for function approximation.

The general steps for using function approximation in an EA can be divided into two stages consisting of ten steps and most of the algorithms discussed in this section follow these steps. In what follows, we present a function approximation framework that captures the core of the EA proposed in the literature utilizing function approximation. Additionally, it is assumed that one type of metamodel is used for all objective and constraint functions involved although it is possible to use different metamodels for different objective functions.

Stage 1

1. Initialize the population either randomly or using a sampling method.
2. Evaluate the individuals of the population using the original functions and add them to an archive.
3. If a prefixed number of generations is completed, go to stage 2.
4. Use EA operators (selection, crossover and mutation) to create a new population and go to step 2.

Stage 2

5. *Build or update a metamodel for each computationally expensive objective and constraint function using the individuals from the archive.*
6. Use EA operators (selection, crossover and mutation) to create a new population.
7. For each objective and constraint function, evaluate the individuals of the new population either using the metamodel or the original function (fixed or adaptive evolution control strategy).
8. If a stopping criterion is met, select the non-dominated individuals from step 7 as the final population and stop. Otherwise, continue.
9. *Select individuals from step 7 for re-evaluation using the original functions if needed.*
10. *Add the individuals from step 9 to the archive and go to step 5.*

In stage 1, a general evolutionary algorithm works and in stage 2, an EA with a metamodel is used. In step 1, a population is initialized either randomly or using a sampling method such as Latin hypercube sampling [91]. Individuals of the population are evaluated using the original functions and the evaluated individuals are added to an archive in step 2. If a prefixed number of generations

is completed in step 3, stage 1 is terminated and the archive is carried over to stage 2. Otherwise, a new population (offspring population) is generated using EA operators such as selection, crossover and mutation in step 4. A prefixed number of generations is required to obtain an archive of a fixed size. In most of the papers cited in this paper, there is no explicit criterion mentioned for choosing the prefixed number of generations. However, it is mentioned in [6] that the prefixed number of generations depends on the dimensions of the problem (both in objective and decision spaces) and the budget of evaluations with the original functions. For instance, in three popular algorithms known as ParEGO [71], SMS-EGO [109] and MOEA/D-EGO [148], a data set of size *11n-1* was used for training the metamodels. We provide the size of the data set used in different algorithms in Table 1.

After completion of stage 1, a metamodel is created for each computationally expensive objective and constraint function to work with the EA algorithm for evaluating individuals in stage 2. A metamodel is created using the individuals from the archive in step 5. In step 6, an offspring population is generated using EA operators. In step 7, either the metamodel or the original functions are used to evaluate individuals i.e. a fixed or an adaptive evolution control strategy is used to manage the metamodel. In step 8, if a termination criterion such as maximum number of generations or function evaluations is met, nondominated individuals from the last population are selected as the final population. Otherwise, some individuals are selected from step 7 and re-evaluated using the original functions in step 9. There are different criteria mentioned in the literature to select individuals for re-evaluation e.g. selection of nondominated individuals, using expected improvement [67], expected hypervolume improvement [109] etc. These individuals are then added to the archive in step 10 to update or re-train the metamodel. If the size of the archive is prefixed, some individuals (e.g. random or dominated individuals) are eliminated from the archive using a predefined criterion.

3.2 Challenges in using metamodels

Before going into the details of different algorithms, we mention here major challenges when applying function approximation in multiobjective optimization when compared to single objective optimization. It is not straightforward to use a metamodel with an EA because several challenges exist which affect the performance of the metamodel used. These challenges are also the main differences in several algorithms in using function approximation.

1. *Using the metamodel:* In case of single objective optimization, often one metamodel is used for the approximation of the objective and constraint function. Using metamodels in multiobjective optimization is not that trivial. For example, one can use different metamodels for different objective functions, one metamodel for all objective functions, different metamodels for one objective function or a single metamodel for scalarized objective functions. The same options may arise for different constraints. In the literature, most of the algorithms use a single metamodel for all objective functions. In addition, there are no guidelines for using different metamodels for different objective functions.

2. *How to update the metamodel:* Updating the metamodel is an important step both in single objective and multiobjective optimization. In case of single objective optimization, different strategies have been used for updating the metamodel e.g. lower confidence bound [33], probability of improvement [135], expected improvement [120, 67] etc. In case of multiobjective optimization, two goals, convergence and diversity have to be achieved in contrast to single objective optimization. In case of multiobjective optimization, one of the simplest ways is to select the nondominated individuals for re-evaluation [63, 64] using the original functions. Alternatively, a clustering method such as K-means clustering [57] can be applied to cluster

the individuals in the objective space and the representative individuals are re-evaluated using the original functions [45, 65]. The representative individuals can be the individuals closest to each cluster center or the best individuals (with highest fitness values) in each cluster. Individuals with low approximation accuracy can also be selected for re-evaluation [13, 39] and this selection may be effective in improving the approximation accuracy of the metamodel.

In the literature, criteria for updating the metamodel in single objective optimization are also modified to be used in multiobjective optimization. For instance, expected improvement in [71, 60, 69, 24, 140] and probability of improvement in [24] are used to update the metamodel in multiobjective optimization. In addition, expected hypervolume improvement is used in [38, 37, 124] to select the individuals for re-evaluation so that the metamodel can be updated. It is to be noted that infilling criterion is sometimes used as a synonym for updating criterion in the literature.

3. *Training time for the metamodel:* Training time is also an issue in case of single objective optimization but when moving to multiobjective optimization, training or updating time becomes more influential. For example, different metamodels for different objective functions may take a different amount of training data (archive size) which increases the training time when compared to single objective optimization. In the literature, unfortunately, most of the algorithms do not mention the training time for the metamodel. This is due to the reason that the training time of the metamodels is usually assumed to be negligible compared to objective function evaluations. However, it may happen that the training time of the metamodels is substantial and the whole aim of reducing computation time is jeopardized.

In addition to the three challenges above, two challenges exist which are common to both single objective and multiobjective optimization. They can influence the performance of an EA while using metamodels.

1. *Choice of the metamodel:* In the literature, there is very little guidance about the choice of the metamodel for approximation of computationally expensive functions. A metamodel is either selected randomly or due to its popularity in the area with which the problem is associated. For instance, in [76], radial basis neural network was used as a metamodel for approximation of objective functions in coastal aquifer management problem and it was mentioned that neural networks were popular for groundwater applications. Kriging was used in [59] due to its promising nature for building accurate global approximations.In [101], different metamodelling techniques were used for different problems i.e. RBF with linear kernel function for ZDT problems and Kriging for UF [149] problems. The reason mentioned was that RBF was used due to simplicity of the function landscape in ZDT problems and and Kriging was used because functions in UF problems are highly non-linear and test with RBF showed a very low accuracy when used for UF problems. However, the algorithm was developed for black-box computationally expensive problems, where function landscape of the problems in question is not known a priori. Moreover, Kriging is most popular technique when compared to others because of its ability to provide uncertainty for the approximated values. To overcome the problem of choosing of a metamodel, an ensemble of metamodels described in Section 2 is also used in the literature [127, 84]. However, there are still some open challenges related to the ensemble of metamodels such as what should be the criterion for choosing different metamodels or how different metamodels can be used simultaneously?

2. *When to update the metamodel:* It is also an important issue to decide when to update the metamodel or in other words, is there any need to update the metamodel. For example, before selecting individuals in step 9, one can check whether the existing metamodel is accurate

enough to predict objective function values in the next generation. If so, the metamodel is not updated and the existing metamodel is used for approximation. An offspring population has to be generated before updating the metamodel so that the metamodel accuracy can be measured for the individuals of the offspring population. There may be a possibility that the metamodel is never updated and the metamodel trained after stage 1 is used in all generations. In that situation, steps 9 and 10 are eliminated from stage 2. In the literature, there is very little focus on the question of when to update the metamodel.

3. *Handling constraints:* While using metamodels in constrained problems, an appropriate set of solutions is needed to train them. Most of the metamodel-based algorithms in the literature are not developed to handle constraints. In addition, few algorithms which are developed to handle constraints assume that feasible solutions are available to train metamodels for constraint function. In many problem, obtaining a feasible solution is not trivial e.g. C1-DTLZ1 problem [58] with very small feasible region. In such instances, using metamodels for constraints can be very challenging. We provide more details of handling constraints while describing different algorithms.

Steps 5, 9 and 10 representing the first three main challenges are written in italics in the function approximation framework to indicate the steps, where algorithms using function approximation in multiobjective optimization mainly differ from each other. Other three challenges, selection of a metamodel, when to update the metamodel and size of the data to train the metamodel are not the major differences in the algorithms.

In what follows, 30 algorithms and from the literature are discussed utilizing the three main steps (5,9 and 10) of the function approximation framework and attention is paid to their efficiency in reducing the computation time. These algorithms are classified according to the number of metamodels they have used e.g. single metamodel, multiple metamodels independently or ensemble of metamodels. Among single metamodel based algorithms, Kriging is used more often than other metamodels. As mentioned, the main advantage from Kriging is the uncertainty information besides the predicted objective and/or constraint function values. This uncertainty information can be further utilized in the algorithm. For instance, uncertainty information is used while updating the metamodel in [37, 18, 110, 71].

In contrast to single metamodel based algorithms, some algorithms use metamodels independently. Moreover, some algorithms use ensemble of metamodels to solve the problem of choosing a metamodel. In what follows, three categories, algorithms based on single metamodel, algorithms based on multiple metamodels and algorithms based on ensemble of metamodels, are used to describe different algorithms. It was found that the number of papers in the literature belonging to the function approximation are largely skewed towards the first category, i.e. algorithms based on a single metamodel. Hence a sub-classification based on the usage of metamodel within different algorithms is devised to enhance clarity and readability. Thus we further classify the algorithms using single metamodel into Kriging and non-Kriging based algorithm. Due to a limited number of articles in other categories, such a sub-classification is not used for others. It is also worth mentioning that most of the algorithms are based on dominance based EAs and are generic for using any metamodel. At the end of this section, a comparison is presented based on which metamodel and type of EAs are used, what is the evolution control strategy and what are the characteristics of the optimization problems.

Parameter values used in these algorithms in stages 1 and 2 of the function approximation framework are presented in Table 1, where "NA" indicates that the parameter is not applicable to the algorithm and "not given" means that this information is not given in the reference. The table collects information of the population size in step 1, the size of archive in step 2, the prefixed number

of generations in step 3 in stage 1 and the number of individuals for re-evaluation in step 9 in stage 2, as reported for example problems solved in the papers cited.

All parameters mentioned in Table 1 are important and can influence the performance of the algorithm. The population size $|N_P|$ is a critical parameter and several studies like [16, 22] show the effect of the population size on the performance of evolutionary algorithms. The size of the archive $|N_A|$ mainly affects the training time of the metamodels and as can be seen from the table, most of the algorithm do not have a fixed size archive. The size of data set $|N_I|$ used to train metamodels is different in different algorithms. One can adjust this parameter based on the resources or maximum number of function evaluations available. In addition, the size of the data set $|N_U|$ to update the surrogates is also important and should be decided based on the resources available. For instance, if it is possible to do parallel evaluations, one can select the size accordingly. An example is given in [72], where the ParEGO algorithm could not be applied in doing experiments for drug design because the algorithm did not have an option to select multiple sample points at a time. The maximum number of function evaluations FE^{max} is also important when comparing different algorithms. Many algorithms use different numbers of function evaluations, which makes it difficult to select an algorithm to solve a given problem. The number varies from 50 to 30000 in the literature. Next, we classify algorithms which use only a single metamodel, which are further classified into Kriging and non-Kriging based algorithms.

3.3 Algorithms based on a single metamodel

In this subsection, we discuss algorithms using one metamodel for all objective and/or constraint function. As mentioned, to make the a clear structure of the paper and also due to wide applicability of Kriging models, we further classify this subsection into Kriging based algorithms and non-Kriging based algorithms. Further these algorithms are discussed year wise, i.e. starting from the year 2008.

3.3.1 Kriging based algorithms

In this section, we discuss algorithms using Kriging.

A DACE model [118] which is a Kriging based approximation was used in [109]. This algorithm is known as SMS-EGO which is an extension of efficient global optimization (EGO) [67] for multi-objective optimization problems. In step 5, the metamodel is built for each objective function and to update it in step 9, one individual having maximum contribution to the hypervolume is selected for re-evaluation using the original functions. This individual is then added to the archive in step 10 and the metamodel is updated in the next generation. The size of the archive is not fixed in this algorithm.

SMS-EGO was tested on five benchmark problems (2-5 objectives and 3-6 decision variables) and compared with ParEGO [71] and the algorithm proposed in [60]. The unary hypervolume indicator [156], the R2 indicator [46] and the unary epsilon indicator [157] were used as the comparison criteria. The proposed algorithm performed better than the other algorithms in all cases when hypervolume was used as the performance measure. In terms of other comparison criteria, it performed worse than the other algorithms for three problems.

In Li et al. [83], Kriging was used as a metamodel with a modified version of NSGA-II. In this algorithm which is known as K-MOGA, the metamodel is used for approximating each objective function in step 5. To update the metamodel in step 9, domination status is measured for each individual evaluated using the metamodel. Individuals which change the domination status are re-evaluated using the original functions and added to the archive in step 10. To check the domination status, minimum of minimum distance (MMD) is measured. To calculate MMD, individuals are

Table 1: Parameter values used in different algorithms: $|N_P|$= population size in EA, $|N_A|$= size of archive in step 10, $|N_I|$= size of the data set to train metamodels in step 5, $|N_U|$= size of the data set for updating the metamodels and maximum

reference	$\|N_P\|$	$\|N_A\|$	$\|N_I\|$	$\|N_U\|$	FE^{max}
[3]	not given	not fixed	not given	4	200-400
[6]	300	300	200	5	2000
[14]	5	not fixed	3-19	10	23-300
[18]	105-275	*11n-1*	*11n-1*	5	250-300
[27]	50-100	not fixed	50-100	not fixed	500-5000
[48]	1000	2000	2000	NA	2000
[52]	100	not fixed	not given	5	not given
[59]	500	not fixed	12	1	66
[76]	40	not fixed	100	1	3100
[81]	20-50	not fixed	16-40	not fixed	58-497
[83]	30	not fixed	30	not fixed	148-901
[84]	100	1000-2000	1000-2000	2	8000-30000
[85]	not given	not fixed	6-18	10	58-242
[86]	140-200	not fixed	30	10-40	140-3600
[90]	300	not fixed	$10n$	10	1000-5000
[95]	100	not fixed	100	not fixed	267-8988
[94]	50	not fixed	50	50	2000
[102]	26	not fixed	150	1	1606
[101]	26-50	not fixed	150	1	NA
[104]	300	not fixed	$10n$	40	1000-2000
[103]	100	not fixed	$10n$	10	200-2000
[107]	100	800	100	1	30000
[109]	not given	not fixed	$11n$-1	1	130
[114]	not given	not fixed	*(n+1)(n+2)/2*	1	200-500
[117]	25	not fixed	150	2	500
[127]	not given	not fixed	1000	not fixed	6000-11900
[128]	NA	not fixed	15-58	1	50-150
[131]	20-60	not fixed	50	1	3000
[148]	300-595	not fixed	$11n$-1	5	200-300
[151]	140	not fixed	6-10	6-10	190-228

partitioned into two sets in the decision space, nondominated (x^{nd}) and dominated (x^{d}). MMD is calculated as, MMD = $min\left\{\left\|\hat{f}(x^{nd})-\hat{f}(x^{d})\right\|\right\}$, where $\hat{f}$ is the predicted objective vector using the metamodel. MMD is then projected to each objective function axis to obtain MMD$\hat{f}_i$ (for i = 1,...,k). Thereafter, a threshold $s_i(x) \leq$ MMD$\hat{f}_i$ is specified by using the standard deviation obtained ($s_i(x)$) from the Kriging model for each objective function. The individuals which do not satisfy this threshold are re-evaluated using the original functions. The size of the archive is not fixed in this algorithm.

