METAMODELING FRAMEWORK FOR SIMULTANEOUS MULTI-OBJECTIVE OPTIMIZATION USING EFFICIENT EVOLUTIONARY ALGORITHMS

By

Proteek Chandan Roy

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science - Doctor of Philosophy

2019

ABSTRACT

METAMODELING FRAMEWORK FOR SIMULTANEOUS MULTI-OBJECTIVE OPTIMIZATION USING EFFICIENT EVOLUTIONARY ALGORITHMS

By

Proteek Chandan Roy

Most real-world problems are comprised of multiple conflicting objectives and solutions to those problems are multiple Pareto-optimal trade-off solutions. The main challenge of these practical problems is that the objectives and constraints do not have any closed functional forms and they are expensive for computation as well. Objectives coming from finite element analysis, computational fluid dynamics software, network flow simulators, crop modeling, weather modeling or any other simulations which involve partial differential equations are good examples of expensive problems. These problems can also be regarded as "low-budget" problems since only a few solution evaluations can be performed given limited time. Nevertheless, parameter estimation and optimization of objectives related to these simulations require a good number of solution evaluations to come up with better parameters or a reasonably good trade-off front. To provide an efficient search process within a limited number of exact evaluations, metamodel-assisted algorithms have been proposed in the literature. These algorithms attempt to construct a computationally inexpensive representative model of the problem, having the same global optima and thereby providing a way to carry out the optimization in metamodel space in an efficient way. Population-based methods like evolutionary algorithms have become standard for solving multi-objective problems and recently Metamodel-based evolutionary algorithms are being used for solving expensive problems. In this thesis, we would like to address a few challenges of metamodel-based optimization algorithms and propose some efficient and innovative ways to construct these algorithms. To

approach efficient design of metamodel-based optimization algorithm, one needs to address the choice of metamodeling functions. The most trivial way is to build metamodels for each objective and constraint separately. But we can reduce the number of metamodel constructions by using some aggregated functions and target either single or multiple optima in each step. We propose a taxonomy of possible metamodel-based algorithmic frameworks which not only includes most algorithms from the literature but also suggests some new ones. We improve each of the frameworks by introducing trust region concepts in the multi-objective scenario and present two strategies for building trust regions. Apart from addressing the main bottleneck of the limited number of solution evaluations, we also propose efficient non-dominated sorting methods that further reduce computational time for a basic step of multi-objective optimization. We have carried out extensive experiments over all representative metamodeling frameworks and shown that each of them can solve a good number of test problems. We have not tried to tune the algorithmic parameters yet and it remains as our future work. Our theoretical analyses and extensive experiments suggest that we can achieve efficient metamodel-based multi-objective optimization algorithms for solving test as well as real-world expensive and low-budget problems.

I would like to dedicate this thesis to my parents Dr. Provash Chandra Roy and Lily Roy for supporting me endlessly towards achieving this goal.

ACKNOWLEDGMENTS

I would like to thank to my advisor Dr. Kalyanmoy Deb who has supported me throughout my PhD career with research grants and enriched my knowledge with innovative ideas. I am grateful to him for the knowledge that I have to write this thesis. I also remember all of my colleagues, current and past lab members: Dr. Haitham Seada, Dr. Abhinav Gaur, Julian Blank, Yeashesh Dhebar, Abhiroop Ghosh, Zhichao Lu, Khaled Talukder, Rayan Hussein, Anirudh Suresh, Yash Vesikar, Dr. Mohamed Abouhawwash, Dr. Ali Ahrari, Dr. Ling Zhu. I would like to thank them for being a part of this journey by participating in the discussions, writing papers and making all the fun activities in the lab.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

Chapter 1

Introduction and Motivation

In order to solve a practical optimization problem, the problem must first be implemented (coded or expressed symbolically) within the optimizer (either a computer code or a commercial software). Often, this implementation process involves linking the optimizer to a third-party evaluation software such as a finite element or a computational fluid dynamics software or a network flow simulator. Most practical optimization problems face a common difficulty: the objective and constraint functions are computationally expensive to evaluate. No matter how efficient and intelligent an optimization algorithm is, every method must evaluate a requisite number of solutions from the search space before arriving at a reasonably good solution. While this can be a time-consuming process, in most occasions, practitioners cannot wait too long to find such a solution. Although the advent and advances of parallel and distributed computing certainly help in reducing the overall computational time, algorithmic efficacy is also extremely important.

Evolutionary algorithms have achieved state-of-the-art results in complicated multi-objective problems due to its robustness nature in search process [2]. Often times, an evolutionary algorithm requires thousands of function evaluations to reach near-optimal solutions. This is very inefficient in a sense that most practical problems require expensive simulation of systems to evaluate objective functions and constraints. Therefore, Metamodel-based evolutionary algorithms are becoming popular choice for simulation optimization. Approximation of objective functions and constraints using metamodels facilitate us to carry out the opti-

mization in the low-fidelity model space efficiently where high-fidelity solution evaluations are not needed after we build the models.

Up until this thesis, no comprehensive taxonomy or classification about the algorithmic view of Metamodel-based optimization has been proposed. Since the exact function definition is not known for objectives and constraints in most cases, accuracy of the models build upon the exactly evaluated solutions is very important. In contrast to traditional approach that keeps the same number of models as the number of objectives and constraints, we can make either more or less number of metamodels while maintaining the same global optima. This would provide more control over local versus global search and bring robustness in the process leading better solutions within limited function evaluations.

Towards efficient search and optimization, one important aspect is to restrict the search space with intelligent guess of optima. We have introduced trust region concept in multi-objective optimization in order to deal with uncertainties of the models. Another aspect of efficient search is to reduce the time complexity of some basic optimization steps. In multi-objective evolutionary methods, non-dominated sorting is an essential step. Mathematically speaking, a solution is said to be better than another solution in a multi-objective scenario if it is better in at least one objective and equal or better in others provided that both are feasible solutions. For a sampling or population based optimization algorithm, it is a fundamental task to rank the solutions in such a way that non-dominated solutions are preferred and dominated solutions less preferred. In this thesis, we have improved this basic step to facilitate optimization.

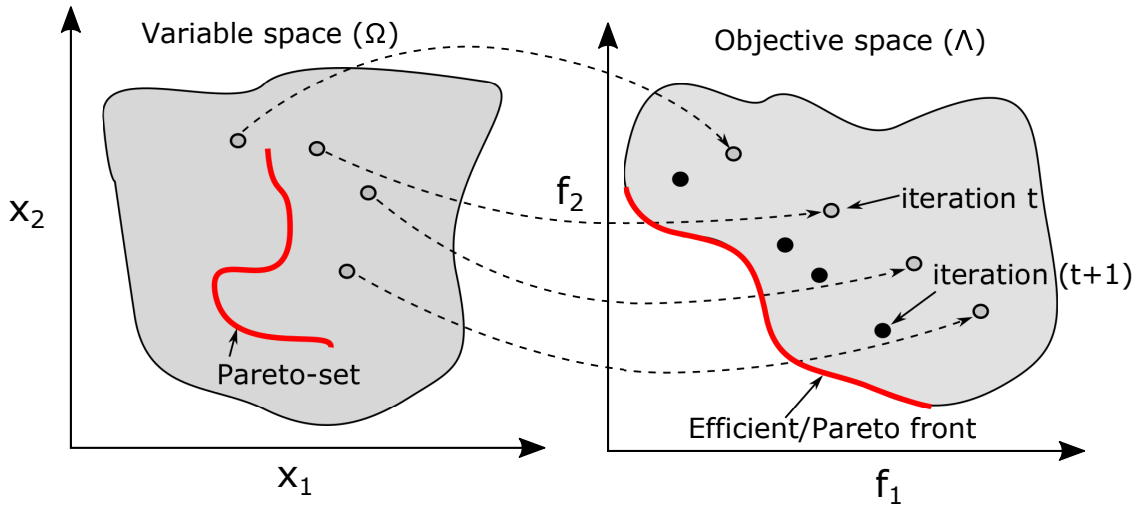

Figure 1.1: Mapping between variable and objective space is presented in a two-objective (f_1 and f_2) two-variable (x_1 and x_2) problem. Pareto-front (red in objective space) and Pareto-set (red in variable space) are optimal solutions in objective and variable space respectively. New solutions are created using genetic operators in the successive steps of an evolutionary algorithm (black-filled circle).

1.1 Problem Definition

We now formally and mathematically define a multi-objective optimization problem which is the premise of this thesis.

1.1.1 Multi-Objective Optimization Problem

A multi-objective optimization problem can be formulated as follows.

$$\begin{aligned} &\text{Minimize } \mathbf{F(x)} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})) \\ &\text{subject to, } g_j(\mathbf{x}) \geq 0, \quad \forall j \in \{1, \ldots, J\} \\ &\mathbf{x} \in \Omega \subseteq \mathbb{R}^n \text{ and, } \mathbf{F} \in \Lambda \subseteq \mathbb{R}^m \end{aligned} \tag{1.1}$$

Here each of the functions f_i has either a functional form or it can come from expensive simulations. Variable space Ω is our search space and objective space Λ is a mapping from

$\mathbf{x}$ to $\mathbf{F}$. Without loss of generality, feasibility is defined using $\geq$ relation. Pareto-dominance relation for this optimization problem can be defined as follows.

Definition 1 (Pareto-dominance)**.** *A solution* $\mathbf{x} \in \mathcal{S} \subseteq \Omega$ *dominates another point* $\mathbf{y} \in \mathcal{S}$ *with Pareto-dominance relation if* $f_i(\boldsymbol{x}) \leq f_i(\mathbf{y}) \quad \forall i \in \{1, \ldots, m\}$ *and* $f_j(\boldsymbol{x}) < f_j(\mathbf{y}) \; \exists j \in \{1, \ldots, m\}$*. We denote this as* $\boldsymbol{x} \succ \boldsymbol{y}$*.*

1.1.2 Metamodel Assisted Multi-Objective Optimization

Many practical optimization problem are confronted with the difficulty that objective functions and constraints are computationally expensive to evaluate. Objective and constraint values often come from an expensive simulation of a system. Because of that, most of the time, researchers are bound to have a limited number of solution evaluations to get near-optimum feasible solutions. Researchers have used metamodels or surrogate models to make a low-fidelity approximation of the objective functions and constraints. A multi-objective optimization problem is called expensive if time complexity of some of the objectives and/or constraints is high compare to basic operations of optimization algorithm itself. As a basic framework, we carry out the optimization in the model space created by the metamodels of the objectives and constraints. We define a basic low-fidelity optimization problem as follows.

$$\begin{aligned} &\text{Minimize } \widehat{\mathbf{F}}(\mathbf{x}) = \left(\widehat{f}_1(\mathbf{x}), \widehat{f}_2(\mathbf{x}), \ldots, \widehat{f}_m(\mathbf{x})\right) \\ &\text{subject to } \widehat{g}_j(\mathbf{x}) \geq 0, \quad \forall j \in \{1, \ldots, J\} \\ &\mathbf{x} \in \widehat{\Omega} \subseteq \mathbb{R}^n \text{ and, } \widehat{\mathbf{F}} \in \widehat{\Lambda} \subseteq \mathbb{R}^m \end{aligned} \tag{1.2}$$

Here the hat sign indicates the cheap approximation model of an exact expensive function.

1.2 Evolutionary Multi-objective Optimization

Evolutionary algorithms (EA) have become very popular and widely used for solving multi-objective problem because of their robustness and flexibility [3, 4]. Most of the classical algorithms (except derivative-free algorithms) need gradient of the objectives and constraints and can mostly solve one single-objective problem at a time. Therefore, a good number of function evaluations are needed to get a good representation of local Pareto-front of a multi-objective problem. In contrast to classical optimization algorithms which, in general, get stuck in local optima, evolutionary multi-objective optimization (EMO) algorithms use their controlled randomness and try to escape them. These algorithms do not assume any convexity or differentiability of the objective functions and constraints. EMO methods start with a population of initial solutions on which genetic operators (e.g., reproduction, crossover, and mutation) are applied. These operators make small (mutation) to large (crossover) change on the solutions and best solutions are then kept for the next generation. After a certain number of generations, the population converge to near-optimal region. Due to the complicated process of genetic operators, the convergence to true optimum is not guaranteed in general. In practice, evolutionary algorithm along classical method outperforms both of them separately applied.

In practice, the user is also satisfied with a reasonable solution rather than true optimum due to the limited budget of function evaluations. The target of EMO algorithms are two fold: a) A well-converged set of trade-off solutions, and, b) A well-diversified set of solutions across the entire Pareto-front [3]. One of the major limitation of EMO algorithms is that, a good number of solution evaluations is needed [5] for evolution to take effect. This issue becomes more prominent when the problem to be solved involves computationally expensive

functions which is another challenge in solving industrial optimization problems. To obtain solutions for expensive problems in a limited number of expensive function evaluations, surrogates (or metamodels) have been used in the literature as an alternative to expensive evaluations.

Before we go to the next subsection, we want to clarify a few notations used over and over again in this thesis. The words 'model', 'metamodel' and 'metademodeling methods' are used as synonyms. These are the techniques that take variables as input and predict objectives and constraints as output. On the other hand, metamodeling framework is the algorithmic aspect of the optimization algorithm which does not depend on which metamodels are used. The infill sampling criteria better known as acquisition function in other literature is the function that is used to distinguish solutions in low-fidelity space. We have used "selection function" instead which better suited for multi-objective scenario. We will define selection functions for different algorithms in the next chapter.

1.3 Summary of Research Contributions

The main research contributions of this study is summarized in the following subsections.

1.3.1 Thesis Statement

This thesis develops new frameworks and strategies to solve computationally expensive optimization problems. This work presents simultaneous optimization of multiple objectives using evolutionary algorithms and it introduces new trust region concepts. Additionally, it develops an adaptive algorithm using an ensemble of frameworks and strategies to solve expensive problems without increasing the total number of function evaluations.

1.3.2 A Taxonomy for Metamodel-based Multi-Objective Optimization

An increased interest in metamodeling efforts has grown from recent developments in optimization methods. Some researchers have made efforts to classify different metamodeling approaches, but only in the realm of single-objective optimization. Most metamodeling efforts in multi-objective optimization, so far, seem to have taken a straightforward extension of single-objective metamodeling approaches. First, every objective and constraint function is modeled independently. Thereafter, a standard EMO methodology is applied to the metamodels, instead of the original objective and constraint functions, to find the non-dominated front. In some studies, the above metamodeling-EMO combination is repeated progressively so that the refinement of the metamodels can occur with iterations. However, with the possibilities of a combined constraint violation function that can be formulated by combining violations of all constraints in a normalized manner [6] and a combined scalaraized objective function (weighted-sum, achievement scalarization function or Tchebyshev function) [7], different metamodeling frameworks can certainly be explored. While the straightforward approach requires the construction of many metamodels, the suggested metamodels for combined objective and constraint violations will reduce the number of needed metamodels. However, the flip side is that each metamodel of the combined functions is likely to be more complex, having discontinuous, non-differentiable, and multi-modal landscapes. Thus, the success of these advanced metamodeling frameworks is closely tied with the advancements in the metamodeling methods. While these advancements are in progress, in this report, we outline, for the first time, a number of different and interesting metamodeling frameworks [8] for multi-objective optimization, utilizing combined approaches of objectives alone, con-

straints alone, as well as objectives and constraints together. Our taxonomy includes one framework that requires $(M + J)$ metamodels (where M and J are the number of objectives and constraints, respectively) to another framework that requires only one metamodel.

1.3.3 Metamodeling Frameworks for Simultaneous Optimization

Despite significant progress in the use of metamodels for single-objective optimization, metamodeling methods have received lukewarm attention for multi-objective optimization. A recent study [9] at Computational Optimization and Innovation (COIN) Laboratory at Michigan State University classified various metamodeling approaches, of which one particular method is interesting, challenging, and novel. In this approach, a selection operator's assignment function, as it is implemented in an evolutionary multi-objective optimization (EMO) algorithm, is directly modeled. Thus, this methodology requires only one or few selection functions to be modeled irrespective of a multitude of objective and constraint functions in a problem. However, the flip side of the framework is that the resulting function is multi-modal having a different optimum for every desired Pareto-optimal solution. We have used two different selection functions based on two recent ideas: (i) KKT proximity measure function and (ii) multi-modal based evolutionary multi-objective (MEMO) selection function. The resulting metamodeling methods are applied to several standard two and three-objective constraint and unconstrained test problems. Near Pareto-optimal solutions are found using only a fraction of high-fidelity solution evaluations compared to usual EMO applications. This approach also reduces the number of metamodel constructions.