K-MOGA was tested on five benchmark (two objectives and 3-6 decision variables) and two real-world problems (two objectives and 2-4 decision variables) and compared with MOGA. For comparison, nondominated individuals from both algorithms were visualized and similar solutions were obtained in fewer function evaluations with K-MOGA.

The same algorithm was extended to KD-MOGA in [81] with one additional element. In KD-MOGA, a fixed number of individuals is generated using constrained maximum entropy design after step 9, which is an extension of unconstrained maximum entropy design [25]. These individuals are then added to the population for the next generation.

KD-MOGA was tested on the same set of problems as K-MOGA with one more real-world problem (two objectives, four decision variables and four constraints). It was compared with both EAs (modified version of NSGA-II) and K-MOGA using visualization of the nondominated individuals. A similar performance was obtained in fewer function evaluations with KD-MOGA.

Kriging was used as a metamodel in [59]. This paper is based on an algorithm called combined AASO-AAMO (adaptive approximation in single objective optimization - adaptive approximation in multiobjective optimization) [144] which uses a multiobjective genetic algorithm [144] as an EA. The metamodel is built for each objective and constraint function in step 5. To update the metamodel in step 9, a fixed number of individuals is selected for re-evaluation using the original functions and added to the archive in step 10. The individuals having the best value according to the maxmin distance design criterion [66], i.e. the largest value of minimum distance from individuals in the archive in the decision space, are selected for re-evaluation. The size of the archive is not fixed in this algorithm.

The proposed algorithm was tested on a fatigue design MOP with two objectives, 17 decision variables and 11 constraints. It took 70 hours to complete 25 generations but the efficiency of the algorithm was not compared to any EA without using the metamodel. The authors mentioned that it was practically impossible to do the optimization without using approximation.

In Zhang et al. [148], Kriging was used as a metamodel with the decomposition based EA MOEA/D [146]. The algorithm is known as MOEA/D-EGO. In MOEA/D-EGO, after evaluating a fixed number of individuals in steps 1-4, the metamodel is built for the scalarized objective function. Chebyshev scalarizing function [93] is used to convert multiobjective optimization problem into single objective optimization problems. After using the metamodel in step 7, the expected improvement (EI) [67] is calculated for each subproblem. Expected improvement is then maximized (for each subproblem) using MOEA/D-DE [82] for a fixed number of generations. In other words, the scalarized problem is changed into another problem to maximize EI. To update the metamodel in step 9, firstly, all individuals after the local search, which are different from the individuals in the archive (in the decision space) are selected. After this, K-means clustering is used to cluster the weight vectors (used in MOEA/D initially) associated with the individuals selected above. From each cluster, an individual with maximum EI is selected and re-evaluated using the original functions. These individuals are then added to the archive in step 10. Moreover, to reduce the training time, a fuzzy clustering [12] is used for selecting a fixed number of individuals for training or updating the metamodel.

MOEA/D-EGO was tested on 12 benchmark problems (2-3 objectives and 2-8 decision variables) and compared with ParEGO [71] and SMS-EGO [110]. Hypervolume and inverted generational distance (IGD) were used as the comparison criteria. The proposed algorithm outperformed on seven problems when compared to ParEGO and performed similar to SMS-EGO. It was also compared with MOEA/D on two problems (two objectives and 2-8 decision variables) and outperformed with IGD as the performance criterion.

A multiobjective variable-fidelity optimization (VFO) algorithm was proposed in [151], where Kriging was used as a metamodel with NSGA-II. Initially, a simplified or approximated problem (having low fidelity functions) is used to replace the original MOP in stage 1. This simplified problem is then solved for a prefixed number of generations using NSGA-II. A fixed number of nondominated individuals obtained after this step is re-evaluated using the original functions and stored in an archive. These individuals are selected based on a simple inter-individual distance metric in the objective space (steps 1-4). These individuals are then used to construct a Kriging model for each objective function in step 5 of stage 2. For the following fixed number of generations, individuals generated by crossover and mutation in step 6 are evaluated either using the approximated problem functions with Kriging model or just the approximated problem functions (step 7) depending on the error [42] in the Kriging model. Thus, an adaptive evolution control strategy is used to manage the metamodel. In other words, if the metamodel is not accurate for an objective function, the original function is used to evaluate individuals. To update the metamodel in step 9, a fixed number of nondominated individuals evaluated using the metamodel is selected for re-evaluation using the original functions. The individuals are selected by computing the error from the Kriging model and added to the archive in step 10. The size of the archive is not fixed in this algorithm.

The VFO algorithm was tested on the ZDT1-3 benchmark problems (with two objectives and five decision variables) and a structural engineering problem (with two objectives and six decision variables). For the ZDT problems, the efficiency of the VFO algorithm was not mentioned in terms of computation time or number of function evaluations.

For the structural engineering problem, an exhaustive search was carried out with different values of decision variables. Around 65 million combinations of decision variable values were evaluated using both original and approximated problem functions. Nondominated individuals after this search were identified and nondominated individuals obtained using original functions were used to compare results of different studies mentioned next. Fourteen studies were performed with changes in the values of three parameters: number of prefixed number of generations in step 3 of stage 1, number of generations before updating the metamodel of stage 2 and number of individuals selected for re-evaluation in step 9 of stage 2. Two out of fourteen studies were performed using NSGA-II without using a metamodel with high and low fidelity functions (to be called case 1 and case 2, respectively). Results from these studies were then compared with the results of exhaustive search with different criteria (inverted generational distance, error ratio, crowding distance [28] and span [80]). Case 1 gave the best results and in comparison with case 1, one of the studies out of thirteen gave similar results (a graphical comparison was performed) in fewer function evaluations.

In [86], VFO algorithm was extended, where instead of one global metamodel, multiple local metamodels were used. The authors mentioned that local metamodels are used for high dimensional problems in the objective space. K-means clustering is used in the decision space to partition the data and to build multiple local metamodels. Other details are the same as in VFO.

This algorithm was tested on six benchmark problems (two objectives and 2-30 decision variables) and one real-world problem (three objectives and six decision variables). For benchmark problem, nondominated individuals obtained from this algorithm and with VFO and NSGA-II were visualized. The authors mentioned that the proposed algorithm outperformed on these problems. In case of the real-world problem, the same performance criterion was used as in the VFO algorithm. In this

case too, the proposed algorithm performed better than NSGA-II and VFO in the same number of function evaluations.

In [128], Kriging was used as a metamodel. In this algorithm, the main focus was to apply different strategy for objective and constraint functions while updating the metamodel. In step 5, a metamodel is built for each objective and constraint function. To update the metamodel in step 9, for objectives, selection is performed using hypervolume based probability of improvement (POI_{hv}) [24] and for constraints, probability of feasibility (POF) [41] is used. After this, $\gamma = POI_{hv} \times POF$ is obtained and a fixed number of individuals with highest γ values is selected for re-evaluation in step 9. These individuals are then added to the archive in step 10. The size of the archive is not fixed in this algorithm.

The proposed algorithm was tested on two real-world problems with two objectives, 2-3 decision variables and 5-7 constraints. This algorithm was implemented using the SUrrogate MOdeling MATLAB Toolbox (SUMO) [44] and was not compared with any other algorithm.

In [95], Kriging models were used with a differential evolution based EA to approximate the objective functions. After building the metamodels in step 5, individuals were generated using differential evolution operator in step 6 and metamodels were then used for approximation in step 7. For each offspring in step 7, uncertainties of the approximated values were compared to its corresponding parent. In other words, if the uncertainty value of offspring was less than that of parent, it would selected and kept in the population. On the other hand, if the uncertainties of both offspring and parent are comparable, both are kept in the population. All the offspring thus selected were re-evaluated with the original function to update the metamodels in step 9 and added the training archive in step 10.

The algorithm was tested on 12 benchmark problem out of which three had constraints. However, it was not mentioned in the article how the constraints were handled. All these 12 problems had two objectives and the number of decision variables for all problems was not mentioned. The algorithm was also tested on a continuous steel casting and an electrocardiography problem with four variables and 2-3 objectives. The algorithm was compared with differential based EA [116] and an algorithm based on NSGA-II-ANN [30]. The algorithm obtained better performance when measured with hypervolume in 10000 function evaluations.

In Chugh et al. [18], an algorithm called K-RVEA was proposed to solve computationally expensive problems with more than three objectives. In this algorithm, the metamodels were updated based on the need of convergence or diversity. Angle penalized distance [17] and uncertainty information from the Kriging models with the help of reference vectors are used to select individuals in step 9 for updating the metamodels. In addition, extra individuals are removed from the archive in step 10 to further reduce the computation time.

The proposed algorithm was tested on DTLZ and WFG benchmark problems with 3-10 objectives and 10 decision variables. It was also compared with ParEGO, MOEA/D-EGO and SMS-EGO using IGD and hypervolume. In addition to benchmark problems, the algorithm was also tested on a free-radical polymerization problem [96] and also compared with the state-of-the-art algorithms. In the given number of function evaluations, the proposed algorithm performed better than other algorithms.

The same algorithm was also extended to handle constraints in [19]. Three different approaches are used to while training metamodels based on the feasibility of solutions. The proposed algorithm was tested on constrained version of DTLZ problems [58] with 3-10 objectives and 10 decision variables. The authors found out that infeasible solutions are having a vital role on the performance of surrogates.

In Roy and Deb [117], a high dimensional model representation (HDMR) [130] was used as a metamodel for each objective function. Kriging was further used within HDMR to approximate

its component functions. The algorithm was proposed to handle problems with large number of decision variables. In each HDMR model, n component functions were approximated using Kriging, therefore $n \times k$ (n and k represent the number of decision variables and objectives respectively) Kriging models were built and k HDMR models were built. After building the metamodels in step 5, NSGA-II was used to from steps 6-7. To update HDMR models, a prefixed number of individuals were re-evaluated using the original functions in step 9. These individuals were selected by doing clustering in the decision space and then added to the training archive in step 10. In addition, bounds of the decision variables were updated to limit the search space after metamodels were updated.

The algorithm was tested on 17 benchmark problems from ZDT, DTLZ and CEC09 suite [149] biobjective problems with 15-30 decision variables. It was compared with NSGA-II and Kriging based NSGA-II [89] using IGD and performed better than others in 500 function evaluations.

3.3.2 Non-Kriging based algorithms

In this subsection, algorithms using metamodels other than Kriging such as neural network, support vector regression and polynomial regression are discussed.

In [131], a multiobjective parallel surrogate-assisted evolutionary algorithm (MOPSA-EA) was proposed, where a feedforward neural network was used as a metamodel with a steady state EA. The metamodel is built for each objective function in step 5. A fixed number of offspring individuals is generated in step 6 using crossover and mutation and evaluated using the metamodel in step 7. The fitness values of offspring individuals are altered as per the fitness values of parents. The authors mentioned that this approach was motivated by fitness inheritance, where the fitness values of offspring depend on the fitness values of parents.

To get the altered fitness values for offspring individuals, the parents which are used for creating the offspring individual are evaluated with the metamodel and the error is calculated between true fitness values and the approximated fitness values (using the metamodel) for parents. For example, if the errors (for a biobjective optimization problem) between the true fitness values and the approximated fitness values for two parents are (a_e, b_e) and (c_e, d_e) and an offspring individual is generated using these parents having fitness values (e, g), then the new altered fitness values for offspring are $(e + w_1 \times a_e + w_2 \times c_e, g + w_2 \times b_e + w_1 \times d_e)$. The weight coefficients w_1 and w_2 are selected based on the influence of parent individuals on an offspring individual during crossover. An individual is selected in step 9 after getting new fitness values for offspring individuals and added to the archive in step 10 to update the metamodel. To select this individual, a nondominated sorting for offspring individuals and individuals in the archive is performed and nondominated individuals in both sets are identified. Let O_{R1} and P_{R1} denote nondominated individuals in offspring and in the archive, respectively. These individuals are combined and individuals in O_{R1} dominating P_{R1} are identified. Among these individuals, individual having the largest Euclidean distance to its closest individual in P_{R1} is selected and added to the archive. A nondominated sorting is performed again for individuals of the archive and the worst individual (having the worst fitness value and the smallest crowding distance) is removed from the archive. The size of the archive is fixed in this algorithm.

The MOPSA-EA was tested on the ZDT1-4 and ZDT6 benchmark problems (2 objectives and not explicit information of the number of decision variables) and on a manufacturing MOP (with two objectives and 11 decision variables). Generational distance (GD), inverted generational distance (IGD), spread and hypervolume metrics were used to compare the proposed algorithm with SMS-EMOA [36], MAES [38] and NSGA-II-ANN [97]. For the same number of function evaluations, MOPSA-EA performed better than other algorithms in Y, Ω and S metrics and in Δ, SMS-EMOA and MOPSA-EA performed equivalent. As the Pareto front was not known for the manufacturing problem, only the S metric was used as the performance criterion and MOPSA-EA gave better

results in S than the other algorithms in the same number of function evaluations.

A quadratic polynomial approximation was used as a metamodel with the EA μ-MOGA [21] in Liu et al. [85]. In this algorithm, the bounds of decision variables are updated after every generation using a trust region algorithm. In step 5, the metamodel is built for each objective and constraint function. To update the metamodel in step 9, a fixed number of uniformly distributed individuals (P_a) from nondominated individuals is re-evaluated using the original functions. The nondominated individuals in the decision space after re-evaluation are stored in a set P_e. The set $P_o = P_a \cap P_e$ is determined which is then used to calculate a reliability index $N(P_o)/N(P_a)$, where $N(P_o)$ and $N(P_a)$ are the numbers of individuals in P_o and the number of nondominated uniformly distributed individuals evaluated using the metamodel, respectively. This reliability index is used to update the bounds of the decision variables in the next generation with a trust region algorithm. The trust region radius is updated according to [4] and the algorithm terminates if the trust region radius is smaller than a predefined limit or after a fixed number of generations (step 8). A Latin hypercube design is used for sampling the decision variables with updated bounds. These individuals with the updated bounds are evaluated with the original functions which are used to update the metamodel. Step 10 is not applicable in this algorithm as individuals after step 9 are not added to the archive.

The proposed algorithm was compared with μ-MOGA using two benchmark problems (two objectives and 2-3 decision variables) and one structural engineering problem (two objectives, three decision variables and one constraint). Using generational distance as a performance metric for the benchmark problems and visual comparison for the structural engineering problem the quality of the obtained set of nondominated solutions within a fixed budget of function evaluations using μ-MOGA with metamodel was better.

A feedforward neural network was used as a metamodel with NSGA-II in [94]. The metamodel is built for each objective function in step 5 if a threshold based on a predicted tolerance is satisfied. This predicted tolerance, which is an indication of the accuracy of the metamodel, is calculated (however, calculation of the predicted tolerance is not detailed in the paper) after every generation. If this predicted tolerance is less than a user-specified tolerance, then the metamodel is used in the next generation. Otherwise, the original functions are used to evaluate individuals. The predicted tolerance is updated after every generation and again a decision is to be made either to use the original functions or the metamodel and thus, an adaptive evolution control strategy is used. Individuals obtained after every generation are added to the archive (steps 9 and 10) and as a result, the size of the archive grows with generations. In this algorithm, the metamodel is not updated after every generation but after every generation, it is checked whether the existing metamodel is sufficient enough to predict function values to the extent of accuracy required.

This algorithm was tested on an iron induration MOP with two objectives, 22 decision variables and three constraints. To check the efficiency of the proposed algorithm, a graphical comparison was presented to the solutions from the algorithm and NSGA-II without using any metamodel. Similar nondominated individuals were obtained in 50% fewer function evaluations and for the same number of function evaluations, better nondominated individuals were obtained.

In Chen et al. [14], an extension of algorithm proposed in [85] was proposed. The main differences include the type of metamodel, stopping criterion in step 8 and selection of individuals for re-evaluation for updating the metamodel in step 9. The algorithm proposed utilizes a radial basis function as a metamodel. In addition to the stopping criteria posed in [85] mentioned earlier, the algorithm also terminates if the bounds of the decision variables are equal to predefined limit and the reliability index is equal to one. In step 9, the individuals are selected using an inherited Latin hypercube design (ILHD) [141] and a local-densifying strategy (to reduce the possibility of an ill-conditioned RBF matrix) for re-evaluation and subsequently updating the RBF.

The proposed algorithm was tested on eight different benchmark problems (two objectives and 1-5

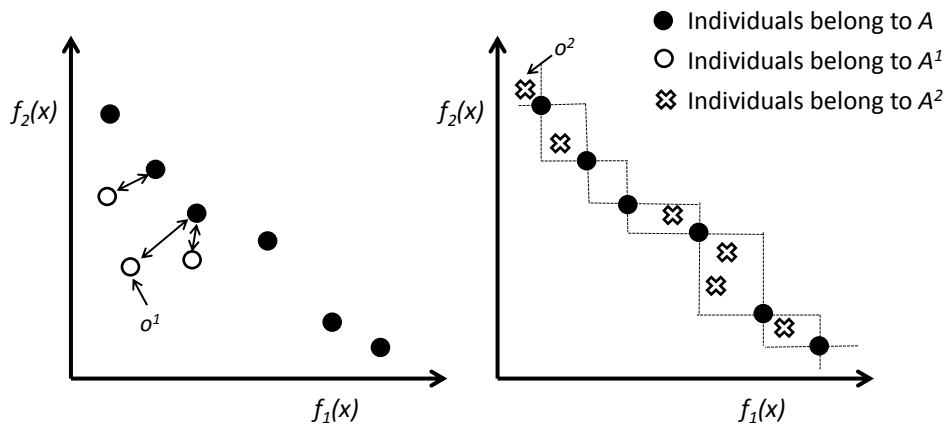

Figure 1: Selection of individual from individuals evaluated using metamodel

decision variables) and a structural engineering problem (two objectives and five decision variables). The results were compared with μ-MOGA without any metamodel and the algorithm proposed in [85]. In case of benchmark problems, the proposed algorithm obtained better values for spread and convergence metrices [31] in fewer function evaluations. For the structural engineering problem, the proposed algorithm was not tested using μ-MOGA without any metamodel.

In [76], a multiobjective surrogate assisted (MOSA) algorithm was proposed, where a modular neural network (MNN) [75] was used as a metamodel with NSGA-II. A metamodel is built for each objective function in step 5. A fixed number (equal to the population size of NSGA-II) of better performing individuals (the authors did not mention any criterion for defining better performing individuals) is used as the population of NSGA-II and the nondominated individuals (A) after this step are determined. The nondominated individuals after evaluating the offspring individuals (generated in step 6) are compared with A to select one individual for re-evaluation (step 9). To do this, the individuals evaluated using the metamodel are clustered into two sets A^1 and A^2 as shown in Figure 1. The first set (A^1) consists of the individuals which dominate at least one individual of A while the second set (A^2) consists of the individuals that do not dominate nor are dominated. For each offspring in A^1, the Euclidean distances between the offspring and the individuals in A are calculated. As shown in left part of Figure 1, individual (o^1) with the largest distance is selected for re-evaluation using the original functions. In case A^1 is empty, an individual is selected from the set A^2 for re-evaluation. To select the individual, for each offspring individual in A^2, the normalized perimeter for rectangles created between consecutive individuals in A (as shown in the right part of Figure 1) is calculated. An individual located in the rectangle with the largest perimeter (o^2) is selected for re-evaluation using the original functions. In case both A^1 and A^2 are empty, offspring are generated again. However, the authors did not mention about how this algorithm can be used for more than two objectives. The selected individual is added to the archive to update the metamodel in step 10. The size of the archive is not fixed in this algorithm.

The algorithm was used to solve a coastal aquifer management optimization problem with two objectives and eight decision variables. The time required to evaluate the original functions was 26 hours while the time to train the metamodels was 63 minutes. A graphical comparison was presented between MOSA and NSGA-II and the proposed algorithm gave similar results in fewer function evaluations.

In Herrera et al. [48], support vector regression was used as a metamodel for approximation of each objective function with NSGA-II. The main focus in this algorithm is to use different basis functions for different kinds of variables such as discrete, continuous and categorical variables. It is to be noted that this algorithm does not use an ensemble of metamodels as only one prediction is obtained for each objective function by using different basis functions for the different kinds of variables. To convert categorical values into real number, dummy coding is used. In step 5, the metamodel is built for each objective function. Steps 9 and 10 are not applicable to this algorithm as the metamodel and the archive are not updated.

The proposed algorithm was tested on one real-world problem (two objectives and 10 decision variables) and compared with NSGA-II. Out of 10 variables, 5 were continuous and 5 were categorical variables. A visualization of nondominated individuals in the objective space was performed to compare the two algorithms. The authors mentioned that the proposed algorithm performed similar to NSGA-II in fewer function evaluations.

In [107], support vector regression was used as a metamodel. This algorithm is known as HO-MOMA, where a metamodel is built for each objective function in step 5. After evaluating new individuals in step 7, a local search is used to optimize the fitness function obtained from the metamodel evaluations. This is done as follows. Firstly, several nondominated fronts are obtained with nondominated sorting. Then, in each front, sorting is performed according to the first objective function values in the ascending order. A reference point is then calculated for each individual in each front using these sorted values. After calculating the reference point, a fitness value for each individual is calculated. Let $\hat{f} = \left\{\hat{f}_1, \hat{f}_2\right\}$ be the predicted objective vector for one individual and $r = \{r_1, r_2\}$ the reference point for that individual. Fitness is then calculated as

$$\text{fitness} = \begin{cases} (r_1 - \hat{f}_1)(r_2 - \hat{f}_2) & \hat{f}_1 < r_1 \text{ and } \hat{f}_2 < r_2 = 0 \\ (r_1 - \hat{f}_1) & \hat{f}_1 > r_1 \text{ and } \hat{f}_2 < r_2 = 0 \\ (r_2 - \hat{f}_2) & \hat{f}_1 < r_1 \text{ and } \hat{f}_2 > r_2 = 0 \\ -d(r, \hat{f}) & \text{otherwise,} \end{cases} \tag{3}$$

where $d(r, \hat{f})$ is the Euclidean distance between r and $\hat{f}$. This fitness is then optimized using CMA-ES [47] as the local search algorithm. To update the metamodel in step 9, nondominated individuals obtained after the local search are added to the archive in step 10. The size of the archive is fixed in this algorithm and extra individuals are removed from it randomly. The authors mention the study for more than two objectives is a future research.

HO-MOMA was tested on 14 benchmark problems with two objectives and 10-30 decision variables. The algorithm was compared with NSGA-II and ASM-MOMA [105] with hypervolume as the performance criterion. The proposed algorithm performed better than the others in nine out of 12 problems.

In [104, 103], an extreme learning based MOEA/D-DE [82] was used. This algorithm is known as ELMOEA/D-DE and inspired by MOEA/D-RBF. The main focus in this algorithm is to use the metamodel for higher dimensional problems in the decision space. Extreme learning is a single-layer feedforward neural network proposed in [50]. In step 5, the metamodel is built for each objective function. To update the metamodel in step 9, the same procedure is used as in MOEA/D-RBF. In addition, a minimum distance (in the decision space) is maintained between the individuals to be selected for re-evaluation. After adding these individuals in the archive, inferior solutions (in terms of scalarized single objective problem) are removed. Otherwise, the closest individual in the objective space is replaced by the individual obtained in step 9. This is done to ensure that every new solution is added to the archive for updating the metamodel.

ELMOEA/D-DE in [104] was tested on ZDT problems with 10-60 decision variables and compared with MOEA/D-DE and MOEA/D-RBF.Later in [103], the algorithm was tested on ZDT, DTLZ and WFG problems with 2-5 objectives and 5-60 decision variables and on an airfoil shape optimization problem with two objectives and 12 decision variables. The algorithm was also compared using addictive-Epsilon and R_2 [156] as the performance indicators. For the given number of function evaluations, the proposed algorithm obtained better performance.

In [3], an algorithm called gap optimized multiobjective optimization using response surfaces (GOMORS) was proposed, where radial basis functions were used to approximate the objective functions. After building the metamodels in step 5 for each objective function, an EA proposed in [139] was used for optimization from step 6-7. After step 7, an another optimization problem was solved using the same EA by reducing the bounds of the decision variables. This problem was referred as gap optimization problem in the article. In step 9 for updating the metamodels, four criterion were used and one individual corresponding to these criterion was selected and added to the training archive in step 10. Four criterion were based on hypervolume, distance to the individuals in the decision space, distance to the individuals in the objective space and hypervolume in the gap optimization problem. A maximum number of function evaluations was used as the termination criterion.

The algorithm was tested on 11 benchmark problems with 8-24 decision variables and two objectives. In addition, a groundwater remediation problem with 6-24 variables and two objectives was also solved. The algorithm was compared with NSGA-II and ParEGO using hypervolume in 200-400 function evaluations. The proposed algorithms obtained better performance than others in the given number of function evaluations.

In [102], an algorithm called surrogate assisted local search memetic algorithm (SS-MOMA) was proposed, where RBF as a metamodel was build on the single objective problem after converting multiobjective optimization problem using a scalarization function. Two common ways of scalarization i.e. Tchebycheff and weighted sum were tested, where weights were generated randomly and the reference point in Tchebycheff scalarization was the current individual objective function values. After generating the offspring population in step 6, metamodels were built locally for each individual. For instance, if the offspring population of size 100 was generated, one metamodel for each individual (i.e. 100 in total) was built. As multiobjective optimization problem was converted into single objective using the scalarzing function, SQP algorithm was used to obtain the solutions. All solutions thus obtained were re-evaluated with the original functions and added to the training archive. Afterward, these solutions were combined with the parent individuals and non-dominated sorting [31] was performed to obtain the population for the next generation. In this way, after every generation metamodels were updated with a number equal to the population size used.

The algorithm was tested on three benchmark problems with 15 variables and two objectives. The algorithm was not compared with any other algorithm. However, in 1606 function evaluations, Tchebycheff scalarization performed better than weighted sum in terms of generational distance [28] and one diversity metric used.

In Datta and Regis [27], RBF as metamodels were built for every objective and constraint function in step 5. A $(\mu + \lambda)$ evolution strategy mutation operator was used to generate new individuals in step 6 which were then evaluated using the metamodels in step 7. After using metamodels for each objective and constraint functions, feasible solutions were found and a nondominated sorting was performed on these feasible solutions. The best individuals i.e. individuals in the first front were re-evaluated with the original functions to update the metamodels in step 9 and added to the training archive in step 10. Therefore, the maximum number of individuals to be updated was λ. In addition, initial training of metamodels in step 5was performed without considering any feasibility or infeasibility of solutions. However, the authors clearly mentioned that the algorithm is not expected

to work well on problems when the feasible region is empty.

The algorithm was tested on 15 benchmark problems with 2-15 decision variables, 2-5 objective functions and 2-13 constraint functions. In addition it was also tested on a manufacturing and robotics problem with 3-7 decision variables, 2-5 objectives and 2-8 constraints. Hypervolume was used to compare the performance of the algorithm against constrained version of $(\mu + \lambda)$ evolution strategy [9] and NSGA-II. The algorithm performed better in 500-5000 function evaluations.

In [114], RBF was used as a metamodel for each objective and constraint function. It was assumed that at least one feasible solution is available to train the metamodels in step 5. To create new individuals in step 6, two different approaches were tested. In the first one, uniform random individuals were generated over the search space and in the second one, individuals were generated by adding Gaussian perturbation centered at the nondominated individual that has the most isolated objective function values. The most isolated individuals is identified by measuring the distance among nondominated individuals. Metamodels were then used to approximate the objective and constraint function values in step 7. Afterward, nondominated solutions with minimum constraint violations were found. Among these individuals, one individual was selected for re-evaluations in step 9 to update the metamodels. One individual was selected by the weighted sum (with equal wights) of two criterion. One criterion was the distance of solutions (in the decision space) obtained in step 7 from the individuals in the training archive and the second criterion was the distance from the nondominated individuals from the last generation in the objective space. Selected individual was then re-evaluated and added to the archive in step 10.

The algorithm was tested on 28 benchmark problems with 2-5 objectives, 2-15 decision variables and 1-11 constraint functions. the algorithm was compared with its two different versions (based on the generation of individuals in step 7), NSGA-II and DirectMultiSearch(DMS) [26] using hypervolume as the performance indicator. Overall, the version where individuals were generated using Gaussian distribution performed better in the given number of function evaluations.

3.4 Algorithms based on multiple metamodels

In this subsection, algorithms using multiple metamodels are discussed. These metamodels are used independently to predict objective and/or constraint functions.