1.3.4 Efficient Search Strategy

We have developed an efficient search strategy by restricting the search regions to the promising ones and by improving the running time of evolutionary multi-objective optimization algorithms. Due to the exponential increase of search space and limited budget of function evaluation, it is impractical to search every region of the space. Since the number of function evaluations is limited, we should rather focus only on the promising regions. In single objective optimization, the idea of focusing on a particular region is called trust regions. Inside a trust region, we assume to have an accurate model of the original space. We grow or shrink that space based on the performance of the algorithm. In this thesis, we extend classical trust region based unconstrained single objective algorithm into a population-based constrained multi-objective optimization algorithm. With the increase of the number of variables and objectives, EMO algorithms, in general, need more solutions to evolve and a good number of generations to converge. When the number of solutions is increased, we need a faster procedure to rank them using non-dominated sorting. In this thesis we develop an efficient non-dominated sorting method which reduces the running time of the algorithm.

1.3.5 Ensemble of Metamodeling Frameworks

Here, our main contribution is twofold. As we mentioned previously regarding the proposed taxonomy of different metamodeling frameworks for multi-objective optimization, there are several ways to build and utilize metamodeling approaches. We argue that it is more efficient to use different metamodeling frameworks at different stages of the optimization process and then propose several switching strategies between the metamodeling frameworks. A switching between metamodeling frameworks, compare to multiple frameworks one at a

time, is an efficient approach since it doesn't increase the number of high-fidelity solution evaluations. On several multi-objective constrained and unconstrained test problems, the switching methods have produced better results by using a low budget of solution evaluations, compared to the individual metamodeling framework alone.

1.4 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we present a literature survey of Metamodel-based optimization methods and a proposed taxonomy over different approaches. In Chapter 3, we present simultaneous metamodeling frameworks for solving computationally expensive problems. Chapter 4 presents efficient search techniques using trust region concept and fast non-dominated sorting procedure. In Chapter 5, an ensemble based algorithm is proposed for solving expensive multi-objective problems. In Chapter 6, results and comparisons among various metamodeling frameworks are discussed. Chapter 7 provides conclusions of the studies performed and discusses the future work.

Chapter 2

A Taxonomy for Metamodel-based Multi-Objective Optimization

In this chapter we describe some of the previous works on Metamodel-based optimization and propose a taxonomy over different approaches to solve expensive multi-objective optimization problems.

2.1 Existing Work in Metamodel-based Optimization

Direct fitness replacement (DFR) [10] has been one of the most straightforward methods to embed surrogate models into MOEAs. DFR assumes that solutions assessed in the surrogate models are comparable to those assessed by the real function (high fidelity function evaluations). DFR is further subdivided into three major model managements [10]: (1) No Evolution control (NEC), which evaluates the MOEA's generated solutions in the surrogate model exclusively (this model trains the surrogate model before the execution of the MOEA), (2) Fixed evolution control (FEC), which only some generations or some individuals are evaluated in the surrogate model while the remaining population is evaluated using the real test function, and (3) Adaptive evolution control (AEC), which avoids any possible poor parameter tuning by the use of an adaptive control that adjusts the number of solutions that will be evaluated in the surrogate model. The recent developments of optimization methods have

led to an increasing interest of approximation models or *surrogate* models [11, 12]. The use of metamodels (or surrogate models) to approximate the functional form of exact objectives and constraints by using a few high-fidelity solution evaluations is a common approach [13]. Among various methods, the Kriging method is one of the widely used metamodels, which can provide an estimated function value and also simultaneously provide an error estimate of the approximation [14].

Emmerich et al. [15] have generalized the probability of improvement and the expected improvement concept to multi-objective optimization. In [16, 17], researchers have used scalarization methods to convert multi-objective optimization in multiple single-objective optimization problems. Several efficient metamodeling frameworks have been proposed recently for multi-objective optimization [18, 19, 20, 21, 22]. These frameworks use different metamodeling methods to approximate objective and constraint functions, such as radial basis functions, Kriging, Bayesian neural network, support vector regression, and others [23]. Zhang et al. [24] proposed an MOEA/D-EGO algorithm that models each objective function independently. They constructed multiple expected global optimization (EGO) functions for multiple reference lines of the MOEA/D approach to find pre-specified number of trade-off solutions in each optimization task. No constraint handling procedure was suggested. Thus, this method falls under our M1-2 framework. Chugh et al. [19] proposed a surrogate-assisted adaptive reference vectors guided evolutionary algorithm (K-RVEA) for computationally expensive optimization problems with more than three objectives. Since all objectives and constraints are modeled separately, this method also falls under our M1-2 framework. Pan et al. [25] proposed a classification based surrogate-assisted evolutionary algorithm (CSEA) for solving unconstrained optimization problems by using an artificial neural network (ANN) as a surrogate model. The surrogate model aims to learn the dominance relationship between

the candidate solutions and a set of selected reference solutions. This algorithm falls in our M3-2 framework.

Ensemble methods have been used in surrogate-assisted optimization for solving expensive problems [26, 27, 28, 29, 30], but in most of these methods, an ensemble of different metamodeling methods, such as RBF, Kriging, response surfaces, are considered to choose a single suitable method. No effort is made to consider an ensemble of metamodeling frameworks for combining multiple objectives and constraints differently and choosing the most suitable one for optimization. In this paper, we use an ensemble of 10 metamodeling frameworks described in the next section and propose an adaptive selection scheme of choosing one thereafter.

In literature, researchers have used various machine learning models such as support vector regression, neural network, RBF, response surface method etc as a surrogate model. Kriging, or the Gaussian process regression, has been one of the most popular choices in surrogate techniques used mainly because of its ability to provide uncertainty information of the approximated values. The term Kriging was proposed by Matheron in 1963 [31] in honor of the South African mining engineer Danie G. Krige [32]. His research was focused on the distribution of gold samples found in mines and the correlation between these samples. He implemented a statistical technique based on a limited amount of samples, which is now known as Kriging. The first work of Kriging as an approximation of simulation-based computer experiments was proposed in 1989 by Sacks et al [33]. However, the most cited algorithm in using Kriging is efficient global optimization (EGO) proposed in 1998 by Jones et al. [34] for single-objective optimization problems. EGO uses a criterion called expected improvement (EI) to select samples for training the Kriging model.

2.2 Scalarization Methods

One of the common ways to solve the generic multi-objective optimization problem is to solve a parameterized *achievement scalarization function (ASF)* optimization problem repeatedly for different parameter values. The ASF approach was originally suggested by Wierzbicki [35]. For a specified reference point $\mathbf{z}$ and a weight vector $\mathbf{w}$ (parameters of the ASF problem), the ASF problem is given as follows:

$$\begin{aligned} \text{Minimize}_{(\mathbf{x})} \quad & \text{ASF}(\mathbf{x}, \mathbf{z}, \mathbf{w}) = \max_{i=1}^{M} \left(\frac{f_i(\mathbf{x}) - z_i}{w_i} \right), \\ \text{Subject to} \quad & g_j(\mathbf{x}) \leq 0, \quad j = 1, 2, \ldots, J. \end{aligned} \tag{2.1}$$

The reference point $\mathbf{z} \in R^M$ is any point in the M-dimensional objective space and the weight vector $\mathbf{w} \in R^M$ is an M-dimensional unit vector for which every $w_i \geq 0$ and $\|\mathbf{w}\| = 1$. To avoid division by zero, we shall consider strictly positive weight values. It has been proven that for above conditions of $\mathbf{z}$ and $\mathbf{w}$, the solution to the above problem is always a local Pareto-optimal solution [7]. The first figure of Figure 2.1 illustrates the ASF procedure of arriving at a weak or a strict Pareto-optimal solution.

2.3 A Taxonomy For Metamodeling Frameworks

We propose a taxonomy of various methods for using metamodeling approach in multiple and many-objective optimization algorithms. Our taxonomy finds six different broad methodologies (M1 to M6), as illustrated in Figure 2.2. Our approach is based on the cardinality of metamodels for objectives and constraints. In the first method (M1), all objectives and constraints are modeled independently, thereby requiring a total of $(M + J)$ metamodels

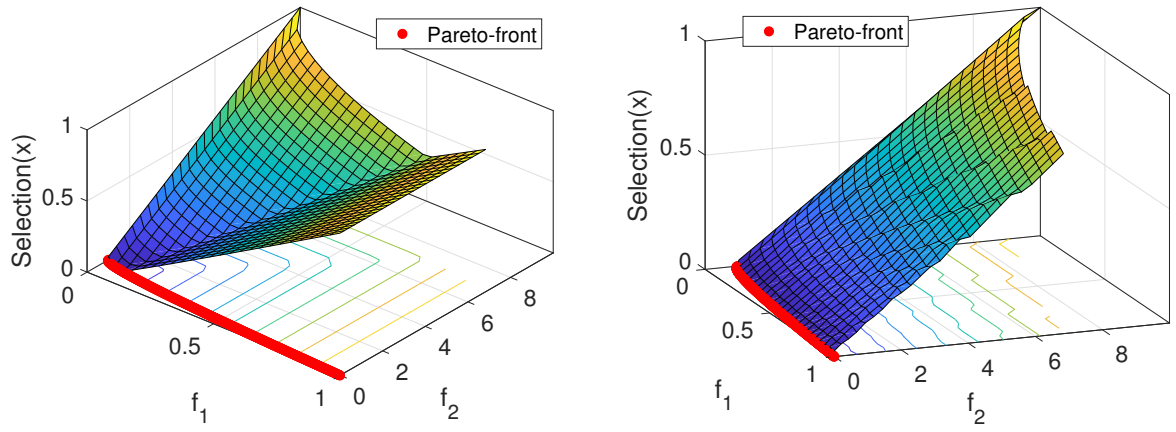

Figure 2.1: Selection function and contour plots of unimodal and multi-modal procedure for finding Pareto-optimal solutions of test function ZDT1 [1] where only one and multiple solutions are targeted respectively.

before a multi-objective optimization approach can be applied. This method is a straightforward extension of the single-objective metamodeling approach. Once all such metamodels are constructed, an EMO algorithm can use them to a find one Pareto-optimal solution at a time (like the generative method used in classical optimization literature [7]) and we call this method M1-1, or they can be used to find several Pareto-optimal solutions simultaneously (similar to evolutionary multi-objective optimization) and we call this method M1-2.

The next metamodeling methodology can approximate an overall estimation function of all constraint violations together as one quantity, thereby reducing the overall number of metamodels to $(M+1)$. The well-known normalized, bracket-operator based constraint violation functions [3, 6] can be used for this purpose. Like in M1, the constructed metamodels can also be used to find one Pareto-optimal solution at a time as a generative approach (we call it M2-1) or simultaneously like in an EMO approach (we call it M2-2).

The next metamodeling framework approximates each constraint function independently, but metamodels a combined objective function involving all M objectives together. For this purpose, any scaralization based multi-objective optimization approach [36, 7] can be used.

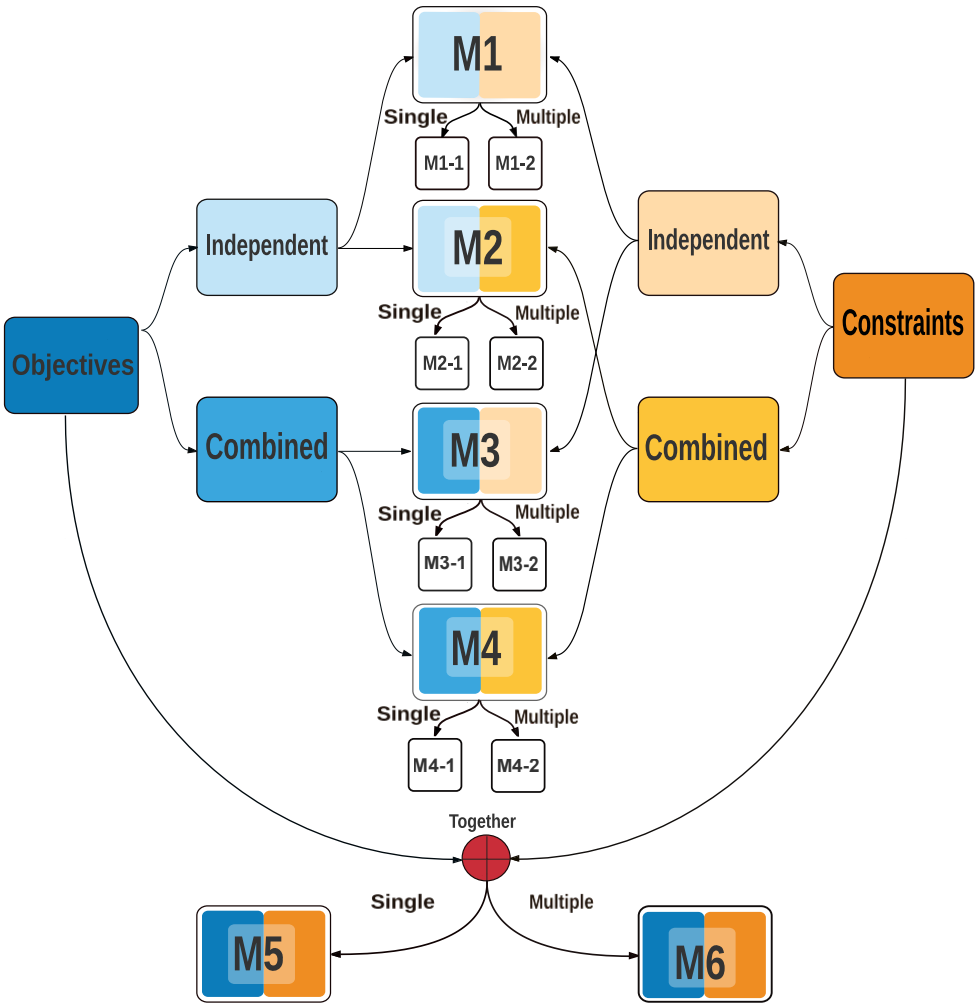

Figure 2.2: The proposed taxonomy of six different metamodeling frameworks for multiple and many-objective optimization.

Thus, this M3 approach requires $(J + 1)$ metamodels. Since a scalarized formulation finds a single Pareto-optimal solution at a time, the M3-1 approach, by default, must be applied multiple times, each time finding a single Pareto-optimal solution. Framework M3-2 can also be proposed to find multiple solutions in a single step. Then, our fourth classification (M4) requires only two metamodels to be constructed at each iteration, in which one metamodel is for a combined objective function and the second metamodel is made for a combined constraint violation (like in M2 approach). Due to the use of scalarization function to

combine all M objectives, M4-1 must also be applied multiple times to find a representative set of Pareto-optimal solutions. Framework M4-2 finds a representative set in a single step of the algorithm. The methods (M1-1, M2-1, M3-1 and M4-1) are ideal for classical point-based optimization algorithms, each requiring multiple applications to find multiple Pareto-optimal solutions. However, methods (M1-2, M2-2, M3-2 and M4-2) are ideal for EMO approaches. Methods M3-1 and M4-1 can also be followed using other scalarized EMO approaches as well [7].

A deeper thought will reveal that there could be two more frameworks, in which objectives and constraints are somehow combined to have a single overall *selection* function which when optimized will lead to one or more Pareto-optimal solutions. In M5, the combined selection function has a single optimum coinciding with a specific Pareto-optimal solution and in M6, the combined selection function is multi-modal and makes multiple Pareto-optimal solutions as its optima. Both M5 and M6 methods involve a single metamodel in each iteration, but if K Pareto-optimal solutions are to be found, M5 needs to be applied K times, whereas M6 still involves a single multi-modal metamodel in finding a set of Pareto-optimal solutions. In EMO algorithms, such as in NSGA-II [37], NSGA-III [38], MOEA/D [39] and others, the combined action of the selection operator involving non-domination and niching operations is an ideal way of visualizing the selection function mentioned above. In this spirit, we believe that M5 and M6 are intricately advantageous for EMO approaches and although has not been paid much attention, remain as potential and fertile areas for metamodeling based EMO algorithms. In this thesis we explore some algorithms related to M6 framework.

Thus, it is observed that according to our proposed taxonomy, methods M1 to M6 require the maximum possible metamodels ($M+J$) to single metamodel in each iteration of the multi-objective metamodeling algorithms. While M6 requires the minimum number of metamodels,

this does not come free and it is expected that the complexity of the metamodels will become more and more from M1 to M6. It then becomes an interesting research task to identify a balance between the number of metamodels and the reduced complexity of metamodels for a particular problem-algorithm combination. In this thesis, we do not study the effect of algorithm per se, but present results of a particular approach on different problems using all six metamodeling methods to illustrate each method's potential in different problems.

On a survey of many existing multiple and many-objective metamodeling studies, we have made a classification of them according to our proposed taxonomy given in Table 2.1. If

Table 2.1: Literature review.