In [52], three independent case studies were performed, where three different metamodels (polynomial approximation, Kriging and radial basis function) were used with NSGA-II. The metamodel is built for each objective function in step 5. The nondominated individuals obtained after this step are improved using a local search algorithm with sequential quadratic programming. To perform local search, a variant of ϵ-constraint algorithm [93] is used i.e. one of the objectives is optimized and other objectives are converted to equality constraints. The improved individuals are combined with individuals from step 7 and dominated and duplicated individuals are eliminated. To update the metamodel in step 9, a fixed number of individuals is selected from the remaining individuals using K-means clustering in the objective space for re-evaluation using the original functions. These individuals are then added to the archive in step 10. The size of the archive is not fixed in this algorithm.

This algorithm was used to solve a heat sink MOP with two objectives and three decision variables. Three different studies were performed to compare the results while using different metamodels. In the first one, nondominated individuals were identified while using each metamodel involving steps 1-8 (i.e. without updating the metamodel) of the function approximation framework. Five representative individuals among nondominated individuals were selected using K-means clustering and re-evaluated with the original functions. The proposed algorithm with Kriging gave the least error in objective function values for these five individuals. In the second study, five representative

individuals obtained using each metamodel were re-evaluated with other metamodels. For example, individuals obtained while using polynomial approximation were re-evaluated with Kriging and radial basis function. Individuals obtained while using Kriging when re-evaluated with other metamodels gave the least error in objective function values. In the third one, nondominated individuals were obtained utilizing steps 1-10 (i.e. by updating the metamodel in steps 9 and 10) and results were compared graphically in the objective space with the first study. While using Kriging and radial basis function, better results were obtained when compared to results of the first study and while using polynomial approximation, similar results were obtained. However, the proposed algorithm was not tested using NSGA-II without any metamodel and the efficiency of the algorithm was not mentioned in terms of computation time or the number of function evaluations.

In [6], five different radial basis functions (RBFs) with different basis functions were used as metamodels along with the EA MODE-LD+SS [5]. However, these metamodels were used independently for each objective function but individuals from each metamodel evaluation were used to update all metamodels as discussed next. A metamodel is built for each objective function in step 5. In step 9, to select the individuals for updating the metamodel, one individual from each metamodel evaluation is selected for re-evaluation using the original functions and added to the archive in step 10. To select one individual, a set of uniformly distributed weight vectors $(\lambda^1, \lambda^2, \ldots, \lambda^N)$ (where N is the population size of EA) is defined. Next, from each metamodel evaluation, the individual is selected that minimizes the Chebyshev scalarizing function [93] given by $\max_{i=1,\ldots,k}(\lambda_i{}^j|\hat{f}_i(x^j) - f_i^*|)$. The values of objective functions obtained after step 7 and the minimum value of the objective function in the population of EA at the current generation are represented by $\hat{f}$ and f^*, respectively. The authors mentioned that this updating criterion can balance the accuracy of the metamodel and the diversity among individuals. The size of the archive is fixed in this algorithm and extra individuals are removed from it based on their rank.

This algorithm was tested on five different aerodynamic shape optimization problems with 2-3 objectives and 12 decision variables. Hypervolume was used as the comparison criterion for the results of MODE-LD+SS with and without metamodels. The proposed algorithm gave a better hypervolume in fewer function evaluations for all five optimization problems.

In Palar et al. [101], SS-MOMA described in Section 3.3.2 was tested with different metamodels for different problems i.e. RBF with linear kernel function for ZDT problems and Kriging for UF problems because of different function landscape of the problems. Using a particular technique for a problem with an assumption that function landscape is known a priori raises the question of applicability of the algorithm to black-box problems. In addition, one modification of achievement scalarizing function used, where reference point was replaced by the upper bounds of the objective function values at the current generation was also tested. An another locals search algorithm called random mutation hill climber [1] was also tested after building the metamodels for each objective function. To handle constraints, metamodels were first built for each objective function and after scalarization, a constrained SQP algorithm was used to solve the single objective optimization problem.

The algorithm was tested on seven biobjective benchmark problems with 8-15 decision variables and a biobjective airfoil optimization problem with 16 variables and one constraint. Different versions of the algorithm with different scalrzing functions were compared with each other also with NSGA-II using IGD. Overall a low number of function evaluations were used to obtain a given IGD value when Tchebycheff function with reference point as the individual objective function values was used.

3.5 Algorithms based on ensemble of metamodels

In this subsection, we discuss algorithms using ensemble of metamodels. As defined in Section 2, either the weights are given to predicted output of the metamodel or a metamodel is selected based on its accuracy. In [84], an ensemble of metamodels (Kriging, polynomial regression and radial basis function) was used with the proposed generalized surrogate multiobjective memetic algorithm (GS-MOMA). In this algorithm, the offspring population is generated in step 6 before building the metamodel. A fixed number of individuals, equal to $n+(n+1)(n+2)/2$, (where n is the number of decision variables) is selected from the archive to build an ensemble of metamodels and a separate polynomial regression metamodel in step 5. The individuals are selected in the decision space using the Euclidean distance between the individuals in the offspring population and the individuals in the archive. The individuals which are closer to offspring individuals are used to build the metamodels.

The ensemble fitness values of offspring individuals are calculated as $F_{ens}(x) = \sum_{j=1}^{m} \omega_j \overline{F^j}(x)$, where ω_j is the weight coefficient for fitness values $\overline{F^j}(x)$ of the jth metamodel (step 7). The weight coefficient is assigned based on the accuracy of the metamodels which is calculated using root mean square error. A local search algorithm (sequential quadratic programming) is also used for single objective optimization of $\overline{F^j}(x)$ to improve the individuals evaluated with the metamodel. The best found individuals after this step and the local search algorithm are combined with the parent population and a selection mechanism is used to select individuals for the next generation. To update the metamodels, an offspring population is generated and individuals are selected from the initial archive based on the mechanism used to build the metamodel. Steps 9 and 10 are not applicable in this algorithm and the size of the archive is fixed.

This algorithm was tested on six benchmark problems (labeled as MF problems in the paper) with 2-3 objectives and with 10-50 decision variables. Three performance criteria namely generalized distance [138], maximum spread [152] and hypervolume ratio [137] were used for comparison between GS-MOMA and NSGA-II without a metamodel. GS-MOMA performed better in all performance criteria for all problems in the same number of function evaluations.

In Singh et al. [127], a surrogate assisted simulated annealing (SASA) algorithm was proposed, where an ensemble of metamodels (quadratic polynomial and radial basis function) was used with constrained Pareto simulated annealing (C-PSA) as an EA. In this algorithm, one of two different metamodels is selected to evaluate individuals based on their accuracy which is calculated using the root mean square error. A fixed number of recently evaluated individuals in stage 1 is used to create the metamodels in step 5. In what follows, for each objective function, either one of the metamodels or the original functions are used to evaluate the offspring individuals in step 7 and thus, an adaptive evolution control strategy is used. In other words, if none of the metamodels is accurate enough (accuracy is compared with a predefined parameter), the original functions are used. To select the individuals for updating the metamodel in step 9, nondominated individuals after step 7 (in case a metamodel is used) are re-evaluated with the original functions and added to the archive in step 10. Dominated individuals are removed from the archive and the remaining individuals are used to update the metamodels. The size of the archive is fixed here and clustering is used via a linkage algorithm [56] to remove extra individuals.

The SASA algorithm was tested on eight benchmark problems [28] with two objectives, 10 decision variables and 1-2 constraints. The hypervolume and displacement metric [11] were used to compare SASA with NSGA-II and the infeasibility driven evolutionary algorithm (IDEA) [113]. For the same number of function evaluations, SASA performed better than the other algorithms for seven problems and for one problem, IDEA performed the best in both performance criteria.

In [90], a similar algorithm to MOEA/D-EGO was proposed, where radial basis function was used as the metamodel. This algorithm is known as MOEA/D-RBF, where the metamodel is built

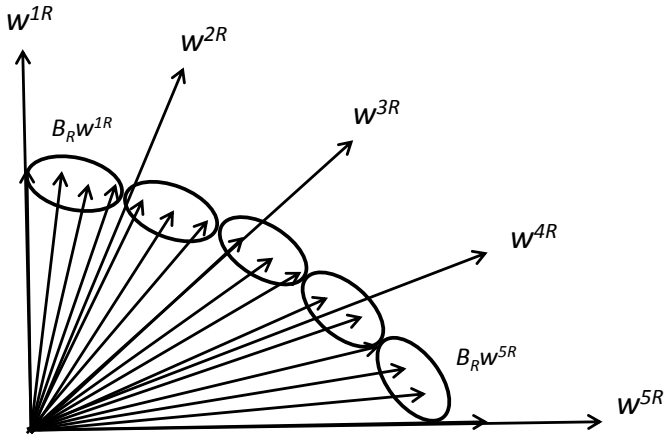

Figure 2: Association of weight vectors from W to W_R

for each objective function instead of scalarized objective function. An ensemble of metamodels (RBF with three different basis functions) is used for each objective function in step 5. A weighted sum of predictions from each metamodel is used in the ensemble of metamodels and the weights are decided based on the prediction error of the metamodel. To update the metamodel in step 9, the following procedure is adopted.

In MOEA/D let, $W = \{w^1, \ldots, w^N\}$ are a uniformly distributed set of weight vectors and R is the number of points to be re-evaluated in step 9. An another set of uniformly distributed weight vectors $W^R = \{w^{1R}, \ldots, w^{NR}\}$ is defined such that size of W^R ¡ size of W as shown in Figure 2. For each $w^{iR} \in W^R$, a neighborhood is defined $B_R(w^{iR}) = \{w^1, \ldots, w^{Na}\}$, such that $w^1, \ldots, w^{Na} \in W$ are the Na closest weight vectors from W to w^{iR}. After this, from each neighborhood, an individual which minimize the single objective optimization problem used in MOEA/D is selected for re-evaluation and added to the archive in step 10. In this algorithm, a penalty boundary intersection approach [146] is used for decomposition of the MOP into single objective optimization problems. The size of the archive is fixed in this algorithm and the same procedure mentioned above is applied to eliminate extra individuals from the archive.

MOEA/D-RBF was tested on five benchmark (two objectives and 8-30 decision variables) and one real-world problem (two objectives and 11 decision variables). It was compared with MOEA/D (for both benchmark and real-world problems) and MOEA/D-EGO (only for benchmark problems) with hypervolume as the comparison criterion. The proposed algorithm obtained better performance in 5 out of six problems in the same number of function evaluations.

3.6 Comparison of function approximation based algorithms

In what follows, a comparison of function approximation based algorithms discussed so far is presented in Figure 3. This figure represents the number of papers with respect to the metamodel (upper part of the figure), EA (lower part of the figure) and evolution control strategy used. The references for these three criteria are mentioned inside the bars of the figure. One should note that some algorithms mentioned in this section were not tested on computationally expensive MOPs. These algorithms are still cited here as they have been proposed for considering the computational burden in any MOP.

The number of papers which used benchmark and real-world problems is also mentioned in Figure 3. Seven algorithms [84, 104, 107, 110, 127, 131, 148] were tested on benchmark problems, seven algorithms [6, 48, 52, 59, 75, 94, 128] on real-world problems and seven algorithms [14, 81, 83, 85, 86, 90, 151, 18] on both. The efficiency of the different algorithms in terms of computation time or number of function evaluations reduced is very important, especially in the case of real-world problems. In some cases, the authors mentioned that it was practically difficult to do optimization without approximations due to high computation time and the algorithm was not compared with an EA without any metamodel.

As far as selection of a metamodel is concerned, as mentioned in Section 3.1, there is no general rule or correlation between a metamodel for approximation and a particular problem to be solved. As shown in Figure 3, single metamodel is used more than multiple metamodels and ensemble of metamodels. In algorithms using single metamodel, Kriging [59, 83, 81, 86, 110, 128, 148, 151] has been used more than other metamodels such as neural networks [14, 76, 94, 104, 131], support vector regression [48, 107] and polynomial regression [85].

In [151], Kriging was used as a metamodel and the authors mentioned that within different metamodels studies to date, perhaps the most common type of metamodel used is the Kriging model. In [14], radial basis functions (RBFs) were used and the authors mentioned that RBF was a kind of approximation having a very good approximation accuracy. In [6], the authors mentioned that RBFs were very powerful functions to represent complex fitness landscapes. In addition, the authors mentioned that Kriging had a strong mathematical basis, and is probably one of the most powerful interpolation algorithms currently available.

Managing the metamodels or evolution control is also very important as it affects the performance of the metamodel used. As shown in Figure 3, the fixed evolution control strategy was used more than the adaptive evolution control strategy. As mentioned in Section 2, using an adaptive evolution control strategy depends on the accuracy of the metamodel used and using a fixed evolution control strategy implies that either the metamodel is accurate enough for approximation or the metamodel accuracy is not important or not checked. There are only five algorithms [86, 94, 95, 127, 151], where adaptive evolution control was used. For instance in [86], after using Kriging models, uncertainty of the approximated values was compared with a predefined value and if that uncertainty was acceptable, then only Kriging models were further used otherwise original function were used. A similar strategy was followed in [94], where accuracy of neural networks was measured after every generation and compared with a predefined value. In [95], uncertainties of offspring was compared with the uncertainties of parents and based on the comparison, the decision was made to use Kriging models or original functions. Singh et al. [127] and Zhu et al. [151] the approximation accuracy of metamodels for using them in subsequent evaluations. Moreover, the training time for different metamodels was not considered in many papers though training time can be substantially high in some cases, particularly, when the metamodel approximation accuracy is important. Among EAs, dominance based algorithms are more widely used than other algorithms e.g. decomposition based or indicator based. Moreover, in dominance based EAs, NSGA-II was used more often than other EAs.

3.7 Problem and fitness approximation

In addition to function approximation, problem and fitness approximations are also used in the literature to reduce the computation time in multiobjective optimization problems. In problem approximation, the original problem is replaced by a simplified problem which is faster to solve. The main goal in problem approximation is to reduce the computational complexity of the problem. For example, in computational fluid dynamics or structural analysis, the governing equations can be modified

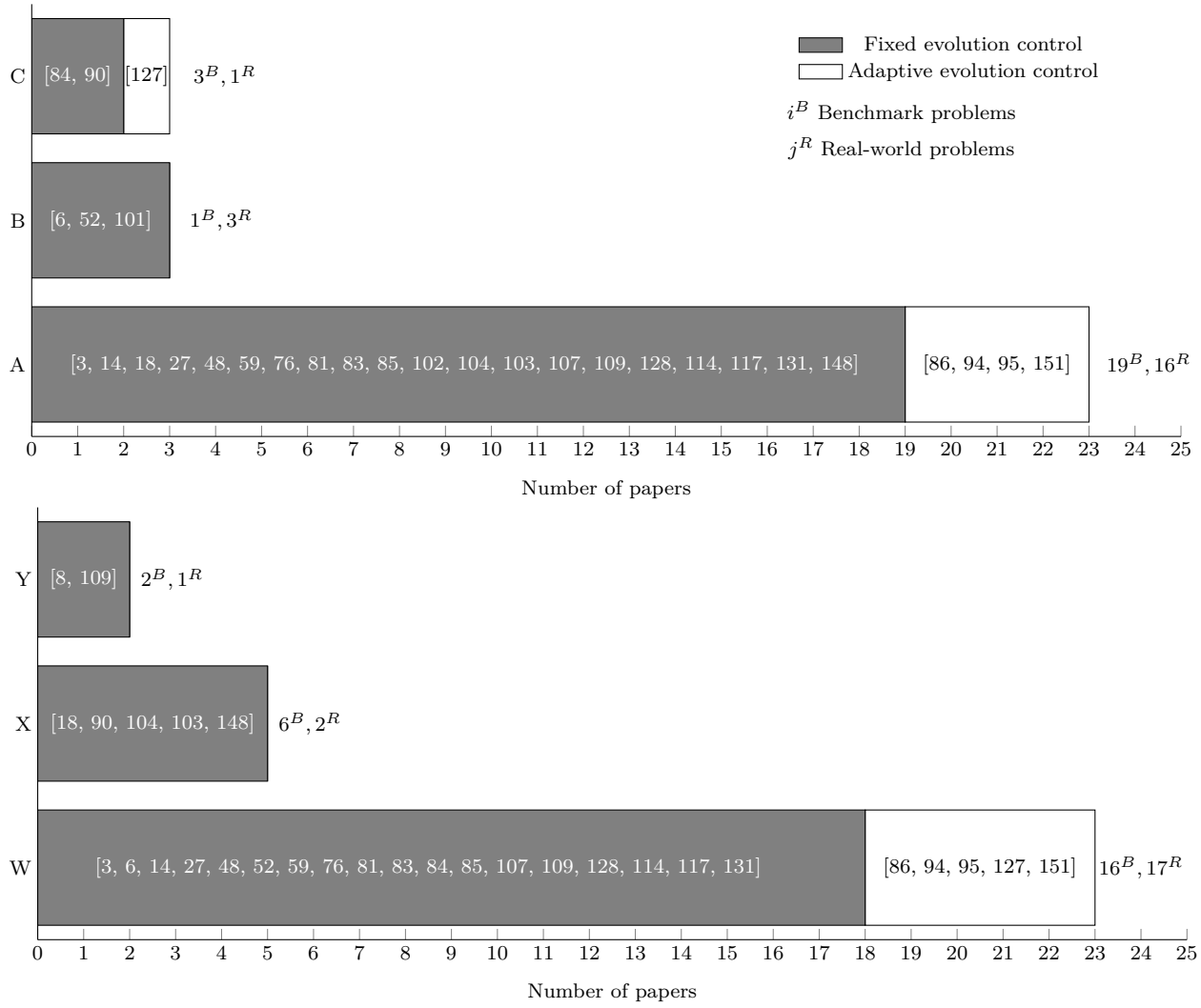

Figure 3: Comparison of different algorithms considering metamodel (upper chart), EA (lower chart) and evolution control strategy used and characteristics of optimization problem considered, [A,B,C] = [single, multiple, ensemble] of metamodels based algorithms; [W,X,Y] = [Dominance, Decomposition, Indicator] based algorithms

to reduce the computation time. As an example, replacing 3-dimensional Navier-Stokes equations by 2-dimensional Euler equations [78] reduces the computational complexity of the problem. In [78] and [79], Euler equations were used instead of Navier-Stokes equations to solve aerodynamic shape optimization problems. A similar approach was followed in [92, 100], where 2-dimensional Navier-Stokes equations were used for solving aerodynamic shape optimization problems.

Recently, data-driven optimization algorithms [142] are also proposed to solve problems, where an analytical forms or simulations models for the objective functions are not available and some data is available obtained through some physical experiments. Therefore, to obtain Pareto optimal solutions, one has to rely upon the data available. In Wang et al. [142], a MOP was formulated using the data available from a trauma system. In addition, the authors found out that accuracy and size of the data used in optimization are conflicting.

In fitness approximation, we have algorithms which use metamodels for approximating some element instead of objective functions. For instance, approximating a rank, classifications of individuals into nondominated and dominated sets and distance to the nondominated individuals. Moreover, we consider the papers in the categories of fitness inheritance and fitness imitation (defined in Section 2) in this section.

Fitness inheritance was originally proposed in [129] to improve the performance of genetic algorithms. Two types of fitness inheritance were proposed in that paper. In the first one, known as averaged inheritance, the fitness values of the offspring were calculated by taking the average of the fitness values of the parents. In the second one, known as proportional inheritance, weighted average of the fitness values of the parents were assigned to offspring and weights were assigned according to common elements between offspring and parents. This algorithm was tested on OneMax problem [2] and the algorithm with fitness inheritance gave better performance (a graphical comparison was performed) than without inheritance in fewer function evaluations.

In what follows, some papers are cited here which use the concept of fitness inheritance. One should note that all these algorithms were not tested on computationally expensive MOPs. In [15], analytical algorithms for computing convergence time and population size were proposed for using the fitness inheritance in MOPs. In [34], the fitness inheritance was used with a binary genetic algorithm (GA) for the ZDT benchmark problems. The authors found a similar performance of the binary GA with and without using the fitness inheritance for problems having convex and continuous Pareto fronts. In case of nonconvex and discontinuous Pareto fronts, the binary GA without inheritance performed better and the authors mentioned that fitness inheritance can only be applied to problems having convex and continuous Pareto fronts. In contrast, in [115], the authors found out that the fitness inheritance can also be applied to problems having nonconvex and discontinuous Pareto fronts.

In Loshchilov et al. [87], a metamodel was used to predict the class for the individuals. This class was the indication that individuals were either nondominated or dominated. This algorithm is known as Pareto-SVM, where one class SVM is used for the classification of individuals in the decision space into nondominated and dominated. In addition, support vector regression is used to predict those individuals (after classification in the decision space) to a target value in the objective space. Aggregate surrogate metamodel terminology is used in the paper as two metamodels are used, one in the decision space and another in the objective space. Instead of binary classification in the decision space, all nondominated individuals mapped to the single value ρ with tolerance ϵ and all dominated individuals are mapped to $]-\infty, \rho-\epsilon[$. In this way, the individuals belonging to $[\rho+\epsilon, +\infty[$ are in the unexplored region. In step 9, an updating criterion for the metamodel is inspired from EA PESA-II [23]. Firstly, individuals obtained after step 7 are added to the archive and duplicate individuals are removed. Then, the objective space is partitioned into a fixed number of equal sized hyperboxes and one individual or nondominated individual (if the box contains nondominated) is selected from

each box randomly. These individuals are re-evaluated using the original functions. The size of the archive is fixed in this algorithm. In addition, to generate offspring in step 6, firstly, a fixed number of individuals (more than the population size) is generated using the variation operators. Then a fixed number of individuals is selected based on their distance to the current nondominated individuals in the decision space.

The proposed algorithm was tested on eight benchmark problems with two objectives and 10-30 decision variables. Hypervolume was used as the performance criterion for comparison of Pareto-SVM with $(\mu + \lambda) - S-$NSGA-II [36] and $\mu \times (1 + \lambda)-$MO-CMA-ES [53]. Pareto-SVM obtained similar performance in fewer function evaluations.

In [88], Pareto-SVM was extended to a rank based aggregate surrogate model. In this algorithm, instead of two classes, various classes were obtained in the decision space based on the rank of individuals. In this way, nondominated individuals at the current generation are not constrained to the bounds. A pairwise dominance relation was used to classify individuals. For example, two possibilities exist, if an individual dominates another individual or is dominated by it. The case for nondominated individuals is mentioned as the future work. Other details are the same as in Pareto-SVM.

The proposed algorithm was tested on the same benchmark problems. It was compared with $(\mu + \lambda) - S-$NSGA-II, $\mu \times (1 + \lambda)-$MO-CMA-ES and Pareto-SVM with hypervolume as the performance criterion. In comparison to Pareto-SVM, it performed similar and in comparison to other two algorithms, it performed better in fewer function evaluations.

In [105], instead of predicting objective functions, a metamodel was used to predict the distance to the current nondominated individuals in the decision space. Three metamodels, linear regression, support vector regression and multilayer perceptron were used independently. This algorithm is known as ASM-MOMA, where the metamodel is built for the distance to the nondominated individuals in step 5. In addition, a local search algorithm is used to improve the individuals obtained after step 7. To update the metamodel in step 9, nondominated individuals at the current generation are added to the archive in step 10. The size of the archive is fixed in this algorithm and extra individuals are removed from it randomly.

ASM-MOMA was tested on four benchmark problems (two objectives and 15 decision variables) and compared with NSGA-II and IBEA [154] with hypervolume as the performance criterion. The proposed algorithm obtained similar performance in fewer function evaluations than other algorithms while using linear regression as the metamodel.

In Pilát and Neruda[106], ASM-MOMA was extended to high dimensional problems (in the objective space) with two different additional elements. Instead of using one global metamodel, multiple local metamodels were used. This algorithm is known as LAMM-MMA, where the training data in the decision space is partitioned into different sets based on the distance of individuals to the current nondominated individuals. Different from ASM-MOMA, the algorithm allocates weights to samples based on the distance to nondominated samples in the decision space. Other details are the same as in ASM-MOMA. This algorithm was tested on four benchmark problems with 5-15 objectives and 20 variables and compared with IBEA and ASM-MOMA with hypervolume as the performance criterion. LAMM-MMA performed better than IBEA and similar to ASM-MOMA in the same number of function evaluations.

In [8], a metamodel was used to predict the contribution to the hypervolume instead of the objective functions. This algorithm is known as NN-SS-IBEA, where a metamodel (neural network) is built for the contribution to the hypervolume of individuals in step 5. After evaluating the new individuals in step 7, one individual having the maximum contribution to the hypervolume is re-evaluated in step 9 and added to the archive in step 10. The size of the archive is fixed in this algorithm and extra individuals are eliminated from the archive based on their hypervolume

contribution.

The proposed algorithm was tested on 12 benchmark (2-3 objectives and 8-30 decision variables) and one real-world problem (two objectives and 11 decision variables). The algorithm was compared with MOEA/D-RBF [90] and IBEA [154]. The hypervolume was used as the comparison criterion. In case of benchmark problems, NN-SS-IBEA performed better in eight out of 12 problems with the same number of function evaluations. In case of the real-world problem, the proposed algorithm performed better than others for a given number of function evaluations.

A similar algorithm to [88] was proposed in [10], where nondominated individuals were also used during classification. In this algorithm, a metamodel is built for three classes while doing pairwise dominance comparison. Here, three possibilities exist, one solution dominates another, one solution is dominated by another or both solutions are nondominated. Ten different classification algorithms were used independently. The metamodel is updated in step 9 after a fixed number of generations with the previously evaluated individuals in step 7. The information on the size of the archive is not mentioned.

The proposed algorithm was tested on three benchmark problems (two objectives and 5-20 decision variables). However, this algorithm was not compared with any other algorithm, but the results of ten classification algorithms were compared using the training time and the accuracy as the comparison criteria. The authors mentioned that SVM, classification trees, k-neural network and quadratic discriminant analysis were the most preferred among other metamodels as these metamodels found the knee region in the objective space.

In [123], a metamodel was used to predict the rank of individuals in the objective space. In this algorithm, the metamodel is used to create boundaries between fronts in the objective space. As shown in Figure 4, θ_1 and θ_2 represent the two boundaries created using individuals of three different fronts. The ranks of offspring individuals are then predicted with an indicator function mentioned in the paper with boundaries created (steps 5-7). For example, individuals that lie below θ_1 are of rank 1, individuals between θ_1 and θ_2 are of rank 2 and individuals above θ_2 are of rank 3. The first rank individuals are re-evaluated using the original functions in step 9 and added to the archive in step 10. The size of the archive is not fixed in this algorithm.

This algorithm was tested on 19 benchmark problems (the authors did not mention explicitly the number of objectives and decision variables) and compared with NSGA-II, SPEA2 [155] and MOEA/D [146]. Three performance criteria (generational distance, inverted generational distance and hypervolume [35]) were used to compare the proposed algorithm and EAs for the same number of function evaluations. Out of 19 problems, the proposed algorithm performed better than the other algorithms in 16, 14 and 12 problems in generational distance, inverted generational distance and hypervolume, respectively. In the rest of the problems, MOEA/D and NSGA-II performed better than the proposed algorithm.

In [133], a study was performed to compare four metamodelling techniques and three different scalarizing functions using two different approaches of approximation. Therefore, for each approximation, 12 different experiments were performed on different problems. A data set of prefixed size was used for training and validation. The error after the validation was used to compare different studies. As no algorithm was proposed in the current study, other steps of the function approximation were not applied. Four metamodels were based on Kernel ridge regression (KRR), Kriging, Generalized linear models and Tree-based regression. Three different scalarizing functions tested were based on Tchebychef (TCH), Boundary intersection (PBI) and weighted sum (WS). As mentioned, two different approaches of approximation were used, in the first one, the metamodel was used for the scalarizing funcion and in the second one, the metamodel was used to approximate the rank of individuals.

In the study, experiments were conducted on DTLZ suite with 4-10 objectives and 8-19 decision

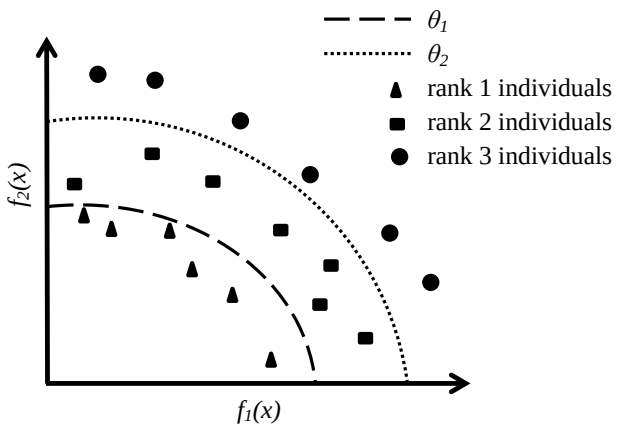

Figure 4: Assignment of ranks to individuals

variables. In both types of approximation used, Kriging performed the best. In comparing different scalarizing functions, TCH performed the best in the first type of approximation and WS in the second type of approximation. Developing a algorithm was considered as a future work.

4 Discussion

In this section, we first summarize promising elements that we found in the algorithms proposed in the literature. Furthermore, we also discuss ways which can be used to design enhanced EAs to handle computationally expensive MOPs. Finally, we discuss the main issues we have observed in the algorithms in the literature with respect to using an approximation in an EA and the numerical settings used to test their efficacy.