Methodology	References	Metamodel
M1-1	[40]	Arti cial Neural Network (ANN)
	[41, 42, 43]	Radial Basis Function (RBF)
	[44, 45]	Support Vector Machines (SVM)
M1-2	[46, 47, 48, 49, 50, 51, 24, 52, 53]	Kriging (KRG)
	[54]	Genetic Programming (GP)
	[55]	KRG+Polynomial Response Surfaces (PRS)
	[56, 57, 58]	KRG+RBF
	[59]	KRG+SVM
	[60, 61, 62, 63, 64]	RBF
	[65, 66]	SVM
M2-2	[67]	KRG
M3	[68]	KRG+RBF+PRS
	[69]	Moving Least Squares (MLS)
	[70]	KRG+ Polynomial Chaos Expansions (PCE)
	[71]	SVM
M4	[72]	RBF+SVM
	[73, 74, 75]	KRG
M5	[76]	KRG

the multi-objective optimization problem is unconstrained, frameworks M1 and M2 becomes identical and so are M3 and M4. Interestingly, M3, M4, and M5 also become identical to each other. Figure 2.3 shows the resulting taxonomy of metamodeling methods in this case. The proposed taxonomy for multi-objective metamodeling frameworks also degenerates to single-objective problems. Figure 2.4 shows the resulting degenerate taxonomy for finding a single optimum in a single-objective problem. In this case, frameworks M1 and M3 are identical and

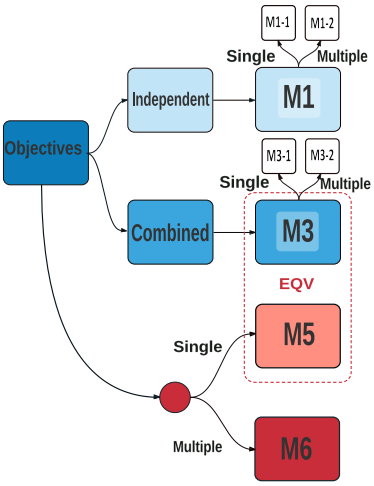

Figure 2.3: Degenerated taxonomy for multi-objective unconstrained optimization.

so are M2 and M4. A similar taxonomy can also be derived for finding multiple optima in a single-objective optimization problem, except that M6 framework will now become relevant. Sub-frameworks M1-1 and M1-2 become relevant in determining whether a single optimum at a time or multiple optima simultaneously, respectively, would be found. Similarly, sub-frameworks M2-1 and M2-2 will also be relevant in this case. For brevity, we do not present the respective diagram for single-objective, multi-modal optimization case here.

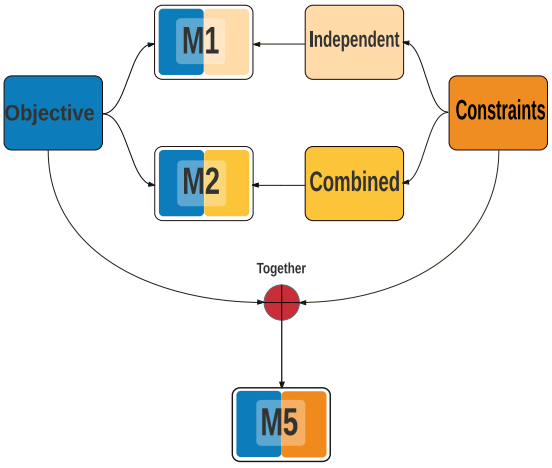

Figure 2.4: Degenerated taxonomy for single-objective optimization for finding a single optimal solution.

Before we discuss the simultaneous frameworks for Metamodel-based optimization in the

next chapter, here we briefly discuss the generative frameworks – M1-1, M2-1, M3-1, M4-1 and M5.

2.4 Generative Frameworks

Mathematically speaking, the algorithms of generative framework targets one Pareto-optimal point while we optimize the model space. The targeted Pareto-optimal point is selected based on uniformly generated reference points in objective space using Das and Dennis's [77] method. A set of trade-off solutions can be obtained by optimizing all the corresponding single-objective sub-problems sequentially. The near-optimal solution is obtained by optimizing in the model space and it is then exactly evaluated and included for solving next sub-problems. Usually, a single-objective meta-heuristic algorithm e.g real-coded genetic algorithm (RGA) is employed for such optimization. Based on modeling separate or aggregation function of objectives and constraints, the algorithms are divided into five different frameworks. When objectives and constraints are modeled separately we have the algorithm M1-1. When objectives are modeled separately but aggregation function of constraints (e.g. CV(.)) is modeled, we have M2-1. When constraints are aggregated using scalarized function (e.g. ASF) but not the objectives we have M3-1. When we build models for aggregation function of both objective and constraints, then we have M4-1. And the last but not the least, in framework M5 all objectives and constraints are combined using some aggregated function then we build metamodel of that. The difference between M5 and M6 is that we target only one Pareto-optimal solution at a time in the model space created by M5 framework.

2.5 Summary of the Chapter

In this chapter, we have presented a taxonomy for metamodeling frameworks for evolutionary multi-objective optimization with a literature review that relates to our study. Moreover, we have provided a brief overview of each of the six frameworks. Additionally, we have discussed the degenerated taxonomy to unconstrained multi-objective optimization and single-objective optimization problems. Generative frameworks of Metamodel-based algorithm, which is out of the scope of this thesis, are presented briefly.

Chapter 3

Simultaneous Frameworks for Metamodel-based Optimization

3.1 Introduction

Every Metamodel-based optimization, in general, can be divided into two stages. The first stage is to build the metamodels for objectives and constraints either separately or for some aggregated functions of them. In the next stage, a multi-objective evolutionary algorithm is applied to find optimal trade-off solutions in the model space. No exact function evaluation is carried out in this step thus it is often called *low-fidelity optimization*. After that, the newly found solutions are evaluated exactly and included in the model. These solutions may not be the true optimum since the metamodel predictions might not be highly accurate. The near-optimal solution can be found by executing successive steps when our metamodels become increasingly more accurate. There may be a natural variation of algorithms based on what function (aggregation or single) should be modeled. The algorithm can also be different based on number of infill points returned by the algorithm for exact evaluation. For a multi-objective optimization algorithm diversity of solutions is also important along with the convergence of solutions.

Based on the number of near-optimal solutions found in a single step of the algorithm, we divide the metamodeling algorithmic framework into two subdivisions– generative and

simultaneous frameworks. One can provide one or multiple infill points from low-fidelity optimization. In contrast to generative method (M1-1, M2-1, M3-1, M4-1 and M5 briefly discussed earlier) that returns one solution in each step, simultaneous frameworks (M1-2, M2-2, M3-2, M4-2 and M6) target either all or a prespecified number of Pareto-optimal solutions in model space. There are some advantages of simultaneous frameworks over generative ones. A simultaneous framework is favorable to batch process of high-fidelity evaluation. Often, computationally expensive objectives are carried out in a multi-threaded or parallel environment in a batch mode. Sequential estimation of solutions would require more time if batch evaluation facility is available. Generative frameworks also require more time for low-fidelity optimization. For example, targeting r Pareto-optimal solutions would increase the number of metamodels and frequency of low-fidelity optimization by the factor r.

3.2 Selection Function in Simultaneous Framework

Selection function plays an important role in simultaneous frameworks of Metamodel-based optimization. Given a set of generated solutions in low-fidelity optimization, the solutions returned by the EMO algorithm depends on the selection procedure i.e. selection function. To get an idea of the selection function, we take two test problems ZDT1 [1] and TNK [78], and show the model space created by different frameworks. Here we have used two variable ZDT1 problem. The procedure of creating these figures is as follows. For each problem, we first create a random sample of 30 points as a training set from the search space comprised of variables x_1 and x_2. We create test set using a grid (50x50) of 2500 solutions over x_1 and x_2. NSGA-II procedure assigns a rank for each solution based on non-domination relationship of the population. For example, the solutions not dominated by any other solution are in rank

1, the solutions dominated by rank 1 solutions are in rank 2 and so on. The solutions within the same rank are again ranked based on their crowding distance. For example, if there are 10 solutions in the first rank, we put ranks $1, 1.1, \ldots 1.9$ with interval 0.1 based on crowding distance. All the solutions of the first rank have lower final rank than the solutions of the second rank and so on. We then plot the ranks as Selection(x) in z-axis.

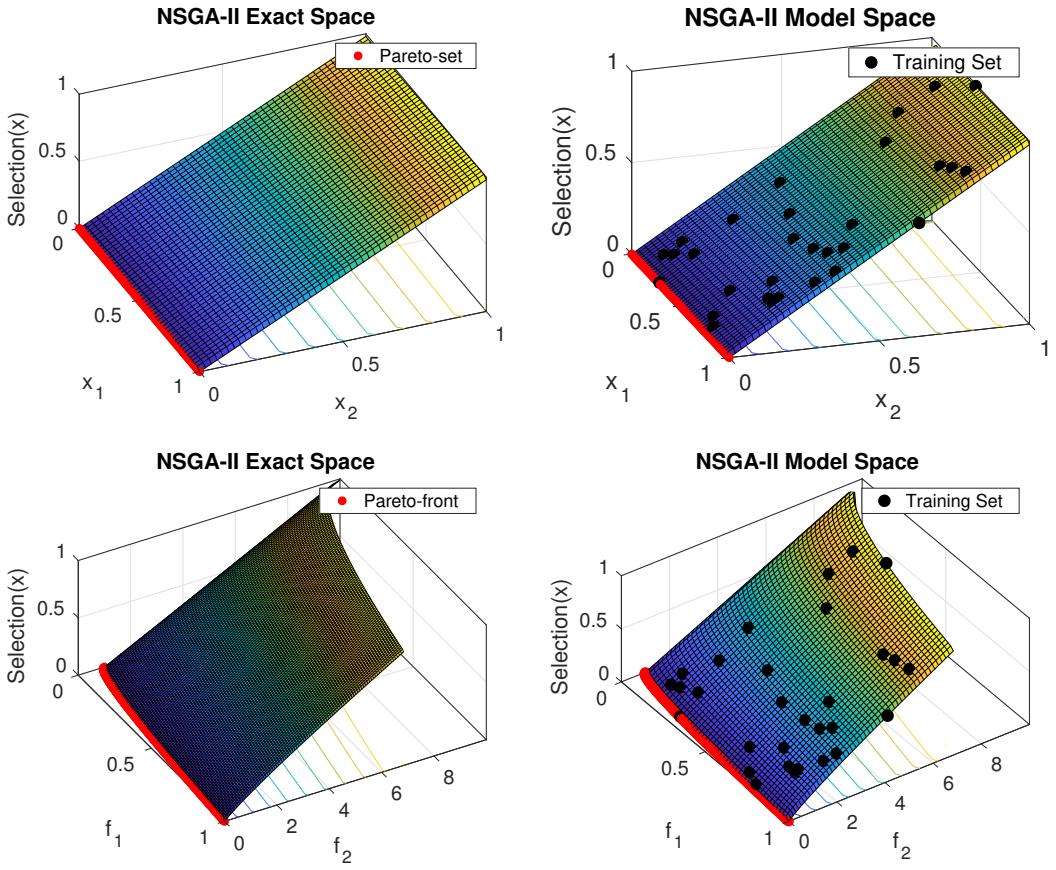

Figure 3.1: Fitness landscape or selection values provided by the selection function of M1-2 framework (NSGA-II selection used here) for ZDT1 test problem w.r.t. variable and objective space respectively. Although we have a finite number of samples, the model is able to capture the direction towards Pareto-optimal front. In each step it targets multiple optimal solutions.

We then build metamodels for each objective and constraint and perform NSGA-II selection function over the predicted objectives and constraints in the same way. We create a surface using the test set and show them with respect to both variables and objectives in

Figure 3.1. The model space created by this method shows that the lowest selection values are associated with near Pareto-optimal points. The low-fidelity optimization algorithm will find near-optimal solutions according to current model space. The newly found solutions are then used to rebuild the models which are supposed to be more accurate and close to the exact NSGA-II selection function (on the left in the figure). The merit of modeling the selection function is that it generates a good distribution of points at each step thereby helping to build better metamodels for the next step. We now briefly demonstrate the working procedures of simultaneous frameworks.

3.3 Framework M1-2 and M2-2

Frameworks M1-2 and M2-2 are two of the most popular frameworks among the researchers. In framework M1-2, each objective and constraints are separately modeled. After model construction, M1-2 finds multiple optimum solutions in each epoch. In framework M2-2, each objective is separately modeled but all the constraint functions are combined using new aggregate constraint violation function (ACV) and then this combined function is modeled. Aggregate Constraint violation function is defined in the following way. If a solution $\mathbf{x}$ is feasible then we sum up the negative values to get the amount of feasibility. If the solution is infeasible, we convert all feasible constraints ($\forall j \; g_j(\mathbf{x}) \leq 0$) from negative to zero and then accumulate the violation of infeasible constraints ($\forall j \; g_j(\mathbf{x}) > 0$) .

$$\text{ACV}(\mathbf{x}) = \begin{cases} \sum_{j=1}^{J} \bar{g}_j(\mathbf{x}), & \text{if } \mathbf{x} \text{ is feasible,} \\ \sum_{j=1}^{J} \langle \bar{g}_j(\mathbf{x}) \rangle, & \text{otherwise.} \end{cases} \tag{3.1}$$

Here, the bracket operator $\langle\alpha\rangle$ is α if $\alpha > 0$ and zero, otherwise. The function $\bar{g}_j$ is a normalized version of constraint function g_j. We define CV() function similar to ACV() with the only difference that CV($\mathbf{x}$) is 0 whenever $\mathbf{x}$ is feasible rather than having aggregated negative values. We illustrate the steps of M1-2 and M2-2 as follows. We create a set of uniformly distributed reference directions (W) on a unit simplex using Das and Dennis Method [79]. The number of reference directions are the same as the number of infill points returned by the algorithm in each epoch. The algorithm creates an archive of μ initial population using Latin hypercube method [80] on entire variable space. Then, metamodels are constructed for all M objectives ($f_j(\mathbf{x}),\ j = 1, 2 \ldots, M$). For M1-2, each constraint function is modeled separately and for M2-2, only one aggregate constraint violation function (ACV($\mathbf{x}$)) is modeled. Then, a multi-objective evolutionary algorithm is run for τ generations to get the non-dominated front. Here we have used NSGA-II for two-objective and NSGA-III for many-objective problems. If the number of non-dominated solution is more than $H = |W|$, we select one solution that minimizes the $\text{ASF}^{(h)}$ function corresponds to each reference direction $w^{(h)} \in W$. The objective values are normalized using population minimum and population maximum before calculating ASF. The ASF formulation is given by the equation.

$$\text{ASF}^{(h)}(\mathbf{x}) = \max_{i=1}^{M} \frac{f_i(\mathbf{x}) - z_i}{w_i^{(h)}}. \tag{3.2}$$

where z is a vector of population-minimum. In each epoch, H solutions are then included in the archive. New metamodels are then created again and the process is repeated until we utilize specified number of solution evaluations. The frameworks are outlined in Algorithm 1 [81].

Algorithm 1: Framework M1-2 and M2-2

Input : Objectives: $[f_1, \ldots, f_M]^T$, constraints: $[g_1, \ldots, g_J]^T$, n (variables), ρ (sample size), E (maximum high fidelity solution evaluations), EMO (multi-objective evolutionary algorithm), τ (EMO generations per metamodel), μ (EMO's population size), Γ (other parameters of EMO), CV (constraint violation function),

Output: P_T

$t \leftarrow 0$;
$\mathrm{P}_t, \mathrm{F}_t, \mathrm{G}_t \leftarrow \emptyset$;
$\mathrm{P}_{new} \leftarrow \mathrm{LHS}(\rho, n)$// Initial solutions
$e \leftarrow |\mathrm{P}_{new}|$;
while true do
  Calculate objectives $\mathrm{F}_{new} = \{f_i(\mathrm{P}_{new}),\ \forall i \in \{1, \ldots, M\}\}$;
  Calculate constraints $\mathrm{G}_{new} = \{g_j(\mathrm{P}_{new}),\ \forall j \in \{1, \ldots, J\}\}$;
  // merge archive
  $\mathrm{P}_{t+1}, \mathrm{F}_{t+1}, \mathrm{G}_{t+1} \leftarrow (\mathrm{P}_t \cup \mathrm{P}_{new}), (\mathrm{F}_t \cup \mathrm{F}_{new})$ and $(\mathrm{G}_t \cup \mathrm{G}_{new})$;
  $e \leftarrow e + |\mathrm{P}_{new}|$// total evaluations
  break if $e \geq E$// termination
  // Build Metamodels for Objectives
  $\widehat{f}^i_{t+1} \leftarrow \textsc{Metamodel}(\mathrm{P}_{t+1}, \mathrm{f}^i_{t+1}),\ \forall i \in \{1, \ldots, M\}$;
  if M1-2 **then**
    $\widehat{g}^j_{t+1} \leftarrow \textsc{Metamodel}(\mathrm{P}_{t+1}, \mathrm{g}^j_{t+1}), \forall j \in \{1, \ldots, J\}$;
  else if M2-2 **then**
    $\widehat{V} \leftarrow \mathrm{CV}(\mathrm{G}_{t+1})$;
    $\widehat{g}_{t+1} \leftarrow \textsc{Metamodel}(\widehat{V}_{t+1})$;
  // Optimize model space
  $\mathrm{P}_{new} \leftarrow \mathrm{EMO}(\widehat{F}_{t+1}, \widehat{g}_{t+1}, \mu, \tau, \Gamma, E\text{-}e)$;
  $t \leftarrow t + 1$;
return $\mathrm{P}_T \leftarrow$ filter the best solutions from P_{t+1}

3.4 Framework M3-2 and M4-2

In frameworks M3-2 and M4-2 we transform the multi-objective optimization problem into a multi-modal one. Instead of finding the entire front, here we are required to find only $H = |Z|$ well-distributed solutions. Thus, we use achievement scalarization function (ASF) using reference points $z^{(h)} \in Z$. Reference directions are kept equi-spaced and equally-angled

from each objective axis i.e. $w_i^{(h)} = 1 \ \forall i$. The ASF formulation is given by the equation.

$$\mathrm{ASF}^{(h)}(\mathbf{x}) = \max_{i=1}^{M} \left(f_i(\mathbf{x}) - z_i^{(h)} \right) \tag{3.3}$$

Similar to frameworks M1-2 and M2-2, both algorithms M3-2 and M4-2 starts with an archive of points created using LHS. Each member is then evaluated exactly. We construct one metamodel for each ASF function ($|Z|$ of them) instead of modeling each objective functions separately given by M1-2 and M2-2. Constraints are modeled either independently (in M3-2) or in a combined way (in M4-2 ACV() is modeled). Generative methods like M3-1 and M4-1 optimizes each ASF functions separately. In contrast, we return multiple infill points by optimizing all ASF functions simultaneously. A multi-objective real-coded genetic algorithm (MO-RGA) is applied to get H trade-off points from the current epoch. Hence we define a selection function based on the predicted value of ASF and ACV. If the solution is feasible it gets the same value as ASF function. Otherwise, we compute the maximum ASF value of over all feasible solutions and then add it to the constraint violation (CV) function obtained either from predicted constraints values (M3-2) or predicted ACV (M4-2, see Eqn. 3.4). Thus all the infeasible solutions are worse than the feasible ones.

$$\mathcal{S}^{(h)}(\mathbf{x}) = \begin{cases} \widehat{\mathrm{ASF}}^{(h)}(\mathbf{x}), & \text{if } \mathbf{x} \text{ feasible} \\ \max_{i=1}^{|P_f|} \{\widehat{\mathrm{ASF}}^{(h)}(\mathbf{P}_f^i)\} + \mathrm{CV}(\widehat{ACV}(\mathbf{x})), & \\ \text{otherwise} & \end{cases} \tag{3.4}$$

Here P_f is the population of feasible solutions in MO-RGA and $\widehat{\mathrm{ASF}}(\mathbf{x})$ and $\widehat{\mathrm{ACV}}(\mathbf{x})$ are the predicted values of ASF and ACV. We sort the entire population by each of the H selection functions. Then we then pick the best solution from each division h. If one or

more solutions are repeated, we continue picking second-best solutions and so on until we pick μ solutions for the next generations. The obtained H solutions are then included in the archive to complete the current epoch.

Algorithm 2: Framework M3-2 and M4-2

Input : Objectives: $[f_1, \ldots, f_m]^T$, Constraints: $[g_1, \ldots, g_J]^T$, n (variables), ρ (initial sample size), E (maximum high-fidelity solution evaluations), MO-RGA (multi-modal evolutionary algorithm) with population size μ, generations for model optimization τ and other parameters Γ, Reference directions W, ASF function, CV (constraint violation function)

Output: P_T

$t \leftarrow 0$;
$\mathrm{P}_t, \mathrm{F}_t, \mathrm{G}_t \leftarrow \emptyset$;
$\mathrm{P}_{new} \leftarrow \mathrm{LHS}(\rho, n)$// Initial solutions
$e \leftarrow |\mathrm{P}_{new}|$;
while true do
  Calculate objectives $\mathrm{F}_{new} = \{f_i(\mathrm{P}_{new}),\ \forall i \in \{1, \ldots, M\}\}$;
  Calculate constraints $\mathrm{G}_{new} = \{g_j(\mathrm{P}_{new}),\ \forall j \in \{1, \ldots, J\}\}$;
  // merge archive
  $\mathrm{P}_{t+1}, \mathrm{F}_{t+1}, \mathrm{G}_{t+1} \leftarrow (\mathrm{P}_t \cup \mathrm{P}_{new}), (\mathrm{F}_t \cup \mathrm{F}_{new})$ and $(\mathrm{G}_t \cup \mathrm{G}_{new})$;
  $e \leftarrow e + |\mathrm{P}_{new}|$// total evaluations
  break if $e \geq E$// termination
  // compute selection function for each w
  $\mathcal{S}_{t+1}^{(h)} \leftarrow \mathcal{S}^{(h)}(\mathrm{P}_{new}),\ \forall w^{(h)} \in W$;
  // construct metamodels
  $\widehat{\mathcal{S}}_{t+1}^{(h)} \leftarrow \textsc{Metamodel}(\mathrm{P}_{t+1}, \mathcal{S}_{t+1}^{(h)}), \forall\ w^{(h)} \in W$;
  if M3-2 **then**
    $\widehat{g}_{t+1}^{j} \leftarrow \textsc{Metamodel}(\mathrm{P}_{t+1}, \mathrm{g}_{t+1}^{j}), \forall j \in \{1, \ldots, J\}$;
  else if M4-2 **then**
    $\widehat{V} \leftarrow \mathrm{CV}(\mathrm{G}_{t+1})$;
    $\widehat{g}_{t+1} \leftarrow \textsc{Metamodel}(\widehat{V}_{t+1})$;
  // Optimize model space
  $\mathrm{P}_{new} \leftarrow \text{MO-RGA}(\widehat{\mathcal{S}}_{t+1}, \widehat{g}_{t+1}, \mu, \tau, \Gamma, E\text{-}e)$;
  $t \leftarrow t + 1$;
return $\mathrm{P}_T \leftarrow$ filter the best solutions from P_{t+1}

For framework M3-2 and M4-2, the algorithm targets multiple Pareto-optimal solutions (H of them) where each of them is specified by the scalarized function ASF. Finding one

solution at a time may not provide a good intermediate distribution of solutions which is necessary to build better metamodels in the successive epochs. This is because diversity among the infill points may not be ensured when we incrementally return the best converged solution from the model space as done in M3-1 and M4-1. Framework M3-2 and M4-2 provides a way to generate a good distribution of infill points in a multi-objective manner. Now if we compare the execution time, frameworks like M3-1 and M4-1 needs to perform low-fidelity optimization H times, thereby increasing the running time by H times compare to M3-2 and M4-2. The algorithm is outlined in Algorithm 2.

3.5 Framework M-6

In contrast to other frameworks that create multiple metamodels, this framework can construct only one or fewer models that target multiple Pareto-optimal solutions. Hence the method requires the least amount of time for low-fidelity optimization.

3.5.1 Metamodeling the Selection Function

EMO algorithms are mostly different from each other in their way of constructing the selection operator. Having multiple conflicting objectives and multiple constraints to be satisfied, an EMO's selection operator treats two aspects essential for converging near to the Pareto-optimal front and for finding a diverse set of solutions: (i) emphasis on non-dominated solutions and (ii) emphasis on diverse solutions. NSGA-II [82] uses a non-dominated sorting procedure for the entire population at any generation to achieve the first aspect and uses a front-wise crowding distance operator to achieve the second aspect. NSGA-III [38] uses the non-dominated sorting for the first aspect, but uses a more complex niching operator

based on a set of given reference directions (W) to achieve the second aspect. No matter what EMO algorithm is considered, it makes a balance between these two aspects to finally provide a ranked list of the population at any generation. Figure 3.1 shows selection function after modeling each objective separately.

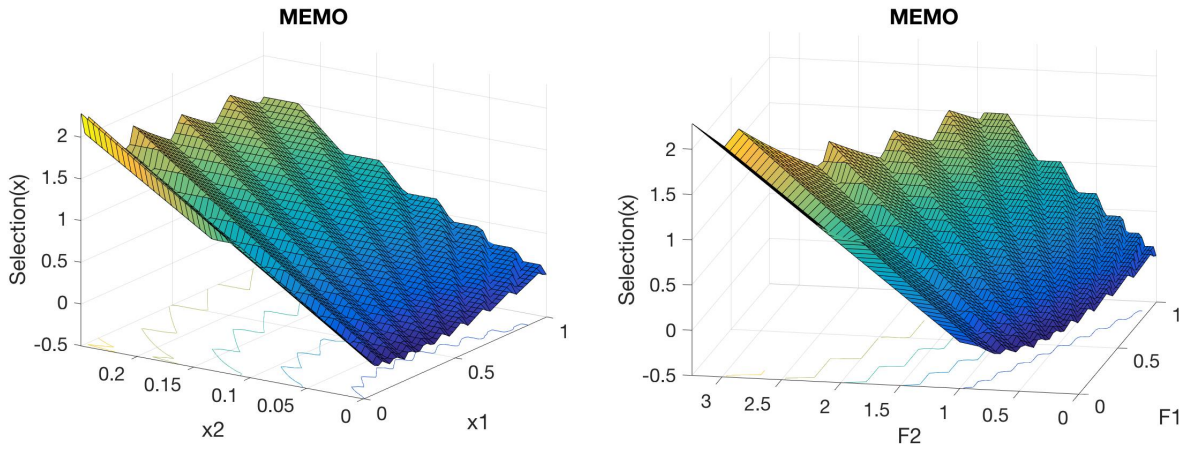

Figure 3.2: Multi-modal selection function in variable and objective space targeting prespecified number of Pareto-optimal solutions.

We now argue that instead of metamodeling each of the two objective functions separately and accumulating approximations from two metamodels, if a single metamodel is performed to approximate the above selection function, the number of metamodeling effort can be reduced. Theoretically, such a selection function has infinite optima, but if we are interested in finding H ($|W| = H$, a finite size) Pareto-optimal solutions dictated by a set of H prespecified reference directions, the above selection function will reduce to a H-modal selection function. Such an idea is novel for multi- and many-objective optimization and the procedure shields the number of objectives and constraints from the number of metamodeling efforts that must be performed.

3.5.2 MEMO Based Approach

Recently, a multi-modal based EMO or MEMO approach [83] was proposed by constructing a multi-modal single-objective function from a multi- or many-objective problem, which possess a finite number of multiple global optima. The location of each optimum is determined by the reference direction. The MEMO approach used the Achievement Scalarization Function (ASF). The selection function of M6 framework for low-fidelity optimization is described here. Initially, we create reference points for ASF similar to other frameworks e.g. M3-2. The objective and constraint functions of the population are first evaluated with high-fidelity evaluation. Based on the reference points, we calculate unimodal selection function using Eqn. 3.4 for exactly evaluated points. We then take the minimum value among all $\mathcal{S}^{(h)}$ to get the multi-modal selection function given below.

$$\mathcal{SF}(\mathbf{x}) = \min_{h=1}^{H} \mathcal{S}^{(h)}(\mathbf{x}) \tag{3.5}$$

In this framework, we build one metamodel for this function $\mathcal{SF}$. Due to its complicated nature, we build a neural network model for $\mathcal{SF}$. The whole algorithm is presented in Algorithm 3.

3.5.3 KKT Proximity Measure Based Approach

The above idea of metamodeling the underlying selection function of an EMO algorithm, instead of metamodeling each objective and constraint function, opens the door for trying the idea on successful EMO algorithms. Karush-Kuhn-Tucker (KKT) proximity measure was recently developed [84] to determine the level of convergence of non-dominated solutions in an EMO algorithm. At any point, the KKTPM value can be computed by using (exact

Algorithm 3: Framework M6

Input : Objectives: $[f_1, \ldots, f_M]^\top$, constraints: $[g_1, \ldots, g_J]^\top$, n (variables), ρ (initial sample size), E (total high fidelity evaluations), N-RGA (multi-modal real-parameter genetic algorithm with population size μ), Γ (parameters of RGA), W (reference direction set), $\mathcal{SF}$ (multi-modal constrained selection function)

Output: P_T

$\mathrm{P} \leftarrow \mathrm{LHS}(\rho, n)$ `// initialization with Latin Hypercube Sampling`
$\mathrm{F} \leftarrow f_m(\mathrm{P}), \forall m \in \{1, \ldots, M\}$ `// high fidelity evaluations (functions)`
$\mathrm{C} \leftarrow g_j(\mathrm{P}), \forall j \in \{1, \ldots, J\}$ `// high fidelity evaluations (constraints)`
$e \leftarrow \rho$ `// number of function evaluations`
while $e < E$ **do**
  for $w \in W$ **do**
    `// for each reference direction` w
    $\mathrm{L}_w \leftarrow$ Sort P according to distance from w and pick the best solution
  $\mathrm{L} = \{\mathrm{L}_1, \ldots, \mathrm{L}_{|W|}\}$ `// vector of` $|W|$ `leaders`
  $\mathrm{Fitness} \leftarrow \mathcal{SF}(\mathrm{F}, \mathrm{C})$ `// Compute selection function`
  $\mathcal{F} \leftarrow$ Metamodel(Fitness) `// Surrogate model for selection function`
  $\mathrm{X} \leftarrow$ N-RGA$(\mathcal{F}, \mathrm{L}, \mu, \Gamma)$ `// returns multiple optimized solutions, one for each reference line; niching is performed in x-space with` L
  if $|\mathrm{X}| + |P| > E$ **then**
    $\mathrm{X} \leftarrow \mathrm{X}(1 : (E - |P|))$ `// Choose best` $(E - |P|)$ `modeled solutions`
  $\mathrm{F}_{\mathrm{X}}^m \leftarrow f_m(\mathrm{X}), \forall m \in \{1, \ldots, M\}$ `// Evaluate objectives of` X
  $\mathrm{C}_{\mathrm{X}}^j \leftarrow g_j(\mathrm{X}), \forall j \in \{1, \ldots, J\}$ `// Evaluate constraints of` X
  $\mathrm{P} \leftarrow \mathrm{P} \cup \mathrm{X}$;
  $\mathrm{F} \leftarrow \mathrm{F} \cup \mathrm{F}_{\mathrm{X}}$;
  $\mathrm{C} \leftarrow \mathrm{C} \cup \mathrm{C}_{\mathrm{X}}$;
  $e \leftarrow e + |\mathrm{X}|$;
return $\mathrm{P}_T \leftarrow \mathrm{P}(1 : |W|)$

or numerical) gradients of objectives and constraint functions. We have used this KKT proximity measure as selection function and build one metamodel directly for this measure. This leads to a new KKT proximity measure based M6 framework.

3.5.4 Steps of M6 Framework

Here we briefly discuss one sample algorithm based on M6 framework.

Initialization: Due to the multi-modal structure of the M6 surrogate model, it is expected that an adequate number of initial points are required to have a reasonable starting metamodel of the problem. To conduct an effective search, we require a good representation of solutions over different parts of the objective space. Although simple Latin Hypercube Sampling is enough for creating good representative solutions, some variable density problems e.g. ZDT6 [1], DTLZ4 [85] etc. require a more sophisticated initialization procedure to get relatively uniform distribution in the objective space. Here we propose a methodology for initialization which is solely based on diversity. First, we create η solutions which is a fraction of the initial population of size ρ using the LHS sampling and evaluate them with high-fidelity evaluations. We then calculate the average distance of each solution from its τ nearest neighbors in the objective space. This average distance is then used to locate new sample point in the search space in a non-uniform manner. New points are added in slabs of $\eta\%$ at a time to fill up the whole initial population. Crossover and mutation probability is set as 1.0 and $1/n$ (where n is the number of variables), respectively. We create η solutions each time and fill up P with ρ solutions in total.

Figure 3.3 shows the effect of incremental initialization procedure on ZDT6 problem which has a bias of solutions on the larger f_1 values.

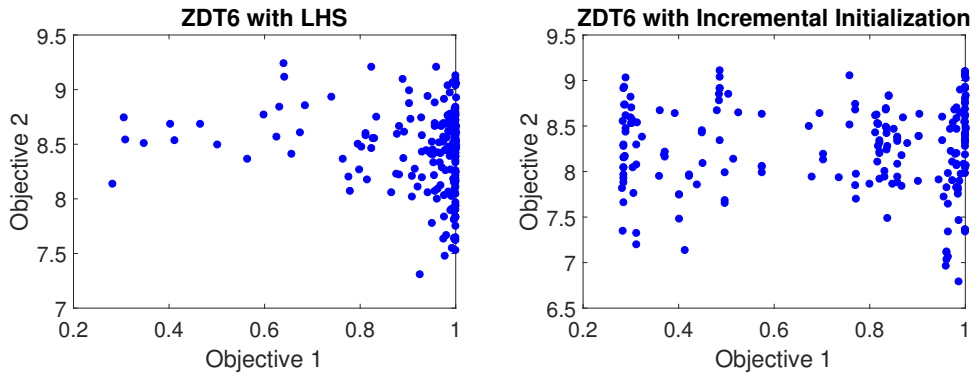

Figure 3.3: Distribution of 200 samples using (a) LHS and (b) incremental initialization.