4.1 Promising elements in the literature

We discuss here promising elements with respect to the choice of the metamodel to be used, building/updating the metamodels and using different types of approximations together to reduce the computation time. Furthermore, we discuss the potential of using hybrid algorithms with function approximation based algorithms to enhance the rate of convergence of EAs near Pareto optimal solutions. Also, one can use these elements while designing an algorithm for using an approximation with EA to reduce the computation time.

Often when metamodels are used, the type of metamodel to be used is a random choice. It is often difficult to know a priori the best metamodel to be used in the solution process. An ensemble of metamodels (see e.g. [84] and [127]) could be an easy fix, where the algorithm is not limited to one single metamodel. In [84], weights are assigned to the fitness values predicted by each metamodel and the weighted sum is then used to get the final fitness value. In [127], one metamodel having the highest accuracy is used for evaluating individuals of the EA population. Moreover, in [6], the authors present an improved way of using multiple metamodels that focus on different parts of the Pareto front. These metamodels are updated using some solutions that were more and less accurate. Selection of these solutions enables the algorithm proposed in [6] to converge faster and explore the search space for promising candidate Pareto optimal solutions.

Table 2: algorithm reference, problems used, dimensions of the problems and metamodel used in different algorithms *ref* : reference, *problem* : multiobjective problem(s), *var/obj/cons* : number of decision variables, objectives and constraints, *metamodel* : metamodel used, *metric* : performance metric used to measure the performance of the algorithm, GD: generational distance, IGD: inverted generational distance

ref	*problem* or *test suite*	*var/obj/cons*	*metamodel*	*performance metric*
[6]	Aerodynamic shape optimization	12/2-3/0	NN	Hypervolume
[8]	ZDT [153], DTLZ [32]	8-30/2-3/0	NN	Hypervolume
	Aerodynamic shape optimization	11/2/0	NN	Hypervolume
[10]	ZDT	5-20/2/0	several classification algorithms	Approximation error and training time
[14]	ZDT, SCH [122], FON [40], KUR [77], TEST [143]	1-5/2/2	NN	GD and Spread
	Design of vehicle door	5/2/2	NN	No performance metric
[18]	DTLZ, WFG [51]	10/3-10/0	Kriging	IGD and Hypervolume
	Free-radical polymerization	4/3/0	Kriging	IGD
[48]	Rigid frame design	10/2/0	SVR	Visually
[52]	Microchannel heat sink	3/2/0	PR, Kriging, NN	Visually
[59]	Fatigue strength assessment for ship	17/2/11	Kriging	No performance measure
[76]	Coastal aquifer management	8/2/0	NN	Visually
[81]	ZDT, OSY [99], TEST	3-6/2/0	Kriging	Visually
	Cabinet problem	2/2/0	Kriging	Visually
	Gear train problem	4/2/0	Kriging	Visually
	Distillation column	4/2/4	Kriging	Visually
[83]	TEST, FON, KUR	3-6/2/0	Kriging	Visually
	Cabinet problem	2/2/0	Kriging	Visually
	Gear train problem	4/2/0	Kriging	Visually
[84]	ZDT	10-50/2	PR, Kriging, NN (ensemble)	GD, Spread, Hypervolume
[85]	TEST, FON, KUR	2-3/2/2	PR	GD
	Car front floor forming	3/2/1	PR	Visually
	Car sheet metal forming	3/2/0	PR	Visually
[86]	ZDT, FON, POL [108], QV [112]	2-30/2/0	Kriging	IGD, Error ratio, Span, Crowding distance
	Stiffed panel problem	6/3/0	Kriging	Visually
[87]	ZDT	10-30/2/0	SVR,SVM	Hypervolume
[88]	ZDT	10-30/2/0	SVR,SVM	Hypervolume
[90]	ZDT	8-30/2/0	NN (ensemble)	Hypervolume
	Aerodynamic shape optimization	11/2/0	NN (ensemble)	Hypervolume
[94]	Iron induration process	22/2/3	NN	Visually
[104]	ZDT	10-60/2/0	NN	Additive epsilon and R2 indicator
[105]	ZDT	15/2/0	LR,SVR,NN	Hypervolume
[106]	DTLZ	20/5-15/0	LR,SVR,NN	Hypervolume
[107]	ZDT, WFG	10-30/2/0	SVR	Hypervolume
[109]	ZDT, DTLZ	3-6/2-5/0	Kriging	Hypervolume, R2 indicator, Unary epsilon indicator
[123]	ZDT, DTLZ, UF [147]	-	SVM	GD, IGD, Hypervolume
[127]	CTP [28]	10/2/2	PR,NN (ensemble)	Hypervolume, Displacement metric
[128]	Nowacki bean problem	2/2/5	Kriging	No performance measure
	Double folded stub microwave filter	3/2/7	Kriging	No performance measure
[131]	ZDT	-/2/0	NN	GD, Spread, IGD, Hypervolume
[142]	Trauma system	-/2/2	PR	IGD, Computation time
[148]	ZDT, DTLZ, LZ08 [82], KNO1 [71], VLMOP2 [136]	2-8/2-3/0	Kriging	IGD, Hypervolume
[151]	ZDT	5/2/0	Kriging	IGD, Error ratio, Span, Crowding distance
	Stiffed panel problem	6/2/0	Kriging	Visually

One more promising element observed in some of the algorithms in the literature is to use a combination of different approximations. In [131], fitness inheritance (a type of fitness approximation) is used with function approximation to reduce the computation time. Problem approximation is used with function approximation in [151] to decrease the computational complexity of the problem in question and the numbers of function evaluations. Moreover, the use of multilevel approaches [43, 68] which depend on the problem to be solved needs more attention from both researchers and practitioners. In these approaches, either different solvers were used to solve governing equations (e.g. Navier-Stokes equations) or different problems were solved at different levels. For more details about multilevel approaches, see [43].

In addition, metamodels have been used in the literature to predict the quality of solutions instead of the objective functions. For instance, in [8], the metamodel was used to predict the hypervolume contribution of the individuals. In [105, 106], the metamodel was used to predict the distance from the current nondominated individuals. Moreover, in [10, 87, 88], the metamodel was used to classify the individuals into different classes based on their dominance relations. In [123], the metamodel was used to predict rank of individuals by creating boundaries between individuals in the objective space. This way of using a metamodel other than function approximation could be affective as instead of actual objective functions, quality of individuals (e.g. using hypervolume or rank) is predicted.

Hybrid EAs with a combination of global and local searches are used to enhance the rate of convergence towards the Pareto front [54, 55, 125, 126]. However, such algorithms are not commonly used together with function approximation to handle computationally expensive MOPs. In the literature e.g. in [52, 84, 134], a local search is used with function approximation. This way of using local search algorithm with function approximation can inherit advantages of both hybrid algorithms and function approximation based algorithms, i.e. fast convergence and reducing the numbers of computationally expensive function evaluations.

4.2 Performance metrics and benchmark problems

It is also important to point out that different algorithms use different performance metrics for measuring their efficiency compared to other algorithms. These metrics e.g. generational distance (GD), inverted generational distance (IGD) and hypervolume (or S metric) are widely used in the literature. For definitions, see [20, 28]. In Table 2, we mention the performance metrics used in different articles. As can be seen in the table, some articles do not use any performance metric which is notated by no preference measure and some articles compare solutions obtained with different algorithms visually e.g. by plotting a scatter plot of nondominated solutions.

Moreover, different algorithms have been tested on different benchmark problems as can also be seen in Table 2. We mention the name of the benchmark problem suite e.g. ZDT, DTLZ and WFG in the table. For more details about these problems, see [20]. From the names of the suites, one can observe the trend of using different benchmark problems in the field of surrogate-assisted multiobjective optimization.

4.3 Guidelines for selecting an algorithm

For practitioners and engineers, it is very important to select an appropriate algorithm for solving a real-world problem. In the literature, different algorithms use different EAs, surrogate techniques and other relevant parameters. Therefore, it is challenging to generalize the efficiency of an algorithm or choose an algorithm. However, one can consider the following points to select an algorithm:

1. dimensions in both objective and decision spaces and

2. budget e.g. maximum number of expensive function evaluations available.

The first point in selecting an algorithm is the dimensions of the problem to be solved both in the objective and decision spaces. One can observe from Table 2 that some algorithms e.g. [6, 8, 86, 136] were tested with up to three objectives, few (i.e. [18, 109, 106]) were tested with more than three and most of the algorithms were tested only on biobjective optimization problems. A similar pattern can be observed for decision variables: only two algorithms i.e. [84, 104] were tested on problems with more than 30 variables. Therefore, references provided in the table give options for selecting an algorithm based on the dimensions of the problem to be solved.

The second key point in selecting an algorithm should be the capability of the algorithm to obtain solutions in a given limited budget. In many real-world problems, the number of expensive function evaluations is limited e.g. because of a time limit and it may be infeasible to do initial experimental runs to select an algorithm. As can be seen in Table 1, some algorithms in the literature use up to 50000 function evaluations to solve benchmark problems and such a high number may not be realistic in solving real problems. Although the efficiency of different algorithms with the number of function evaluations is not visible in most of the articles, one can look at the maximum number of function evaluations used to choose an algorithm.

4.4 Issues with the present algorithms

Let us discuss next the main issues we have observed in the algorithms in the literature related to using an approximation in an EA and the numerical settings used to test their efficacy. These main issues include: 1. type of test problems used (benchmark or real-world), 2. dimensions of the test problems both in objective and decision spaces considered, 3. scarce use of constrained problems for numerical testing, 4. neglecting the training time for fitting the metamodel used when reporting results, 5. less focus on the accuracy of the metamodel, 6. not well explored updating criterion of the metamodel, 7. not well structured ensemble of metamodels, and 8. less emphasis to algorithms that consider problem information when available. To augment the discussion on the issues enumerated above, we present in Table 2 an overview of the test problems considered in the literature with their dimensions both in objective and decision spaces and the type of metamodel used. Issues 1-3 and 4-8 address the shortcomings in numerical testing and solution algorithms used, respectively.

In Table 2, it can be clearly seen that both benchmark and real-world problems have been widely used by researchers to test their proposed EAs. Benchmark problems available in the literature are designed to be simple to implement, fast to evaluate, have known global optimal solutions, pose varied challenges to EAs such as several locally optimal solutions etc. Benchmark problems solved are not computationally expensive and used just to test the efficiency of a particular algorithm. For more details about the characteristics of benchmark problems, see [28, 20]. On the other hand, real-world problems either are actual problems considered in the industry or an emulation of them. Often, real-world problems are computationally expensive to evaluate, have uncertainties in their input and output variables, which in turn does not allow a thorough testing of an EA in practice using several real-world problems. Thus, it could be a good practice, if a numerical test setting involving several benchmark and a few real-world problems could be used to thoroughly test EAs. However, in the literature very few researchers have adopted this practice. One reason for this could be attributed to the lack of open availability of real-world problems to all researchers.

Real-world problems in industries often involve a large number of objectives, a large decision space and a large number of constraints. It can be seen in Table 2 that most of the problems considered in the literature had low dimensions both in decision and objective spaces. Additionally, only six algorithms in Table 2 considered constraints besides the bounds for the decision variables. Thus, significant attention is needed towards issues 2 and 3 and more constrained benchmark and

real-world problems need to be included in the numerical test settings. The number of objectives considered did not usually exceed three and thus the applicability of existing surrogate-based algorithms is limited. In the literature, the term many-objective optimization is also used as a synonym for optimizing (usually) more than three objectives. Moreover, the maximum number of decision variables considered is under 60. In fact, only three algorithms were used on more than three objective functions, four algorithms were used for three objective functions and the rest of the algorithms were used for solving biobjective optimization problems.

Furthermore, it is worth noting the connection between the type of metamodels used and the number of decision variables associated with the problem. In one instance, neural network was used when the number of decision variables was high (60). Next, in the decreasing order of popularity are algorithms using ensemble of metamodels, Kriging, support vector regression and polynomial approximation (including linear regression) for handling high dimensional (decision space) problems. What is missing in the literature is a study with each of the algorithms about the efficacy of using different metamodels when the dimensions of the decision space increases, especially in algorithms where the type of the metamodel to be used within the algorithm is not binding. Hence, further research towards dimensions and types of metamodels is needed.

The issues 4-7 concern the use of metamodels. The major concern in the literature is how to alleviate the computational cost of the MOP. However, the training time needed to build/update a metamodel is completely neglected when reporting the results. This is mainly based on the assumption by the researchers that the computation time needed to compute objective and constraint function values is significantly higher as compared to the building/updating time of the metamodel used. This assumption may not hold in all cases and, specifically, when a large amount of data is used to build/update the metamodel.

Most of the algorithms discussed in this survey used a fixed evolution control strategy, i.e. the accuracy of the metamodel is not considered as an important issue. In this survey, we defined the adaptive evolution control, where if the metamodel is not accurate, the original functions are used for evaluating individuals. A desired accuracy for a metamodel can be predefined (e.g. in terms of a parameter) or changed by the user during the solution process. The accuracy can be calculated by different statistical measurements such as root mean square error, coefficient of determination, analysis of variance (ANOVA) etc. [132]. As the accuracy of the metamodel is not considered in many algorithms in the literature and this is a valid future research direction.

The updating criterion of the metamodel can influence the quality of nondominated solutions. Considering the function approximation framework mentioned in Section 3, selected individuals from step 7 are re-evaluated with the original functions. The number and selection criterion of individuals to be used for re-evaluation from step 7 is not well explored in the literature, however, few algorithms considered both exploration and exploitation while updating the metamodel. We consider this as an important issue as selection of these individuals affects the updating time and accuracy of the metamodel, which may finally affect the quality of the set of nondominated solutions generated by the algorithm.

Although we consider an ensemble of metamodels to be a very promising element in the literature, yet, we consider it as an issue as well. In most of the algorithms proposed in the literature, only the accuracy of the metamodel is used as the criterion to select a particular metamodel. However, there can be other criteria that can be considered in addition to the accuracy of the metamodel, e.g. diversity among individuals, training time of the metamodel etc. Considering these criteria for structuring an ensemble of metamodel is a future research direction.

Finally, the information about the MOP such as a different computation time required for evaluating different objective and/or constraint functions must be considered by the EA. Here we suggest two types of problem information that can be considered: firstly, if any objective function is not

computationally expensive, a metamodel need not be created for that function. However, many real-world problems involve black box functions and therefore, an explicit information about them can be difficult to obtain. Secondly, sometimes performing an experiment to calculate the value of an objective function may take less time or may give better results when compared to using a numerical model to evaluate the same. For example in [73], a closed-loop optimization was performed, where values of objective functions are obtained by performing real experiments and selection while crossover and mutation operations are performed by using an EA. For more details about closed-loop evolutionary multiobjective optimization, see [73]. Thus, both types of problem information can lead to significant savings in computation time.