Assigning Leader Solutions: Leader solutions L_w for each cluster w are selected based on the high-fidelity fitness function. First, we sort the population according to orthogonal distances from each of the reference directions. Each solution is assigned to the nearest reference direction. Thus, there are $|W|$ possible clusters, although some clusters may not contain any member. We assign the best solution for each cluster to be the leader of that cluster. This leader is then used for niching in N-RGA by using the nearest leader in the variable space.

Creating Surrogate Model: As described before, we used two multi-modal selection function $\mathcal{SF}(.)$ that can constitute multiple optima, each for a specific Pareto-optimal solution. Each reference vector $w \in W$ targets one global optimum in general. To make the model more accurate, only solutions which are evaluated exactly are used for model construction. In this study, we use either Kriging or Artificial Neural Network method to perform this step. The choice of our models is two-fold. First, we want to investigate the performance with a structured modeling procedure (Kriging). Knowing the fact that a Kriging method may be difficult to model a multi-modal landscape, we include an ANN method to model such a complex landscape, particularly motivated by the recent progress on deep learning [86] method's ability to model any arbitrarily complex relationship. The comparison between Kriging and ANN in the context of M6 metamodeling approach is an important part of this study.

Niched Real-parameter Genetic Algorithm (N-RGA): To solve a multi-modal problem for finding multiple optimal solutions, we also need an optimization algorithm that is capable of finding multiple solutions. We devise a niching based genetic algorithm which preserves solutions from each niche with the help of leader solutions, described above. Objective functions and constraints are provided by Kriging or ANN, whichever is being used. A

mating selection is used to restrict parents to be chosen from the same cluster. We use other parameters similar to a standard real-parameter genetic algorithm [3]. At the end algorithm N-RGA that runs with μ population, we pick at most one solution from each cluster for further high-fidelity evaluation. After we finish this cycle, we re-evaluate the leader of each cluster for the next iteration. Experimental results of this approach is presented in Chapter 6.

3.6 Summary of the Chapter

In this chapter, we have discussed simultaneous frameworks for Metamodel-based optimization algorithms. We have presented popular frameworks like M1-2, and M2-2 as well as some new frameworks like M3-2, M4-2 and M6. Detailed algorithms have been presented for these frameworks. In the next chapter, we shall discuss more efficient versions of these methods.

Chapter 4

Efficient Search Strategy

Computationally expensive problems are challenging primarily because of limited function evaluation. In practice, we can restrict our search space near better solutions and gradually increase or decrease the region based on algorithmic performance. In this chapter we have developed trust region method for Metamodel-based multi-objective optimization. We have also developed new algorithm for one of the basic steps of multi-objective evolutionary algorithm.

4.1 Trust Region Method

Most of the real-world problems involve time-consuming experiments and simulations that cause optimization to be increasingly expensive. To face this challenge and to reduce the computational cost, metamodels as approximations of exact models are used for the optimization task. Although most existing methods are directed towards proposing more accurate metamodels or introducing efficient search schemes, there is a need for managing error uncertainty for an under-performing metamodel during optimization. A better management of a metamodel can, not only restrain the model from becoming worse, but also boost the performance by recognizing the inherent complexity of search regions. In this chapter, we introduce a trust region concept for multi-objective optimization to reduce model uncertainty during Metamodel-based optimization. This allows a progressive convergence towards Pareto-front

instead of relying completely on assumptions of the metamodel from the first iteration on.

4.1.1 Related Studies

There have been several studies in Metamodel-based multi-objective evolutionary algorithms for constrained and unconstrained problems. ParEGO [17], MOEA/D-EGO [24] and SMS-EGO [87] use scalarization methods (e.g., Tchebycheff) to combine multiple objectives into one and solve multiple scalarized versions of them to find a trade-off set of solutions. While these methods are useful for unconstrained problems, they need to be modified for constrained scenarios. Hypervolume-based expected improvement [88] and maximum hypervolume contribution [87] are used as a performance criteria for infill points. Few recent studies [89, 90] outperformed standard evolutionary multi-objective optimization methods for unconstrained test problems.

Trust region methods are an effective mechanism to identify new infill points with a specific certainty. A few researchers have suggested using Metamodel-based optimization with a trust region concept [91, 92]. They proposed a trust region framework using approximation models with varying fidelity. Their approach is based on the trust region concept from nonlinear programming literature and was shown to be provably convergent for some problems. A sequential quadratic approximation model was used in their study. In [92], a global version of the trust region method — Global Stochastic Trust Augmented Region (G-STAR) — was proposed. The trust region was used to focus on simulation effort and balance between exploration and exploitation. They used Kriging as a metamodel for unconstrained single-objective optimization problems. Few recent studies have considered bi-objective [93] and multi-objective [94] problems with a convergence guarantee under mild conditions. While most trust region based methods are directed towards mostly single-

objective and unconstrained problems with quadratic approximations, we introduce here trust region based algorithm for multi-objective constrained and unconstrained problems using evolutionary algorithms.

4.1.2 Proposed Concept

The classical trust region method for single-objective optimization proceeds by building a metamodel $\widehat{f}(.)$ for the original objective function $f(.)$. The prediction of the metamodel $\widehat{f}(.)$ is minimized to obtain new infill points [91]:

$$\begin{aligned} &\text{Minimize}_q \quad \widehat{f}(q) \\ &\text{subject to } \|q-p\| \leq \delta_k. \end{aligned} \tag{4.1}$$

Here p is the current iterate and q is the new predicted point that can replace p in the next iteration. Typically, a quadratic model is used as $\widehat{f}(.)$. The search is restricted within a radius δ_k from the current point p so that the metamodel approximates f well. The distance $\|q-p\|$ can be calculated using any norm. Without loss of generality, we use Euclidean norm in this chapter. The trust region is updated by comparing the exact and the predicted value of the new point ($f(q)$ and $\widehat{f}(q)$) with respect to the old point p by the following equation [91].

$$r = \frac{f(p) - f(q)}{f(p) - \widehat{f}(q)}. \tag{4.2}$$

Depending on the performance indicator r, the trust region might increase, decrease or remain the same. To decide what operation should be performed, two constants r_1 and r_2 are defined and the trust region is adapted as follows:

- If the model fails to improve objective value (that is, $r < r_1$), we reduce the trust

region by multiplying existing δ_k with c_1 (< 1) and do not replace p with the new point q.

- If the model performs well at predicting function improvement from previous solution (that is, $r > r_2$), we increase δ_k for the next iteration by multiplying existing δ_k with c_2 (> 1) and we replace the old point p by new point q.

- Otherwise, we leave the trust region size δ_k as it was before.

We replace the old point p with the new point q, whenever q is a better point. The current point (p or q) is always associated with the updated trust radius. Suitable values of c_1 and c_2 are used.

4.1.3 Trust Region in Multi-Objective Optimization

The main challenges for applying the trust region concept in multi-objective evolutionary algorithms (MOEA) are handling multiple objectives and constraints. In addition, since MOEAs are population based methods, we also need to deal with multiple solutions and their trust regions. Moreover, there is a need for a meaningful performance metric to adapt trust radii of multiple high fidelity solutions. In the following subsection, we discuss our proposed concepts. The main algorithms will be discussed in Section 4.1.10.

A multi-objective optimization problem can be formulated as follows. Here, we omit the vector notation and use $\{x, p, q\}$ and F to denote a multi-dimensional point and objective

vector respectively.

$$\begin{aligned} \text{Minimize} \quad & F(x) = (f_1(x), f_2(x), \ldots, f_M(x))^T \\ \text{subject to} \quad & g_j(x) \geq 0, \quad \forall j \in \{1, \ldots, J\} \\ & x \in \Omega \subseteq \mathbb{R}^n \quad \text{and,} \quad F \in \Lambda \subseteq \mathbb{R}^M \end{aligned} \tag{4.3}$$

Here, feasible variable space and respective feasible objective space are defined by Ω and Λ, respectively. The goal of this optimization is to find the best trade-off hyper-surface. Now we propose our trust region concept for solving the above problem.

4.1.4 Proposed Trust Region Concept

We propose several modifications to the classical trust region method in order to make it applicable for Metamodel-based multi-objective evolutionary algorithms:

1. We store all high fidelity solutions in an archive A, instead of replacing them with better solutions.

2. We maintain separate trust regions in variable space for each solution. The regions might be overlapping. They can either grow or shrink in size during optimization according to the quality of prediction. The algorithm optimizes within the combined trust regions of A.

3. To compare a newly predicted point q with the neighbor point p (q is within trust region of p), we define two performance indicators PI that calculate r (analogous to Equation 4.2) for a multi-objective problem. Moreover, we propose a novel scheme to compare between feasible and infeasible solutions.

4. If the new point q is within the trust regions of multiple points $P \subseteq A$ then we update the trust radius δ_k for each of them using a pair-wise performance metric. The trust radius of point q will be the minimum of trust radii of P.

Therefore, we optimize the following Metamodel-based optimization to obtain a set of new infill points.

$$\begin{aligned} \text{minimize}_{q\in\Omega} \quad & \widehat{f}_1(q), \ldots, \widehat{f}_M(q) \\ \text{subject to} \quad & \|q - p_i\| \leq \delta_k^{p_i}, \quad \forall p_i \in A \\ & \widehat{g}_j(x) \geq 0, \quad \forall j \in \{1, \ldots, J\} \end{aligned} \tag{4.4}$$

Here $p_i \in A$ are the exactly evaluated solutions from the current archive.

Figure 4.1 illustrates the population based extension of the trust region method. Five exactly evaluated points $\{P_1, P_2, P_3, P_4, P_5\}$ with their trust regions (regions within the circles) are shown. Say, a new point P_{new} is predicted by the algorithm after optimization in the model space. Note that P_{new} is inside the trust regions of P_1 and P_2. Assuming that the performance indicator reports an improvement of P_{new} over P_2, but no improvement over P_1. Then we reduce the size of the trust region of P_2 and increase that of P_1. The trust radius of the new point will be the smallest of the trust radius sizes of P_1 and P_2.

4.1.5 Performance Indicator for Updating Trust Radius

To update the trust radius of solutions, we define two performance indicators (PI) for the solutions.

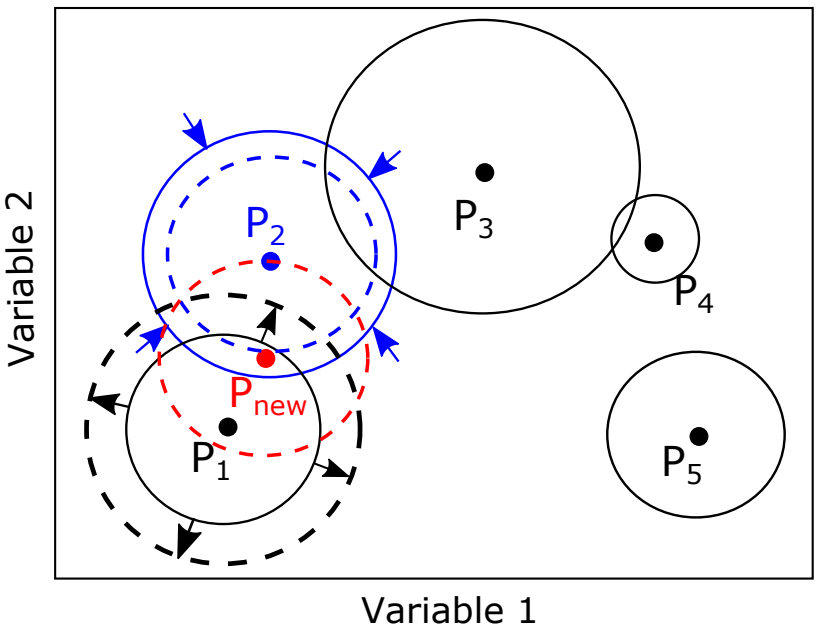

Figure 4.1: Adaptive trust region concept for multiple solutions. Newly found solution P_{new} will be responsible for updating trust radii of nearby solutions P_1 and P_2

4.1.6 Scalarization based Performance Indicator (PI_{ASF})

Scalarization method is used to convert a multi-objective problem into a number of parameterized single-objective optimization problems. We use one of these methods, namely, ASF (Achievement Scalarization Function [35]) as a performance indicator. The scalarization is based on a weight vector w and a reference point z. Reference directions are differently angled directions from each objective axis using the same reference point z. The ASF formulation is given below:

$$\text{ASF}(F(x)) = \max_{i=1}^{M} \frac{f_i(x) - z_i}{w_i}. \tag{4.5}$$

The proposed performance criteria using ASF function can be presented as follows.

$$PI_{ASF}(q) = \frac{ASF(F(p)) - ASF(F(q))}{ASF(F(p)) - ASF(\widehat{F}(q))}. \tag{4.6}$$

Note that this metric uses a direct pair-wise comparison between predicted and actual ASF values. The estimated value may differ for different reference directions.

4.1.7 Hypervolume based Performance Indicator (PI_{HV})

Hypervolume [95] is a widely used indicator in multi-objective optimization. It takes a set of solutions and a reference point, and computes the dominated region (in objective space) enclosed by the set and the reference point. In order to find the improvement of a new point over old point, we calculate the difference of their absolute hypervolume measures. We include archive points (A) as a common ground for computation. We then compute the ratio between actual improvement and predicted improvement and adjust the trust radii of old points. The predicted hypervolume is calculated by the objective values evaluated in model space using $\widehat{F}(.)$. Since larger values indicate better hypervolume, we use negative of the hypervolume.

$$PI_{HV}(q) = \frac{HV(F(A) \cup F(q)) - HV(F(A))}{HV(F(A) \cup \widehat{F}(q)) - HV(F(A))}. \tag{4.7}$$

4.1.8 Criteria for Constrained Problems

The criteria mentioned above do not consider whether the points are feasible or not. Therefore, we use constrained violation function CV, instead of ASF or HV functions, if both solutions are infeasible. The definition of CV(x) is defined in Chapter 2.

$$PI_{CV}(q) = \frac{CV(G(p)) - CV(G(q))}{CV(G(q)) - CV(\widehat{G}(q))} \tag{4.8}$$

Here G and $\widehat{G}$ are the vector representations of constraint functions $G = (g_1, \ldots, g_J)$ and their predictions $\widehat{G} = (\widehat{g}_1, \ldots, \widehat{g}_J)$.

4.1.9 Trust Region Adaptation

We now describe the procedure of updating the trust regions using the performance indicators described above. Assume that solution p is one of the high-fidelity points and q is the predicted new point which is within the trust region of p. We measure the performance improvement by the following equation.

$$r = \begin{cases} PI_{HV}(q) \text{ or } PI_{ASF}(q), & \text{if both } p \text{ and } q \text{ feasible,} \\ r_2 + \epsilon, & \text{if } p \text{ infeasible, } q \text{ feasible,} \\ r_1 - \epsilon, & \text{if } p \text{ feasible, } q \text{ infeasible,} \\ PI_{CV}(q), & \text{otherwise.} \end{cases} \tag{4.9}$$

Here $\epsilon > 0$ is a small positive real number. The predefined positive constants $0 < r_1 < r_2 < 1$ are the hyper-parameters that regulate expansion and contraction of the trust regions. After estimating performance indicator PI of a new point q with respect to old point p we update trust radius of p by the following rule.

$$\delta_{k+1}^{p} = \begin{cases} c_1 \delta_k^p, & \text{if } r < r_1 \\ \min\{c_2 \delta_k^p, \Delta_{max}\}, & \text{if } r > r_2 \\ \delta_k^p, & \text{otherwise} \end{cases} \tag{4.10}$$

The positive constants $0 < c_1 < 1$ and $c_2 > 1$ controls the size of subsequent trust radius. As mentioned earlier, we assign the trust radius of q to be the smallest of the trust radii of all neighboring solutions of which q is inside their trust regions. The parameter Δ_{max} is the largest allowed trust radius for the solutions.

4.1.10 Proposed Algorithm

Based on the proposed trust region adaptation scheme in the previous section, we briefly present two algorithms for multi-objective optimization for low-budget problems. In both algorithms, each objective and constraint are modeled separately (line 13-14 for Algorithm 4 and line 11-12 for Algorithm 5).

4.1.10.1 Scalarized Indicator-based Trust Region Method

In this algorithm, we transform the multi-objective optimization problem into a number of parameterized single-objective problems. We use the ASF scalarization function with a set of predefined reference directions W. The algorithm starts by sampling ρ initial solutions with Latin hypercube method (LHS). Initial trust radius ($\delta^i = \delta_{init}$) is assigned for each solution i. For each reference direction w, we optimize objective function given by ASF in model space. A single-objective evolutionary algorithm (RGA, real-parameter genetic algorithm) is executed for this purpose which gives an approximate best infill point for direction w. This point is then undergone a high-fidelity evaluation. After generating and evaluating $|W|$ new points, we update the trust radii of new and old solutions as discussed before. To reduce the time complexity, we do not update the trust region after each execution of RGA. The algorithm is run until the allowed number of evaluations (E) is elapsed. The main steps are presented in Algorithm 4.

4.1.10.2 Hypervolume-based Trust Region Method

Similar to Algorithm 4, the Metamodel-based algorithm starts with an archive of ρ initial population members created using the Latin hypercube method on the entire search space. The trust radii of initial solutions are then set to a predefined initial value δ_{init} as before.

Algorithm 4: Scalarized Indicator based Trust Region Method

Input : Objectives: $[f_1, \ldots, f_m]^T$, Constraints: $[g_1, \ldots, g_J]^T$, n (variables), ρ (sample size), E (maximum high-fidelity solution evaluations), RGA (real-parameter genetic algorithm) with population size μ, generations for model optimization τ and other parameters Γ, W (reference direction set), **ASF** (scalarization function), **CV** (constrained violation function), Trust region parameters– $\delta_{init}, \Delta_{max}, c_1, c_2, r_1, r_2$

Output: Solution set P_T

$t, e \leftarrow 0$;
$\mathrm{P}_t, \mathrm{F}_t, \mathrm{G}_t, \mathrm{P}_{new} \leftarrow \emptyset$;
$\mathrm{X}_{new} \leftarrow \mathrm{LHS}(\rho, n)$// Initial solutions
$\delta^i \leftarrow \delta_{init}\ \forall i \in \{1, \ldots, \rho\}$;
while *True* **do**
  for $w \in W$ **do** // for each reference direction w
    $\mathrm{F}^i_{new} \leftarrow f_i(\mathrm{X}_{new}), \forall i \in \{1, \ldots, M\}$// eval obj.
    $\mathrm{G}^j_{new} \leftarrow g_j(\mathrm{X}_{new}), \forall j \in \{1, \ldots, J\}$// eval constr.
    $\mathrm{P}_{t+1}, \mathrm{F}_{t+1}, \mathrm{G}_{t+1} \leftarrow (\mathrm{P}_t \cup \mathrm{X}_{new}), (\mathrm{F}_t \cup \mathrm{F}_{new})$ and $(\mathrm{G}_t \cup \mathrm{G}_{new})$;
    $e \leftarrow e + |X_{new}|$;
    break if $e \geq E$;
    $\widehat{f}^i_{t+1} \leftarrow \textsc{Metamodel}(\mathrm{F}^i_{t+1}), \forall i \in \{1, \ldots, M\}$ // metamodel obj.
    $\widehat{g}^j_{t+1} \leftarrow \textsc{Metamodel}(\mathrm{G}^j_{t+1}), \forall j \in \{1, \ldots, J\}$// metamodel constrt.
    $\mathrm{X}_{new} \leftarrow \mathrm{RGA}(\widehat{f}_{t+1}, \widehat{g}_{t+1}, \mu, \tau, \Gamma, \mathbf{CV}, w, \mathbf{ASF}, \delta)$; // Optimize model space
    $\mathrm{P}_{new} \leftarrow \mathrm{P}_{new} \cup \{X_{new}\}$
  break if $e \geq E$;
  $\widehat{\mathrm{F}}^i_{new} \leftarrow \widehat{f}^i_{t+1}(\mathrm{P}_{new}), \forall i \in \{1, \ldots, M\}$// predicted
  $\widehat{\mathrm{G}}^j_{new} \leftarrow \widehat{g}^j_{t+1}(\mathrm{P}_{new}), \forall j \in \{1, \ldots, J\}$// predicted
  $\delta \leftarrow \textsc{Update\_TrustRegion}(\mathrm{F}_{t+1}, \widehat{\mathrm{F}}_{new}, \mathrm{G}_{t+1}, \widehat{\mathrm{G}}_{new}, \delta)$;
  $\mathrm{P}_{new} \leftarrow \emptyset$;
  $t \leftarrow t + 1$;
return $\mathrm{P}_T \leftarrow$ filter the best solutions from P_{t+1}

Thereafter, these solutions are evaluated exactly (high-fidelity) and metamodels are constructed for all M objectives ($\widehat{f}_i(x); i = 1, \ldots, M$) and J constraints ($\widehat{g}_j(x); j = 1, \ldots, J$). Then, a multi-objective evolutionary algorithm (NSGA-II is used here) procedure is run for τ generations starting with μ initial random solutions in model space. The NSGA-II algorithm returns $\min(\mu, E - e)$ solutions where e is the current number of exact solution evaluations. The solutions are then evaluated using high-fidelity simulation and included in the archive (line 8). Then, new metamodels are then build from scratch and the process is repeated until termination. The trust radii are updated after each NSGA-II run, for new and old points according to the update rules discussed before. The major steps of this method are outlined in Algorithm 5.

4.2 Efficient Non-dominated Sorting Algorithm

Non-dominated sorting (NDS) has emerged as a critical component for practical multi-objective (mostly two or three) optimization problems (MOPs). In contrast to single objective optimization where we try to find the best possible solution, the desired result of an MOP is typically a set of Pareto-optimal solutions that reflect the trade-offs among different objectives. The search space grows exponentially with the number of objectives and the size of sampling points should also grow accordingly. So it is important to devise a fast practical algorithm for large scale many-objective problems. Due to its high 'worst case' complexity ($O(mn^2)$ where n is the number of solutions and m is the number of objectives), repeated use of NDS algorithm is a computational bottleneck for multi- and many-objective evolutionary algorithms (MOEAs). Other key operations such as crossover, mutation or tournament selection are typically fast (linear time) compared to NDS algorithm. Stated another way,

Algorithm 5: Hypervolume based Trust Region Method

Input : Objectives: $[f_1, \ldots, f_m]^T$, Constraints: $[g_1, \ldots, g_J]^T$, n (variables), ρ (sample size), E (maximum high-fidelity solution evaluations), NSGA-II (multi-objective evolutionary algorithm) with population size μ, generations for model optimization τ and other parameters Γ, **CV** (constrained violation function), Trust region parameters– $\delta_{init}, \Delta_{max}, c_1, c_2, r_1, r_2$

Output: Solution set P_T

$t, e \leftarrow 0$;
$\mathrm{P}_t, \mathrm{F}_t, \mathrm{G}_t \leftarrow \emptyset$;
$\mathrm{P}_{new} \leftarrow \mathrm{LHS}(\rho, n)$// `Initial solutions`
$\delta^i \leftarrow \delta_{init}\ \forall i \in \{1, \ldots, \rho\}$;
while *True* **do**
  $\mathrm{F}^i_{new} \leftarrow f_i(\mathrm{P}_{new}), \forall i \in \{1, \ldots, M\}$// `eval obj.`
  $\mathrm{G}^j_{new} \leftarrow g_j(\mathrm{P}_{new}), \forall j \in \{1, \ldots, J\}$// `eval constr.`
  if $t > 0$ **then**
    $\widehat{\mathrm{F}}^i_{new} \leftarrow \widehat{f}^i_t(\mathrm{P}_{new}), \forall i \in \{1, \ldots, M\}$// `predicted`
    $\widehat{\mathrm{G}}^j_{new} \leftarrow \widehat{g}^j_t(\mathrm{P}_{new}), \forall j \in \{1, \ldots, J\}$// `predicted`
    $\delta \leftarrow$ UPDATE_TRUSTREGION$(\mathrm{F}_t, \widehat{\mathrm{F}}_{new}, \mathrm{G}_t, \widehat{\mathrm{G}}_{new}, \delta)$
  $\mathrm{P}_{t+1}, \mathrm{F}_{t+1}, \mathrm{G}_{t+1} \leftarrow (\mathrm{P}_t \cup \mathrm{P}_{new}), (\mathrm{F}_t \cup \mathrm{F}_{new})$ and $(\mathrm{G}_t \cup \mathrm{G}_{new})$;
  $e \leftarrow e + |\mathrm{P}_{new}|$;
  break if $e \geq E$;
  $\widehat{f}^i_{t+1} \leftarrow$ METAMODEL$(\mathrm{F}^i_{t+1}), \forall i \in \{1, \ldots, M\}$// `metamodel obj.`
  $\widehat{g}^j_{t+1} \leftarrow$ METAMODEL$(\mathrm{G}^j_{t+1}), \forall j \in \{1, \ldots, J\}$// `metamodel constrt.`
  $\mathrm{P}_{new} \leftarrow$ NSGA-II$(\widehat{f}_{t+1}, \widehat{g}_{t+1}, \mu, \tau, \Gamma, E - e, \mathbf{CV}, \delta)$; // `Optimize model space`
  $t \leftarrow t + 1$;
return $\mathrm{P}_T \leftarrow$ filter the best solutions from P_{t+1}

speeding up non-dominated sorting will allow MOEAs to run with larger populations, more generations, and more objectives leading to better solutions for most problem domains.

We primarily use the terminology from the MOEA community; that is, we usually refer to points as solutions, dimensions as objectives, and a set of solutions as a population. We use rank, layer or front number interchangeably. The problem input is a set of m-objective solutions of size n, $P = \{p^i \mid 1 \leq i \leq n\}$, where the value of solution p^i in the j-th objective is denoted as p^i_j for $1 \leq i \leq n$ and $1 \leq j \leq m$. We refer to Chapter 1 for Pareto-dominance

relation. We start by defining the concept of solution domination.

Definition 2 (Solution Domination)**.** *We say that a solution p dominates another solution p', denoted by $p \succ p'$, if p is better than or equal to p' in all objectives ($p_j \leq p'_j$ for $1 \leq j \leq m$) and p is strictly better in any objective ($\exists j\ p_j < p'_j$). Otherwise, $p \not\succ p'$. If p cannot dominate p' and p' cannot dominate p, then they are mutually non-dominated.*

We define $R(p)$, the layer number, front number or rank of solution $p \in P$, using solution domination. Intuitively, $\max(R(p))$ denotes the longest path in the directed graph defined by the $\succ$ relation on solutions of P. Rank 1 solutions are also denoted as the non-dominated front, maximal layer or Pareto front of set P.

Definition 3 (Non-dominated Sorting/Layers of Maxima)**.** *Given a set P of m-objective solutions where $|P| = n$, we define $R(p)$, the rank of each solution $p \in P$, as follows.*

- $R(p) = 1$ *if for* $\forall p' \in P$, $p' \not\succ p$. *Otherwise,*
- $R(p) = 1 + \max\{R(p') | p' \succ p\}$.

Here this relation holds– $\forall p' \in P,\ p' \succ p \Rightarrow R(p') \leq R(p)$. In other words, if solutions are not dominated by any other solution, they have rank 1. Otherwise, the rank of a solution p is one plus the rank of the largest ranked solution p' that dominates p.

Most MOEAs generate a new population of solutions from the current population where only the "best" solutions of the current population contribute to the next population. These MOEAs such as NSGA, NSGA2, SPEA2, PAES, PESA, EPCS, MOEA/DD, RVEA [96, 2, 97, 98, 99, 100, 101, 102, 103] use NDS to identify the "best" solutions of the current generation. How they define the best solution differs by algorithm. Some require

full rank information; others require only the top rank. Apart from the area of multi-objective optimization, non-dominated sorting has been studied in many application areas ranging from gene selection, ranking to data clustering even in database skyline queries [104, 105, 106, 107, 108]. In the next section, we discuss previous state-of-the-art solutions to this problem and mention our contribution to this field.

4.2.1 Related Work

The non-dominated sorting problem is completely solved when $m = 2$ or 3 with a worst case time complexity of $\Theta(n \log n)$ [109, 110]. Srinivas and Deb provided the first non-dominated sorting algorithm in their MOEA named NSGA which ran in $O(mn^3)$ time and requires $O(n)$ space [96]. This was improved to $O(mn^2)$ time at the cost of using $O(n^2)$ space in the NSGA-II algorithm [2]. Several methods improved upon NSGA-II by eliminating some unnecessary comparisons by inferring some dominance relationships using the results of already completed comparisons and intelligently choosing which solutions to compare next. These include Tang *et al.*'s arena principle non-dominated sorting algorithm [111], Clymont and Keedwell's deductive sort [112], and Wang and Yao's corner sort [113], Fang *et al.*'s [114] domination tree– all of which run in $O(mn^2)$ time and use $O(n)$ space, in the worst case.

An alternative approach is to use a divide-and-conquer (D&C), often referred to as Jensen's sort [115, 116, 117, 118, 119, 120]. For $m > 2$, D&C requires $O(n \log^{m-1} n)$ time which is good for small m but quickly becomes intractable for even moderate m.

Zhang *et al.* identified the following key issue with almost every existing non-dominated sorting algorithm [121]: they work by computing each front in order. Zhang *et al.* presented an improved algorithm, ENS, that overcomes this issue by first sorting all the solutions using a single objective. Sorting requires $O(n \log n)$ time. They then process the solutions in this

sorted order comparing each solution against the solutions located before its position in the sorted list to determine its exact front. Despite this clever optimization for ENS and the ability to eliminate half of the comparisons in the worst case, ENS still has a worst-case time complexity of $O(mn^2)$ and a space complexity of $O(n)$.

Gustavsson *et al.* [122] introduced a non-domination based data structure, a variant of a bucket k-dimensional tree (k-d tree) to reduce the running time of comparing the whole front. This method adopted ENS-BS structure, and they limit the depth of k-d tree and used bucket of size three. It is useful mostly for large dimensions and single front where most of the objectives are not correlated. Another tree based method with the same purpose is proposed in [123].

Some approaches [124, 125, 126] deal with the problem of dynamic or online update of the non-dominated set. These algorithms require more time than static NDS algorithms since the addition or removal of one point may disrupt the non-domination structure. We do not consider the online NDS problem.

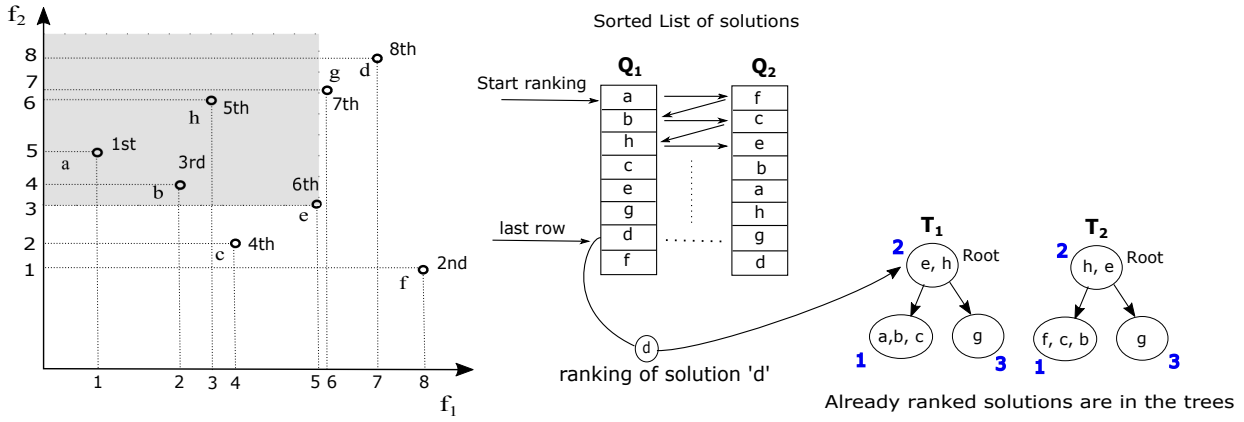

Figure 4.2: Basic steps of Best Order Sort (BOS) algorithm are shown for two-dimensional case. Q_1 and Q_2 are the sorted lists of solutions according to objective 1 and 2 respectively. The algorithm goes over Q_1 and Q_2 from left to right and top to bottom and rank the extracted solutions only by comparing the solutions above (better in the corresponding objective) it. An improvement over BOS would be to keep the ranked solution in trees T_1 and T_2 to facilitate further reduction of solution comparisons.

4.2.2 Proposed Approach: Best Order Sort

Our input P is a set of solutions $\{s_j \in \mathbb{R}^m \mid 1 \leq j \leq n\}$ where s_j^i is the value of solution j in objective i. Our goal is to compute the rank of all solutions in P. We assume without loss of generality that solutions are unique but may have identical values in some objectives.

We divide the problem of ranking solutions into two phases, ordering and ranking. We first order all the solutions in m objectives. We then extract the minimum unprocessed solution from each of our m ordered objectives and rank that solution if it has not already been ranked until all solutions are ranked. This basic approach builds upon Zhang *et al.*'s ENS method [121]. We improve upon ENS by ordering each solution in m objectives. Compared to ENS, we spend more time in ordering but hopefully spend less time in ranking because we compare each solution against fewer other solutions. Basic steps of the proposed method are presented in Figure 4.2.

4.2.2.1 Ordering Phase

In the ordering phase, we order the solutions in P based on each objective i for $1 \leq i \leq m$ using an ordering function $<_i$ which we define below. We first define the lexical order of solutions in P, denoted by $<_\ell$, using objectives 1 to m in order as follows. For any two solutions s_u and s_v in P, let k be the smallest integer such that $s_u^k \neq s_v^k$. If $s_u^k < s_v^k$, then $s_u <_\ell s_v$; else $s_v <_\ell s_u$. It follows that $<_\ell$ defines a total order on the solutions in P. Then, for $1 \leq i \leq m$ and any two solutions s_u and s_v, $s_u <_i s_v$ if $(s_u^i < s_v^i)$ or $((s_u^i = s_v^i)$ and $(s_u <_\ell s_v))$; otherwise $s_v <_i s_u$. That is, we first order s_u and s_v by their values s_u^i and s_v^i. If that does not resolve their order, we order s_u and s_v by their lexical order.