5 Conclusions

In this paper, we have presented a survey of different algorithms to reduce the computation time while solving computationally expensive multiobjective optimization problems. Altogether, 45 algorithms were found in the literature published in years 2008-2016, which were classified based on the type of approximation used, i.e. problem, function or fitness approximation. We found that function approximation or using surrogates is the most common approach for reducing the computation time. Different algorithms were summarized based on the steps of a unified function approximation framework involving an evolutionary algorithm. Additionally, six important challenges were identified, involving using the metamodel, updating the metamodel, training time, type of the metamodel to be used, when to update the metamodel and constraint handling for implementing the proposed approximation framework. Subsequently, the first three important challenges were used in the survey to highlight the differences among different algorithms.

A proper management of the metamodel makes an algorithm efficient to solve a computationally expensive problem. As different algorithms used different strategies to manage the metamodels and solved problems with different characteristics, it is difficult to generalize the efficiency of a method. However, the efficiency of an algorithm can be attributed by 1. number of function evaluations used, 2. dimensions (both in objective and decision spaces) of problems solved and 3. characteristics of the problems solved e.g. multimodality or disconnected Pareto front. As metamodel based algorithms are generally developed for black-box problems, where characteristics of the problems to be solved are not known a priori, one can measure the efficiency of an algorithm by its ability to provide meaningful solutions in a least number of function evaluations. We also provided the dimensions and the number of function evaluations used to solve a particular problem.

We also compared these algorithms with respect to different criteria such as type of metamodel and evolutionary algorithm used, type of problem (benchmark or real-world) solved and evolution control. Some of the major findings were: 1. Kriging and neural networks were the most commonly used metamodels, 2. most of the algorithms were based on dominance based evolutionary algorithms, 3. most of the algorithms solved the problems with no more than three objectives, 4. most of the algorithms were not developed to handle constraints, 5. the number of decision variables was also limited especially when using Kriging, 6. only few algorithms used an ensemble of metamodels, 7. many algorithms were tested only on benchmark problems which were not at all computationally expensive

We discussed problem and fitness approximation based algorithms with respect to their potential towards reducing the computational complexity and the number of function evaluations. We also identified some promising elements and issues among algorithms in the literature. Promising elements involve using an ensemble of metamodels, hybrid algorithms with function approximation based algorithms and different types of approximations together. We plan to address several issues

observed related to shortcomings in numerical testings and algorithms in our future research.

Acknowledgments: The research of Tinkle Chugh was funded by the COMAS Doctoral Program (at the University of Jyväskylä) and FiDiPro Project DeCoMo (funded by Tekes, the Finnish Funding Agency for Innovation) and the research of Dr. Karthik Sindhya was funded by by SIMPRO project funded by Tekes as well as DeCoMo.

References

[1] E. Aarts and J. Lenstra, editors. *Local Search in Combinatorial Optimization.* Princeton University Press, 2003.

[2] D. Ackley. *A Connectionist Machine for Genetic Hillclimbing.* Kluwer Academic Publishers, Boston, 1987.

[3] T. Akhtar and C. Shoemaker. Multi objective optimization of computationally expensive multimodal functions with RBF surrogates and multi-rule selection. *Journal of Global Optimization*, 64:17–32, 2015.

[4] N. Alexandrov, J. Jr., R. Lewis, and V. Torczon. A trust-region framework for managing the use of approximation models in optimization. *Structural Optimization*, 15:16–23, 1998.

[5] A. Arias-Montano, C. Coello, and E. Mezura-Montes. MODE-LD+SS: A novel differential evolution algorithm incorporating local dominance and scalar selection mechanisms for multiobjective optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

[6] A. Arias-Montano, C. Coello, and E. Mezura-Montes. Multi-objective airfoil shape optimization using a multiple-surrogate approach. In *Proceedings of the IEEE Congress on Evolutonary Computation*, pages 1–8. IEEE, 2012.

[7] H. Aytug and S. Sayin. Using support vector machines to learn the efficient set in multiple objective discrete optimization. *European Journal of Operational Research*, 193:510–519, 2009.

[8] N. Azzouz, S. Bechikh, and L. Said. Steady state IBEA assisted by MLP neural networks for expensive multi-objective optimization problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 581–588. ACM, 2014.

[9] T. Bäck. *Evolutionary Algorithms in Theory and Practice.* New York: Oxford University Press, 1996.

[10] S. Bandaru, A. Ng, and K. Deb. On the performance of classification algorithms for learning Pareto-dominance relations. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1139–1146. IEEE, 2014.

[11] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb. A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation*, 12:269–283, 2008.

[12] J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Kluwer, 1981.

[13] J. Branke and C. Schmidt. Faster convergence by means of fitness estimation. *Soft Computing*, 9:13–20, 2005.

[14] G. Chen, X. Han, G. Liu, C. Jiang, and Z. Zhao. An efficient multi-objective optimization method for black-box functions using sequential approximate technique. *Applied Soft Computing*, 12:14–27, 2012.

[15] J.-H. Chen, D. E. Goldberg, S.-Y. Ho, and K. Sastry. Fitness inheritance in multiobjective optimization. In W. B. L. et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 319–326. Morgan Kaufmann, 2002.

[16] T. Chen, K. Tang, G. Chen, and X. Yao. A large population size can be unhelpful in evolutionary algorithms. *Theoretical Computer Science*, 436:54–70, 2012.

[17] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20:773–791, 2016.

[18] T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, and K. Sindhya. A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. *IEEE Transactions on Evolutionary Computation*, to appear, doi: 10.1109/TEVC.2016.2622301.

[19] T. Chugh, K. Sindhya, K. Miettinen, J. Hakanen, and Y. Jin. On constraint handling in surrogate-assisted evolutionary many-objective optimization. In J. e. a. Handl, editor, *Proceedings of the 14th Parallel Problem Solving from Nature-PPSN XIV*, pages 214–224. Springer, 2016.

[20] C. Coello, G. Lamont, and D. Veldhuizen. *Evolutionary Algorithms for Solving Multi-objective Problems*. Springer, New York, New York, 2nd edition, 2007.

[21] C. Coello and G. Pulido. A micro multi-objective genetic algorithm for multi-objective optimizations. In E. Zitzler, L. Thiele, K. Deb, C. Coello, and D. Corne, editors, *Proceedings of the Evolutionary Multi-Criterion Optimization*, pages 126–140. Springer, 2001.

[22] C. C. Coello and G. Lamont, editors. *Applications of Multi-Objective Evolutionary Algorithms*. World Scientific, Singapore, 2004.

[23] D. Corne, N. Jerram, J. Knowles, and M. Oates. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 283–290. Morgan Kaufmann, 2001.

[24] I. Couckuyt, D. Deschrijver, and T. Dhaene. Fast calculation of multiobjective probability of improvement and expected improvement criteria for Pareto optimization. *Journal of Global Optimization*, 60:575–594, 2014.

[25] C. Currin, M. Mitchell, M. Morris, and D. Ylvisaker. A Bayesian approach to the design and analysis of computer experiments. Technical report, Oak Ridge National Laboratory, 1998.

[26] A. L. Custodio, J. Madeira, A. Vaz, and L. N. Vicente. Direct multisearch for multiobjective optimization. *SIAM Journal on Optimization*, 21:1109–1140, 2011.

[27] R. Datta and R. Regis. A surrogate-assisted evolution strategy for constrained multi-objective optimization. *Expert Systems with Applications*, 57:270–284, 2016.

[28] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, Chichester, 2001.

[29] K. Deb, K. Miettinen, and S. Chaudhuri. Toward an estimation of nadir objective vector using a hybrid of evolutionary and local search approaches. *IEEE Transactions on Evolutionary Computation*, 6:821–841, 2010.

[30] K. Deb and P. Nain. An evolutionary multi-objective adaptive meta-modelling procedure using artificial neural networks. In S. Yan, Y.-S. Ong, and Y. Jin, editors, *Proceedings of the Evolutionary Compitation in Dynamic and Uncertain Environments*, pages 297–322. Springer, 2007.

[31] K. Deb, A. Prarap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[32] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. *Scalable Test Problems for Evolutionary Multiobjective Optimization*, pages 105–145. Springer London, London, 2005.

[33] J. Dennis and V. Torczon. Managing approximation models in optimization. In N. Alexandrov and N. Hussaini, editors, *Proceedings of the Multidisciplinary Design Optimization: State-of-the-Art*, pages 330–347, 1995.

[34] E. Ducheyne, B. Baets, and R. Wulf. Is fitness inheritance useful for real-world applications? In C. Fonseca, P. Fleming, E. Zitzler, L. Thiele, and K. Deb, editors, *Proceedings of the Evolutionary Multi-Criterion Optimization*, pages 31–42. Springer, 2003.

[35] J. Durillo, A. Nebro, and E. Alba. The jmetal framework for multi-objective optimization: Design and architecture. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

[36] M. Emmerich, N. Beume, and B. Naujoks. An EMO algorithm using the hypervolume measure as selection criterion. In C. Coello, A. Aguirre, and E. Zitzler, editors, *Proceedings of the Evolutionary Multi-Criterion Optimization*, pages 62–76. Springer, 2005.

[37] M. Emmerich, A. Deutz, and J. Klinkenberg. Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2147–2154. IEEE, 2011.

[38] M. Emmerich, K. Giannakoglou, and B. Naujoks. Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10:421–439, 2006.

[39] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou. Metamodel-assisted evolution strategies. In J. M.-G. et al., editor, *Proceedings of the Parallel Problem Solving from Nature-PPSN VII*, pages 361–370. Springer, 2002.

[40] C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[41] A. Forrester and A. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45:50–79, 2009.

[42] S. E. Gano, J. E. Renaud, J. Martin, and T. Simpson. Update strategies for Kriging models used in variable fidelity optimization. *Structural and Multidisciplinary Optimization*, 32:287–298, 2006.

[43] K. Giannakoglou and I. Kampolis. Multilevel optimization algorithms based on metamodel- and fitness inheritance-asisted evolutionary algorithms. In Y. Tenne and C.-K. Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, pages 61–84. Springer, 2010.

[44] D. Gorissen, I. Couckuyt, P. Demeester, T. Dhaene, and K. Crombecq. A surrogate modelling and adaptive sampling toolbox for computer based design. *Journal of Machine Learning Research*, 11:2051–2055, 2010.

[45] L. Gräning, Y. Jin, and B. Sendhoff. Individual-based management of meta-models for evolutionary optimization with application to three-dimensional blade optimization. In S. Yang, Y.-S. Ong, and Y. Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, pages 225–250. Springer, 2007.

[46] M. Hansen and A. Jaskiewicz. Evaluating the quality of approximation to the non-dominated set. Technical report, Technical University of Denmark, 1998.

[47] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9:159–195, 2001.

[48] M. Herrera, A. Guglielmetti, M. Xiao, and R. Coelho. Metamodel-assisted optimization based on multiple kernel regression for mixed variables. *Structural and Multidisciplinary Optimization*, 49:979–991, 2014.

[49] D. Horn, T. Wagner, D. Biermann, C. Weihs, and B. Bischl. Model-based multi-objective optimization: Taxonomy, multi-point proposal, toolbox and benchmark. In A. Gasper-Cunha, C. H. Antunes, and C. Coello, editors, *Evolutionary Multi-criterion Optimization*, pages 64–78. Springer, 2015.

[50] G. Huang, Q. Zhu, and C. Siew. Extreme learning machine: A new learning scheme of feedforward neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages 985–990. IEEE, 2004.

[51] S. Huband, L. Barone, L. While, and P. Hingston. A scalable multi-objective test problem toolkit. In C. Coello, A. H. Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, pages 280–295. Springer, 2005.

[52] A. Husain and K.-Y. Kim. Enhanced multi-objective optimization of a microchannel heat sink through evolutionary algorithm coupled with multiple surrogate models. *Applied Thermal Engineering*, 30:1683–1691, 2010.

[53] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15:1–28, 2007.

[54] H. Ishibuchi, Y. Hitotsuyanagi, N. Tsukamoto, and Y. Nojima. Use of heuristic local search for single-objective optimization in multiobjective memetic algorithms. In G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, editors, *Proceedings of the Parallel Problem Solving from Nature-PPSN X*, pages 743–752. Springer, 2008.

[55] H. Ishibuchi, Y. Hitotsuyanagi, Y. Wakamatsu, and Y. Nojima. How to choose solutions for local search in multiobjective combinatorial memetic algorithms. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *Proceedings of the Parallel Problem Solving from Nature-PPSN XI*, pages 516–525. Springer, 2010.

[56] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1998.

[57] A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31:264–323, 1999.

[58] H. Jain and K. Deb. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18:602–622, 2014.

[59] B.-S. Jang, D.-E. Ko, Y.-S. Suh, and Y.-S. Yang. Adaptive approximation in multi-objective optimization for full stochastic fatigue design problem. *Marine Structures*, 22:610–632, 2009.

[60] S. Jeong and S. Obayashi. Efficient global optimization (EGO) for multi-objective problem and data mining. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2138–2145. IEEE, 2005.

[61] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9:3–12, 2005.

[62] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1:61–70, 2011.

[63] Y. Jin, M. Olhofer, and B. Sendhoff. On evolutionary optimization with approximate fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 786–793. Morgan Kaufmann, 2000.

[64] Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6:481–494, 2002.

[65] Y. Jin and B. Sendhoff. Reducing fitness evaluations using clustering techniques and neural network ensembles. In K. Deb, editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 688–699. Springer, 2004.

[66] M. Johnson, L. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26:131–148, 1990.

[67] D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[68] I. Kampolis and K. Giannakoglou. A multilevel approach to single and multiobjective aerodynamic optimization. *Computer Methods in Applied Mechanics and Engineering*, 197:2963–2975, 2008.

[69] A. Keane. Statistical improvement criteria for use in multiobjective design optimization. *AIAA Journal*, 44:879–891, 2006.

[70] H.-S. Kim and S.-B. Cho. An efficient genetic algorithm with less fitness evaluation by clustering. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 887–894. IEEE, 2001.

[71] J. Knowles. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10:50–66, 2006.

[72] J. Knowles. Closed-loop evolutionary multiobjective optimization. *IEEE Computational Intelligence Magazine*, 4:77–91, 2009.

[73] J. Knowles. Closed-loop evolutionary multiobjective optimization. *IEEE Computational Intelligence Magazine*, 4:77–91, 2009.

[74] J. Knowles and H. Nakayama. Meta-modeling in multiobjective optimization. In J. Branke, K. Deb, K. Miettinen, and R. Slowinski, editors, *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 245–284. Springer, 2008.

[75] G. Kourakos and A. Mantoglou. Pumping optimization of coastal aquifers based on evolutionary algorithms and surrogate modular neural network models. *Advances in Water Resources*, 32:507–521, 2009.

[76] G. Kourakos and A. Mantoglou. Development of a multi-objective optimization algorithm using surrogate models for coastal aquifer management. *Journal of Hydrology*, 479:13–23, 2013.