For each $1 \leq i \leq m$, we store the solutions P ordered by $<_i$ in an ordering data structure Q_i that supports two operations: (i) construct Q_i given P and $<_i$ and (ii) extract minimum

which will be used during the ranking phase. We consider two standard data structures for Q_i. The first is a sorted linked list or sorted array which supports construction in $O(n \log n)$ time and extract minimum in $O(1)$ time. The second is a binary heap which supports construction in $O(n)$ time and extract minimum in $O(\log n)$ time. For simplicity, it is easier to think about the sorted linked list or sorted array, but the binary heap may be faster, especially when we process some but not all the solutions in Q_i during the ranking phase.

Algorithm 6: BEST ORDER SORT

Input : Population $P = \{s_j \in \mathbb{R}^m \mid 1 \leq j \leq n\}$
Output: Ranking R of solutions in P
// Ordering phase
$R \leftarrow \{\}$ // no solutions ranked yet
$Q_1 \leftarrow$ sort P using lexical order ;
Initialize $L_i = \emptyset \; \forall i = 1$ to n// no solutions ranked
$Q_i \leftarrow Order(P, i), \; \forall i = 1$ to n// lexicographic sort by i-th objective
for $j = 1$ **to** n **do**
    for $i = 1$ **to** m **do**
        Put $q_i \leftarrow ExtractMin(Q_i)$ in the sorted order in Q_i
        if *all n solutions are extracted once* **then**
            $index \leftarrow i$
            break out of both loops
$objSeq \leftarrow$ Find order of objectives. Use the reverse order till depth $index$ from Q, other objectives are randomly ordered// global
$C(P) \leftarrow m$// it counts # obj. to compare, global
// Ranking phase
while $|R| < n$ **do**
    for $i = 1$ **to** m **do**
        $q_i \leftarrow ExtractTop(Q_i)$ // Q_i is already sorted till $index$
        $C(q_i) \leftarrow C(q_i) - 1$
        if $q_i \notin R$ **then**
            rank$(q_i) \leftarrow$ INSERT(P, $objSeq$, L_i, C, q_i, max (rank(R)))
            $R \leftarrow R \cup \{\text{rank}(q_i)\}$
        else
            INSERTINTORANK$(L_i, q_i, \text{rank}(q_i))$
return R

4.2.2.2 Ranking Phase

We perform ranking in rounds. In a round, for objective $1 \leq i \leq m$, we process q_i which is the minimum unprocessed solution from Q_i adding q_i to a ranking data structure L_i which organizes the processed solutions from Q_i to facilitate fast ranking. There are two possibilities for how we process q_i. If q_i is unranked ($q_i \notin R$), then we simultaneously rank q_i and insert q_i into L_i and R. If q_i is already ranked ($q_i \in R$), then we just insert q_i into L_i. In both cases, we do not modify the ranks of already ranked items.

We first consider the case where q_i is unranked ($q_i \notin R$). The key observation (due to Zhang *et al.* [121]) is that no solution $s \in P \setminus L_i$ can dominate q_i because $q_i <_i s$ which means either $q_i^i < s^i$ or $q_i <_\ell s$. Thus, we only need to compute the rank of q_i against the solutions in L_i. The exact details of how we compute this rank depends on the details of L_i. We assume that Insert(L_i,q_i) will insert q_i into L_i while determining and returning q_i's rank.

We next consider the case where q_i is already ranked ($q_i \in R$) but was previously unprocessed in Q_i. In this case, we assume that InsertIntoRank(L_i, q_i, rank(q_i)) will correctly insert q_i into L_i.

The algorithm can safely terminate if all solutions are ranked before n rounds, so we only continue if there are unranked solutions ($|R| < n$).

We now describe a basic implementation of the ranking data structure L using arrays of linked lists. Observe that solutions in L can be partitioned into a list of solutions with the same rank. Let L^k be the solutions with rank k, and r be the maximum rank in L. We have that $L = L^1 \sqcup L^2 \sqcup \ldots \sqcup L^r$ where $\sqcup$ denotes disjoint union. So, L^k can be indexed by k using an array, and each L^k can be a linked list of solutions with rank k.

To implement InsertIntoRank(L, q,rank(q)), we simply add the new solution q into $L^{\text{rank}(q)}$. One can verify that Algorithm 6 always has rank$(q) \leq r+1$. If rank$(q) = r+1$, we create a new list L^{r+1} which will be initialized to hold just solution q.

To implement Insert(L, q, SP), we find the rank of q and then insert into $L^{\text{rank}(q)}$. To find the rank of q, we use the following domination check (DC) primitive. Given $1 \leq j \leq r$ and q, $DC(j, q)$ is true if any solution in L^j dominates q; otherwise $DC(j, q)$ is false. We then check $DC(j, q)$ starting with $j = 1$ and incrementing j until $DC(j, q)$ becomes false. Then $rank(q)$ is this value of j.

Algorithm 7: InsertIntoRank

Input : List of solutions L, q (solution to be inserted), r (rank of q)
Output: -

```
1 Insert q at the front of the list L^r
```

Algorithm 8: Insert

Input : P (Population), $objSeq$ (Objective Sequence), L (List of solutions), C (Counter), q (the solution to be ranked), RC (Total number of ranks found so far)
Output: Rank r of solution q

```
1 for k = 1 to RC do // for all discovered ranks
2     dominated ← false// initialize
3     for t ∈ L^k do // for all solutions in L^k
4         dominated ← DominationCheck(P, C, objSeq, s, t)// domination check
5         if dominated then // if dominated
6             break// go to next rank
7     if dominated == false then // not dominated by any solution of rank k
8         return k
9 return (k+1)
```

Algorithm 9: DominationCheck or DC

Input : P (Population), $objSeq$ (Objective Sequence), C (Counter), Solution t, Solution q (checks if t dominates q)

Output: **true** if t dominates s, **false** otherwise

```
1  equal ← true// checks if two values are equal
2  if C(t)==0 then
3  |  Check whether q is duplicate of t. If duplicate return false, else return true;
4  for j = 1 to C(t) do // for first C(t) objectives
5  |  if P[t][objSeq[t][i]] > P[q][objSeq[t][i]] then
6  |  |  return false// t cannot dominate q
7  |  else if P[t][objSeq[t][i]] == P[q][objSeq[t][i]] then
8  |  |  equal ← false// not equal
9  if equal == true then
10 |  return false// two values are equal
11 else
12 |  return true// t dominates q
```

4.2.3 Results of Best Order Sort

We compared the proposed algorithm with four different algorithms- fast non-dominated sort [2], deductive sort [112], corner sort [113] and divide-and-conquer algorithm [117]. These algorithms are compared in cloud dataset, fixed front dataset and MOEA dataset obtained from multi-objective evolutionary algorithm (MOEA). Cloud dataset is a uniform random data generated using random uniform distribution. Fixed front data is the dataset where number of fronts is controlled. We have used the procedure described in [113] for generating cloud and fixed front dataset. We vary size of population N from 500 to 10,000 with 500 increment in cloud dataset. In another test, number of objectives are varied from 2 to 20 to evaluate performance with population size 10,000. For fixed front dataset, number of front is varied from 1 to 10 where number of solution is kept 10,000 with objectives 5, 10, 15 and 20. MOEA dataset is obtained by running 200 generations of NSGA-II algorithm in DTLZ1 and DTLZ2 [85], WFG1 and WFG2 [127] problems with 5, 10, 15 and 20 objectives. In these

cases, all the parameter values are kept as standard ones. For example, simulated binary crossover with polynomial mutation are employed with probabilities 0.80 and (1/number of variables) respectively. Each test case is repeated in 30 different datasets. All the algorithms are optimized and implemented in Java Development Kit 1.8 update 65 and run in Intel core-i7 with 64 bit Windows 7 machine.

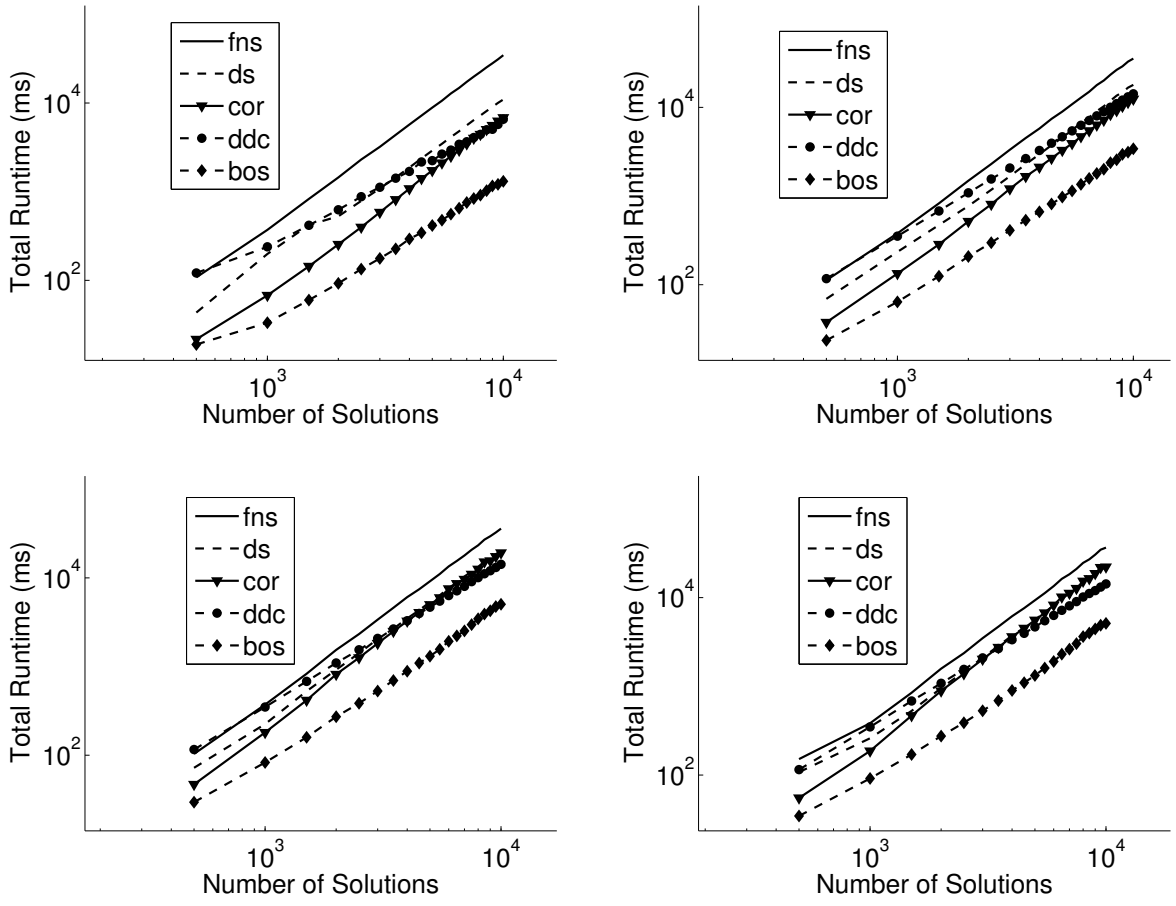

Figure 4.3: Figure describes running time (in milliseconds) with increasing population size for cloud dataset with objectives 5, 10, 15 and 20 respectively. Results for fast non-dominated sort (fns), deductive sort (ds), corner sort (cor), divide-and-corner sort (ddc) and best order sort (bos) is shown.

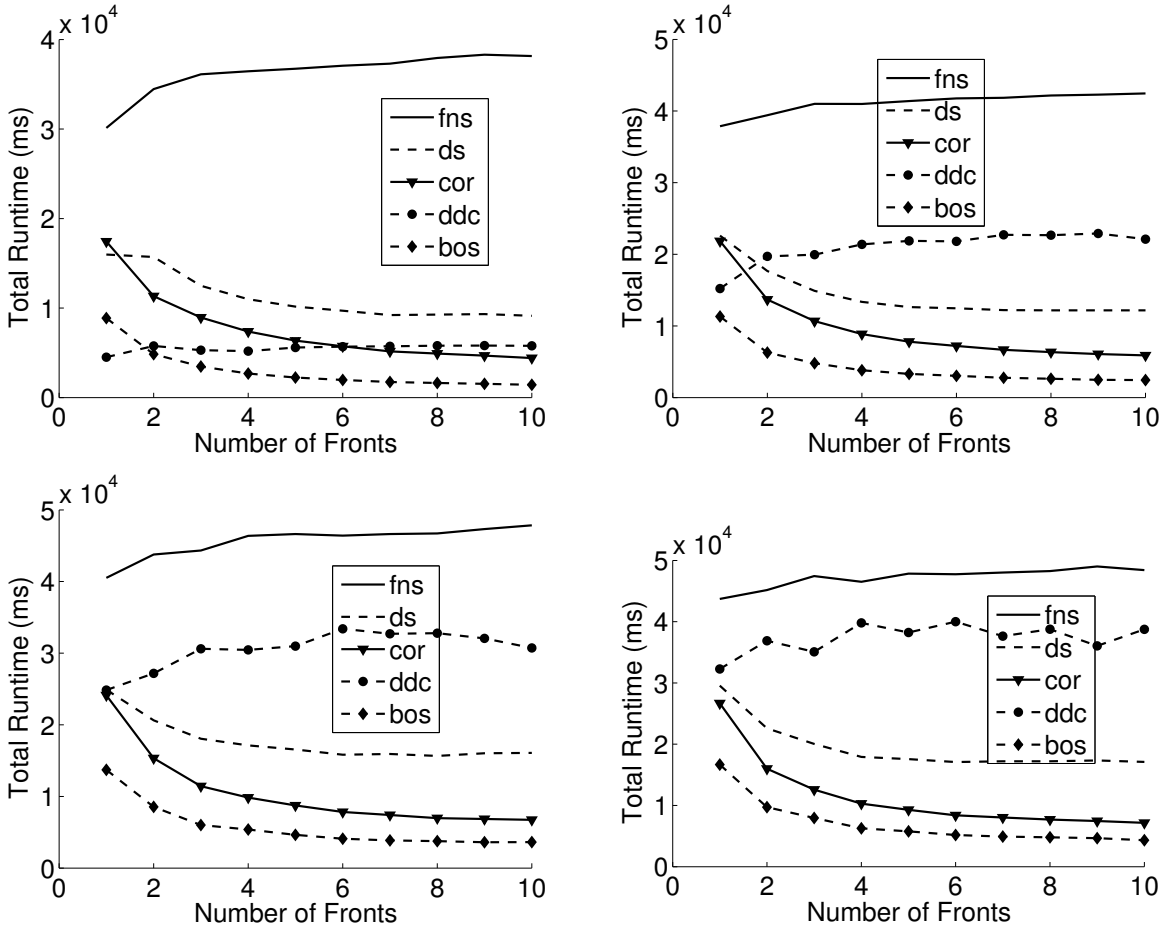

Figure 4.4: Figure describes running time (in milliseconds) with increasing number of fronts for fixed front dataset with objectives 5, 10, 15 and 20 respectively. Results for fast non-dominated sort (fns), deductive sort (ds), corner sort (cor), divide-and-corner sort (ddc) and best order sort (bos) is shown.

4.2.4 Discussions on Best Order Sort Results

The results describe the average case behavior of the algorithms in three different cases. Figure 4.3 shows that with increased number of objectives, number of comparisons and running time increases for deductive sort, corner sort, divide-and-conquer sort and best order sort. Fast non-dominated sort performs worst in two objectives compare to other number of objectives. This is because, number of fronts is very high in two objective random data and fast non-dominated sort takes most of the time just for saving dominated solutions

Table 4.1: Total number of comparisons (#cmp) and running time (in milliseconds) for DTLZ1, DTLZ2, WFG1 and WFG2 problems in 5, 10, 15 and 20 objectives.