[77] F. Kursawe. A variant of evolution strategies for vector optimization. In *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature-PPSN I*, pages 193–197. Springer-Verlag, 1991.

[78] V. Lattarulo, P. Seshadri, and G. Parks. Optimization of a supersonic airfoil using the multi-objective alliance algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1333–1340. ACM, 2013.

[79] D. Lee, L. Gonzalez, J. Periaux, and K. Srinivas. Robust design optimisation using multi-objective evolutionary algorithms. *Computers & Fluids*, 37:565–583, 2008.

[80] S. Lee, P. Almon, W. Fink, A. Petropoulos, and R. Terrile. Comparison of multi-objective genetic algorithms in optimizing q-law low-thrust orbit transfers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 25–29. ACM, 2005.

[81] G. Li, M. Li, S. Azarm, S. Hashimi, T. Ameri, and N. Qasas. Improving multi-objective genetic algorithm with adaptive design of expriments and online metamdeling. *Structural and Multidisciplinary Optimization*, 37:447–461, 2009.

[82] H. Li and Q. Zhang. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 12:284–302, 2009.

[83] M. Li, G. Li, and S. Azarm. A kriging metamodel assisted multi-objective genetic algorithm for design optimization. *Journal of Mechanical Design*, 130:1–10, 2008.

[84] D. Lim and Y. Jin. Generalizing surrogate-assisted evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 14:329–354, 2010.

[85] G. Liu, X. Han, and C. Jiang. A novel multi-objective optimization method based on an approximation model management technique. *Computer Methods in Applied Mechanics and Engineering*, 197:2719–2731, 2008.

[86] Y. Liu and M. Collette. Improving surrogate-assisted variable fidelity multi-objective optimization using a clustering algorithm. *Applied Soft Computing*, 24:482–493, 2014.

[87] I. Loshchilov, M. Schoenauer, and M. Sebag. A mono surrogate for multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 471–478. ACM, 2009.

[88] I. Loshchilov, M. Schoenauer, and M. Sebag. Dominance-based Pareto-surrogate for multi-objective optimization. In K. Deb, A. Bhattacharya, N. Chakroborty, S. Das, J. Dutta, S. Gupta, A. Jain, V. Aggarwal, J. Branke, S. Louis, and K. Tan, editors, *Proceedings of the Simulated Evolution and Learning*, pages 230–239. Springer, 2010.

[89] C. Luo, K. Shimoyama, and S. Obayashi. Kriging model based many-objective optimization with efficient calculation of expected hypervolume improvement. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1187–1194. IEEE, 2014.

[90] S. Martinez and C. Coello. MOEA/D assisted by RBF networks for expensive multi-objective optimization problems. In C. Blum, editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1405–1412. ACM, NY, 2013.

[91] M. Mckay, R. Beckman, and W. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42:55–61, 2000.

[92] T. Mengistu and W. Ghaly. Aerodynamic optimization of turbomachinery blades using evolutionary methods and ANN-based surrogate models. *Optimization and Engineering*, 9:239–255, 2008.

[93] K. Miettinen. *Nonlinear multiobjective optimization.* Kluwer, Boston, MA, 1999.

[94] K. Mitra and S. Majumder. Successive approximate model based multi-objective optimization for an industrial straight grate iron ore induration process using evolutionary algorithm. *Chemical Engineering Science*, 66:3471–3481, 2011.

[95] M. Mlakar, D. Petelin, T. Tusar, and B. Filipic. GP-DEMO: differential evolution with multiobjective optimization based on gaussian process models. *European Journal of Operational Research*, 243:347–361, 2015.

[96] A. Mogilicharla, T. Chugh, S. Majumder, and K. Mitra. Multi-objective optimization of bulk vinyl acetate polymerization with branching. *Materials and Manufacturing Processes*, 29:210–217, 2014.

[97] P. Nain and K. Deb. A multi-objective optimization procedure with successive approximate models. Technical Report 2005002, KanGAL, Indian Institute of Technology Kanpur, India, 2005.

[98] H. Nakayama, K. Inoue, and Y. Yoshimori. Approximate optimization using computaional intelligence and its application to reinforcement of cable-stayed bridges. In *Proceedings of the Integrated Intelligent Systems for Engineering Design*, pages 289–304. IOS press, 2006.

[99] A. Osyczka and S. Kundu. A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural optimization*, 10(2):94–99, 1995.

[100] A. Oyama, Y. Okabe, K. Shimoyama, and K. Fujii. Aerodynamic multiobjective design exploration of a flapping airfoil using a navier-stokes solver. *Journal of Aerospace Computing, Information, and Communication*, 6:256–270, 2009.

[101] P. Palar, T. Tsuchiya, and G. Parks. A comparative study of local search within a surrogate-assisted multi-objective memetic algorithm framework for expensive problems. *Applied Soft Computing*, 43:1–19, 2016.

[102] P. S. Palar, T. Tsuchiya, and G. Parks. Comparison of scalarization functions within a local surrogate assisted multi-objective memetic algorithm framework for expensive problems. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 862–869, 2015.

[103] L. Pavelski, M. Delgado, C. Almeida, R. Gonaslves, and S. Venske. Extreme learning surrogate models in multi-objective optimization based on decomposition. *Neurocomputing*, 180:55–67, 2016.

[104] L. Pavelski, M. Delgado, C. Almeida, R. Goncalves, and S. Venske. ELMOEA/D-DE: Extreme learning surrogate models in multi-objective optimization based on decomposition and differential evolution. In *Proceedings of the Brazilian Conference on Intelligent Systems*, pages 318–323. IEEE, 2014.

[105] M. Pilát and R. Neruda. ASM-MOMA: Multiobjective memetic algorithm with aggregate surrogate model. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1202–1208. IEEE, 2011.

[106] M. Pilát and R. Neruda. Improving many-objective optimizers with aggreagte meta-models. In *Proceedings of the 11th International Conference on Hybrid Intelligent Systems*, pages 555–560. IEEE, 2011.

[107] M. Pilát and R. Neruda. Hypervolume-based local seacrh in multi-objective evolutionary optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 637–644. ACM, 2014.

[108] C. Poloni, A. Giurgevich, L. Onesti, and V. Pediroda. Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 186(2):403 – 420, 2000.

[109] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisted S-metric selection. In *Proceedings of the Parallel Problem Solving from Nature-PPSN X*, pages 784–794. Springer, 2008.

[110] W. Ponweiser, T. Wagner, and M. Vincze. Clustered multiple generalized expected improvement: A novel infill sampling criterion for surrogate models. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3515–3522. IEEE, 2008.

[111] S. Qasem, S. Shamsuddin, S. Hashim, M. Darus, and E.A.-Shammari. Memetic multiobjective particle swarm optimization-based radial basis function network for classification problems. *Information Sciences*, 239:165 – 190, 2013.

[112] D. Quagliarella and A. Vicini. *Genetic Algorithms and Evolutionary Strategies in Engineering and Computer Science*, chapter Coupling genetic algorithms and gradient based optimization techniques, pages 289–309. Chichester, Wiley, UK, 1998.

[113] T. Ray, H. Singh, A. Isaacs, and W. Smith. Infeasibility driven evolutionary algorithm for constrained optimization. In E. Mezura-Montes, editor, *Constraint-Handling in Evolutionary Optimization*, pages 145–165. Springer, 2009.

[114] R. Regis. Multi-objective constrained black-box optimization using radial basis function surrogates. *Journal of Computational Science*, 16:140–155, 2016.

[115] M. Reyes-Sierra and C. Coello. A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization. In *Proceedings of the EEE Congress on Evolutionary Computation*, pages 65–72. IEEE, 2005.

[116] T. Robic and B. Filipic. DEMO: Differential evolution for multiobjective optimization. In *Proceedings of Evolutionary Multi-criterion Optimization*, pages 520–533. Springer, 2005.

[117] P. Roy and K.Deb. High dimensional model representation for solving expensive multi-objective optimization problems. Technical Report COIN Report Number 2016012, Michigan State University, 2016.

[118] J. Sacks, W. Welch, T. Mitchell, and H. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4:409–423, 1989.

[119] L. Santana-Quintero, A. M. no, and C. Coello. A review of techniques for handling expensive functions in evolutionary multi-objective optimization. In Y. Tenne and C.-K. Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, pages 29–59. Springer, 2010.

[120] M. Sasena, P. Papalambros, and P. Goovaerts. Exploration of metamodeling sampling criteria for constrained global optimization. *Engineering Optimization*, 34:263–278, 2002.

[121] K. Sastry, D. E. Goldberg, and M. Pelikan. Don't evaluate, inherit. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 551–558. Morgan Kaufmann, 2001.

[122] J. Schaffer. *Some experiments in machine learning using vector evaluated genetic algorithms.* PhD thesis, Vanderbilt Univ.,Nashville, TN, Jan 1985.

[123] C.-W. Seah, Y.-S. Ong, I. W. Tsang, and S. Jiang. Pareto rank learning in multi-objective evolutionary algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012.

[124] K. Shimoyama, K. Sato, S. Jeong, and S. Obayashi. Updating kriging surrogate models based on the hypervolume indicator in multi-objectie optimization. *Journal of Mechanical Design*, 135:1–7, 2013.

[125] K. Sindhya, K. Deb, and K. Miettinen. Improving convergence of evolutionary multi-objective optimization with local search: a concurrent-hybrid algorithm. *Natural Computing*, 10:1407–1430, 2011.

[126] K. Sindhya, K. Miettinen, and K. Deb. A hybrid framework for evolutionary multi-objective optimization. *IEEE Transactions on Evolutionary Computation,*, 17:495–511, 2013.

[127] H. Singh, T. Ray, and W. Smith. Surrogate assisted simulated annealing (SASA) for constrained multi-objective optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

[128] P. Singh, I. Couckuyt, F. Ferranti, and T. Dhaene. A constrained multi-objective surrogate-based optimization algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3080–3087. IEEE, 2014.

[129] R. Smith, B. Dike, and S. Stegmann. Fitness inheritance in genetic algorithms. In *Proceedings of the ACM Symposium on Applied Computing*, pages 345–350. ACM, 1995.

[130] I. Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and Computers in Simulation*, 55:271–280, 2001.

[131] A. Syberfeldt, H. Grimm, A. Ng, and R. John. A parallel surrogate-assisted multi-objective evolutionary algorithm for computationally expensive optimization problems. In *Proceedings of the IEEE World Congress on Computational Intelligence*, pages 3177–3184. IEEE, 2008.

[132] Y. Tenne and S. Armfield. Metamodel accuracy assessment in evolutionary optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1505–1512. IEEE, 2008.

[133] G. Toscano and K. Deb. Study of the approximation of the fitness landscape and the ranking process of scalarizing functions for many-objective problems. Technical Report COIN Report Number 2016018, Michigan State University, 2016.

[134] A. Turco. Metahybrid: Combining metamodels and gradient-based techniques in a hybrid multi-objective genetic algorithm. In C. Coello, editor, *Proceedings of the Learning and Intelligent Optimization*, pages 293–307. Springer, 2011.

[135] H. Ulmer, F. Streichert, and A. Zell. Evolution strategies assisted by gaussian processes with improved preselection criterion. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 692–699. IEEE, 2003.

[136] D. A. van Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm test suites. In *Proceedings of the 1999 ACM Symposium on Applied Computing*, pages 351–357. ACM, 1999.

[137] D. Veldhuizen. *Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations*. PhD thesis, Graduate School of Engineering of the Air Force Institute of Technology Air University, Dayton, 1999.

[138] D. Veldhuizen and G. Lamont. Evolutionary computation and convergence to a Pareto front. In *Proceedings of the Genetic Programming*, pages 221–228. Morgan Kaufmann, 1998.

[139] J. Vrgut and B. Robinson. Improved evolutionary optimization from genetically adaptive multimethod search. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 104, pages 708–711, 2007.

[140] T. Wagner, M. Emmerich, A. Deutz, and W. Ponweiser. On expected-improvement criteria for model-based multi-objective optimization. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *Proceedings of the Parallel Problem Solving from Nature-PPSN XI*, pages 718–727. Springer, 2010.

[141] G. Wang. Adaptive response surface method using inherited latin hypercube design points. *Journal of Mechanical Design*, 125:210–220, 2003.

[142] H. Wang, Y. Jin, and J. Jansen. Data-driven surrogate-assisted multi-objective evolutionary optimization of a trauma system. *IEEE Transactions on Evolutionary Computation*, 20:939–952, 2016.

[143] B. Wilson, D. Cappelleri, T. W. Simpson, and M. Frecker. Efficient pareto frontier exploration using surrogate approximations. *Optimization and Engineering*, 2(1):31–50, 2001.

[144] B. Yang, Y.-S. Yeun, and W.-S. Managing approximation models in multiobjective optimization. *Structural and Multidisciplinary Optimization*, 24:141–156, 2002.

[145] R. Yuan and B. Guangchen. Comparison of neural network and kriging method for creating simulation-optimization metamodels. In B. Yang, W. Zhu, Y. Dai, L. T. Yang, and J. Ma, editors, *Proceedings of the 8th IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 815–821. IEEE, 2009.

[146] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11:712–731, 2007.

[147] Q. Zhang, W. Liu, and H. Li. The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 203–208. IEEE, 2009.

[148] Q. Zhang, W. Liu, E. Tsang, and B. Virginas. Expensive multiobjective optimization by MOEA/D with gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14:456–474, 2010.

[149] Q. Zhang, A. Zhau, S. Zhao, P. Suganthan, W. Liu, and S. Tiwari. Multiobjective optimization test instances for the CEC 2009 special session and competition. Technical Report CES-487, University of Essex/ Nanyang Technological University, Essex, U.K./Singapore, 2009.

[150] Y. Zheng, B. Julstrom, and W. Cheng. Design of vector quantization codebooks using a genetic algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 525–529. IEEE, 1997.

[151] J. Zhu, Y.-J. Wang, and M. Collette. A multi-objective variable-fidelity optimization method for genetic algorithms. *Engineering Optimization*, 46:521–542, 2013.

[152] E. Zitzler. *Evolutionary algorithms for multiobjective optimization: methods and applications*. PhD thesis, Swiss Federal Institute of Technology Zurich, 1999.

[153] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8:173–195, 2000.

[154] E. Zitzler and S. Kunzli. Indicator-based selection in multiobjective search. In X. Y. et al., editor, *Proceedings of the Parallel Problem Solving from Nature-PPSN VIII*, pages 832–842. Springer, 2004.

[155] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In K. Giannakoglou, editor, *Proceedings of the Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pages 95–100. CIMNE, 2002.

[156] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms-a comparative case study. In A. Eiben, editor, *Proceedings of the Parallel Problem Solving from Nature-PPSN V*, pages 292–301. Springer, 1998.

[157] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 8:117–132, 2003.