Test Problem	**Obj.**	FNS		DS		COR		DDC		BOS	
		#cmp	time(ms)	#cmp	time(ms)	#cmp	time(ms)	#cmp	time(ms)	#cmp	time(ms)
DTLZ1	5	3.25e+08	1.03e+03	1.02e+08	4.48e+02	7.12e+07	2.18e+02	9.41e+06	3.79e+02	**7.95e+06**	**1.06e+02**
	10	5.13e+08	1.21e+03	2.74e+08	7.33e+02	1.71e+08	4.05e+02	**1.44e+07**	5.03e+02	2.03e+07	**2.53e+02**
	15	7.09e+08	1.41e+03	4.23e+08	1.02e+03	2.67e+08	5.52e+02	**1.50e+07**	5.42e+02	2.86e+07	**3.87e+02**
	20	8.98e+08	1.59e+03	5.67e+08	1.21e+03	3.55e+08	6.56e+02	**1.56e+07**	5.55e+02	3.51e+07	**4.72e+02**
DTLZ2	5	2.97e+08	8.59e+02	1.24e+08	4.77e+02	8.22e+07	2.58e+02	**9.52e+06**	3.52e+02	1.07e+07	**1.17e+02**
	10	4.30e+08	1.11e+03	2.31e+08	6.82e+02	1.59e+08	4.36e+02	**1.55e+07**	5.46e+02	1.80e+07	**2.35e+02**
	15	5.58e+08	1.27e+03	3.31e+08	8.37e+02	2.20e+08	5.41e+02	**1.63e+07**	5.84e+02	2.20e+07	**3.15e+02**
	20	6.95e+08	1.40e+03	4.34e+08	1.02e+03	2.81e+08	6.36e+02	**1.65e+07**	5.97e+02	2.49e+07	**3.73e+02**
WFG1	5	2.67e+08	7.99e+02	1.12e+08	4.38e+02	6.59e+07	2.44e+02	**9.89e+06**	3.53e+02	1.11e+07	**1.18e+02**
	10	2.95e+08	9.30e+02	1.47e+08	5.26e+02	1.03e+08	3.64e+02	**2.19e+07**	7.74e+02	2.09e+07	**2.63e+02**
	15	3.26e+08	9.65e+02	1.75e+08	5.85e+02	1.27e+08	4.47e+02	**2.41e+07**	8.74e+02	2.58e+07	**3.64e+02**
	20	3.57e+08	1.07e+03	2.00e+08	6.34e+02	1.47e+08	5.06e+02	**2.46e+07**	8.99e+02	2.91e+07	**4.50e+02**
WFG2	5	3.00e+08	9.18e+02	1.10e+08	4.69e+02	6.68e+07	2.06e+02	**9.64e+06**	3.55e+02	1.11e+07	**1.25e+02**
	10	5.56e+08	1.16e+03	2.80e+08	7.19e+02	1.78e+08	3.27e+02	**1.53e+07**	5.30e+02	3.02e+07	**3.28e+02**
	15	8.98e+08	1.52e+03	5.03e+08	1.06e+03	3.26e+08	4.71e+02	**1.58e+07**	5.53e+02	5.60e+07	**5.36e+02**
	20	1.26e+09	1.75e+03	7.46e+08	1.47e+03	4.88e+08	6.30e+02	**1.53e+07**	**5.40e+02**	8.30e+07	7.22e+02

in a list of size $O(N^2)$. Best order sort performs the best followed by divide-and-conquer, corner sort and deductive sort. Log-based plots in Figure 4.3 show that fast non-dominated has highest and best order sort has the lowest order in terms of number of comparisons and running time in objectives 5, 10, 15 and 20. Corner sort performs better than deductive sort in terms of comparisons in most of the cases but the running time performance deteriorates with increasing number of objectives. Divide-and-conquer algorithm performs better than all sequential type algorithms except the proposed method. The number of comparisons and runtime decreases with the increasing number of fronts (Figure 4.4) except fast non-dominated sort and divide-and-conquer sort. In those two cases, running time and number of comparisons increases with the increased number of fronts. Best order sort performs better than all other algorithms followed by corner sort and deductive sort respectively. In MOEAs, divide-and-conquer algorithm has fewest number of comparisons in most of the cases but running time is slightly worse than best order sort. Best order sort becomes second in terms of comparisons followed by corner sort and deductive sort. Divide-and-conquer algorithm has advantage with small size of data in MOEAs. Best order sort outperforms all the comparing algorithms when size of population and number of objectives are increased.

4.3 Summary of the Chapter

In this chapter, we have presented an adaptive trust region concept for multi-objective optimization for low budget problems. We have proposed two performance indicators, based on scalarization and hypervolume, to adapt the selected trust regions. A constraint handling scheme is presented in order to handle the trust region adaptation for constraint violations. The results with trust region methods are presented in Chapter 6. We have also presented an efficient approach to non-dominated sorting for efficient low-fidelity optimization. Compare to other non-dominated sorting algorithms, this approach takes advantage of the faster scalar sorting methods to reduce the number of solution comparisons as presented in the results. The proposed approach is also somewhat suitable for parallel implementation since each sorted list can be operated independently to find ranks.

Chapter 5

Ensemble Algorithm

5.1 A Brief Overview of Metamodeling Frameworks

In this section we provide a brief discussion on metamodeling frameworks proposed in this thesis. We demonstrate the acquisition functions or metamodeling functions for each framework. Then we propose a performance metric for selecting the best framework.

5.1.1 Frameworks M1-1 and M2-1

The metamodeling algorithm for M1-1 and M2-1 starts with an archive of initial population ($\mathcal{A}_0$ of size N_0) created using the Latin hypercube sampling (LHS) method on the entire search space. Each objective function ($f_i(\mathbf{x})$ for $i = 1, \ldots, M$), is first normalized to obtain a normalized function $\underline{f}_i(\mathbf{x})$ using high-fidelity evaluation of initial archive members, so that the minimum and maximum values of $\underline{f}_i(\mathbf{x})$ evaluations are zero and one, respectively. For M1-1, each constraint function ($g_j(\mathbf{x})$, for $j = 1, \ldots, J$) is first normalized to obtain a normalized constraint function ($\underline{g}_j(\mathbf{x})$) using standard approaches [128], and then metamodeled separately to obtain an approximation function ($\tilde{\underline{g}}_j(\mathbf{x})$). For M2-1, a single aggregated constraint violation function (ACV($\mathbf{x}$)) is constructed using the normalized constraint functions, as follows:

$$\text{ACV}(\mathbf{x}) = \begin{cases} \sum_{j=1}^{J} \underline{g}_j(\mathbf{x}), & \text{if } \mathbf{x} \text{ is feasible,} \\ \sum_{j=1}^{J} \langle \underline{g}_j(\mathbf{x}) \rangle, & \text{Otherwise,} \end{cases} \tag{5.1}$$

where the bracket operator $\langle\alpha\rangle$ gives value α if $\alpha > 0$ and gives zero, otherwise. In M2-1, the constraint violation function is then metamodeled to obtain $\widetilde{\text{ACV}}(\mathbf{x})$.

We then optimize a scalarization function using metamodeled objective functions $\underline{\tilde{f}}_i(\mathbf{x})$ for $i = 1, \ldots, M$ and using all J constraints $\underline{g}_j(\mathbf{x})$ (for M1-1) or $\widetilde{\text{ACV}}(\mathbf{x})$ (for M2-1) to find a single in-fill point using a single-objective evolutionary optimization algorithm (real-coded genetic algorithm (RGA) [3] used here).

$$\text{ASF}_{12}(\mathbf{x}, \mathbf{z}) = \max_{j=1}^{M} \left(\underline{\tilde{f}}_j(\mathbf{x}) - z_j \right), \tag{5.2}$$

where the vector $\mathbf{z}$ is one of the Das and Dennis's [77] approach on the unit simplex on the M-dimensional hyper-space. Thus, for H different $\mathbf{z}$ vectors, H different ASF_{12} functions are formed and optimized one after the other. The best solution for each problem constitutes one in-fill point and is sent for a high-fidelity evaluation and process is continued until all function evaluations are utilized.

5.1.2 Frameworks M1-2 and M2-2

These two frameworks optimize metamodeled normalized objective functions $\underline{\tilde{f}}_i(\mathbf{x})$ for $i = 1, \ldots, M$ using a multi-objective evolutionary algorithm to obtain a set of non-dominated solutions in each epoch. Constraints for M1-2 and M2-2 are formulated as discussed for frameworks M1-1 and M2-1, respectively, as above. In this paper, we use NSGA-II procedure [37] for two-objective problems, and NSGA-III [129] for three or more objective problems to get H in-fill solutions. All H solutions are then evaluated using high-fidelity models and are included in the archive for another round of metamodel construction and optimization for the next epoch.

5.1.3 Frameworks M3-1 and M4-1

In these two methods, instead of metamodeling the normalized objective functions $\underline{f}_i(\mathbf{x})$ for $i = 1, \ldots, M$, we first aggregate them to form the following ASF_{34} function:

$$\text{ASF}_{34}(\mathbf{x}, \mathbf{z}) = \max_{j=1}^{M} \left(\underline{f}_j(\mathbf{x}) - z_j \right), \tag{5.3}$$

where $\mathbf{z}$ is defined as before. The $\text{ASF}_{34}(\mathbf{x}, \mathbf{z})$ for each of a total H predefined $\mathbf{z}$-vectors is now metamodeled. In M3-1, each normalized constraint function is metamodeled separately as in M1-1. In M4-1, the aggregated constraint violation function, as described in Equation 5.1 as in M2-1, is constructed and metamodeled. The single objective RGA with $\text{ASF}_{34}(\mathbf{x}, \mathbf{z})$ objective function and constraint as described above is used to solve each optimization problem for each $\mathbf{z}$-vector. Thus, both M3-1 and M4-1 are applied H times with a systematic variation of $\mathbf{z}$-vectors (Das and Dennis's [77] approach used here) to obtained H in-fill points at each epoch.

5.1.4 Frameworks M3-2 and M4-2

In these two frameworks, we build metamodels for an effective constraint function (ECV) in the same way as in M3-1 and M4-1, respectively:

$$\text{ECV}(\mathbf{x}) = \begin{cases} \sum_{j=1}^{J} \langle \widetilde{\underline{g}}_j(\mathbf{x}) \rangle, & \text{for M3-2,} \\ \langle \widetilde{\text{ACV}}(\mathbf{x}) \rangle, & \text{for M4-2.} \end{cases} \tag{5.4}$$

We then apply a *multi-modal* single-objective evolutionary algorithm to find H in-fill points simultaneously. The proposed multi-modal RGA (or, MM-RGA) starts with a random population of size N for this purpose. In each generation, the population (P_t) is modified to a new

population (P_{t+1}) by using selection, recombination and mutation operators. The selection operator emphasizes multiple diverse solutions as follows. First, a *fitness* is assigned to each population member $\mathbf{x}$ by computing $\widetilde{\text{ASF}}_{34}(\mathbf{x}, \mathbf{z})$ for all H, $\mathbf{z}$-vectors and then assigning the smallest value as fitness. Then, we apply the binary tournament selection to choose a parent using the following selection function:

$$\text{SF}(\mathbf{x}, \mathbf{z}) = \begin{cases} \widetilde{\text{ASF}}_{34}(\mathbf{x}, \mathbf{z}), & \text{if } \mathbf{x} \text{ is feasible,} \\ \widetilde{\text{ASF}}^{\max}_{34,\mathbf{z}} + \text{ECV}(\mathbf{x}), & \text{otherwise,} \end{cases} \tag{5.5}$$

where $\widetilde{\text{ASF}}^{\max}_{34,\mathbf{z}}$ is the maximum $\widetilde{\text{ASF}}_{34}(\mathbf{x}, \mathbf{z})$ value of all feasible population members of MM-RGA. The above selection function has the following effects. If two solutions are feasible based on ECV($\mathbf{x}$), SF($\mathbf{x}$,$\mathbf{z}$) is used to select the winner. If one is feasible and the other is infeasible, the former is chosen by the use of $\widetilde{\text{ASF}}^{\max}_{34,\mathbf{z}}$ for infeasible members in the SF expression. For two infeasible members, the one with minimum ECV($\mathbf{x}$) is chosen.

After N offspring population members are thus created, we merge the population to form a combined population of $2N$ members. The best solution to each $\mathbf{z}$-vector is then copied to P_{t+1}. In the event of a duplicate, the second best solution for the $\mathbf{z}$-vector is chosen. If H is smaller than N, then the process is repeated to select a second population member for as many $\mathbf{z}$-vectors as possible. Thus, at the end of the MM-RGA procedure, exactly H in-fill solutions are obtained. Thus, for M4-2, one metamodel $\widetilde{\text{ACV}}(\mathbf{x})$ for all constraints and H metamodels $\widetilde{\text{ASF}}_{34}(\mathbf{x}, \mathbf{z})$ for H $\mathbf{z}$-vectors are required to be formed. In M3-2, J metamodels $\widetilde{g}_j(\mathbf{x})$ for all J constraint functions are needed.

5.1.5 Framework M5

The focus of M5 is to use a generative multi-objective optimization approach in which a single Pareto-optimal solution is found at a time for a $\mathbf{z}$-vector by using a combined *selection* function involving all objective and constraint functions together, as used in a specific generative EMO algorithm. The following *selection* function is created:

$$\mathcal{S}_5(\mathbf{x},\mathbf{z}) = \begin{cases} \mathrm{ASF}_{34}(\mathbf{x},\mathbf{z}), & \text{if } \mathbf{x} \text{ is feasible,} \\ \mathrm{ASF}_{34}^{\max}(\mathbf{x},\mathbf{z}) + \langle \mathrm{ACV}(\mathbf{x}) \rangle, & \text{otherwise.} \end{cases} \tag{5.6}$$

Here, the parameter $\mathrm{ASF}_{34}^{\max}(\mathbf{x},\mathbf{z})$ is the worst ASF_{34} function value (described in Equation 5.3) of all feasible solutions from the archive. Two functions – selection function $\mathcal{S}_5(\mathbf{x}, \mathbf{z})$ and constraint violation function $\mathrm{ACV}(\mathbf{x})$ (Equation 5.1) – are now metamodeled to obtain $\widetilde{\mathcal{S}}_5(\mathbf{x},\mathbf{z})$ and $\widetilde{\mathrm{ACV}}(\mathbf{x})$ for the RGA to optimize and find one in-fill solution for each $\mathbf{z}$-vector. Thus, H objective metamodels and one constraint metamodel are required for M5 in each epoch.

5.1.6 Framework M6

Framework M6 constructs a single metamodel in each epoch by combining all M objectives and J constraints together. A multi-modal function having each optimum corresponding to a distinct Pareto-optimal solution is formed for this purpose:

$$\mathrm{ASF}_6(\mathbf{x}) = \min_{\mathbf{z}\in} \max_{i=1}^{M} \left(\underline{f}_i(\mathbf{x}) - z_i \right). \tag{5.7}$$

Then, the following selection function is constructed:

$$\mathcal{S}_6(\mathbf{x}) = \begin{cases} \text{ASF}_6(\mathbf{x}), & \text{if } \mathbf{x} \text{ is feasible,} \\ \text{ASF}_{6,\max} + \text{CV}(\mathbf{x}), & \text{otherwise,} \end{cases} \tag{5.8}$$

where $\text{ASF}_{6,\max}$ is the maximum ASF_6 value of all feasible archive members. For each archive member $\mathbf{x}$, $\mathcal{S}_6(\mathbf{x})$ is first computed. $\text{CV}(\mathbf{x})$ is same as $\text{ACV}(\mathbf{x})$, except that for a feasible $\mathbf{x}$, CV is zero. Due to the complexity involved in the $\mathcal{S}_6$-function, we employ a neural network $\widetilde{\mathcal{S}}_6(\mathbf{x})$ to metamodel this selection function. A niched RGA [130] similar to that described in Section 5.1.4 is used here.

A summary of metamodeled functions and the optimization algorithms used to optimize them for all 10 frameworks is provided in Table 5.1. The relative computational cost for each framework can be derived from this table. M3-1 and M3-2 require to construct the maximum number of metamodels among all the frameworks and M6 requires the least, involving only one metamodel. All five generative frameworks (M1-1 to M4-1 and M5) require H independent applications of a single-objective optimization algorithm (RGA) and all simultaneous frameworks (M1-2 to M4-2 and M6) employ an EMO or an multi-modal RGA or a niched RGA once in every epoch.

5.2 Adaptive Switching based Metamodeling (ASM) Frameworks

Each metaomodeling framework in our proposed taxonomy requires to build metamodels for different individual or aggregated objective and constraint functions. Thus, it is expected

Table 5.1: Summary of metamodeled functions and optimization algorithms needed in each epoch for all 10 frameworks.

Frame-work	Metamodeling functions	#Metamodels	Optimization method	#Opt. runs
M1-1	$(\underline{f}_1, \ldots, \underline{f}_M)$ $(\underline{g}_1, \ldots, \underline{g}_J)$	$M + J$	RGA	H
M1-2	Same as above	$M + J$	NSGA-II	1
M2-1	$(\underline{f}_1, \ldots, \underline{f}_M)$ & ACV	$M + 1$	RGA	H
M2-2	Same as above	$M + 1$	NSGA-II	1
M3-1	ASF_{34} & $(\underline{g}_1, \ldots, \underline{g}_J)$	$H + J$	RGA	H
M3-2	Same as above	$H + J$	MM-RGA	1
M4-1	ASF_{34} & ACV	$H + 1$	RGA	H
M4-2	Same as above	$H + 1$	MM-RGA	1
M5	$\mathcal{S}_5$	H	RGA	H
M6	$\mathcal{S}_6$	1	N-RGA	1

that each framework may be most suitable for certain function landscapes that produce a smaller approximation error, but that framework may not fair well in other landscapes. During an optimization process, an algorithm usually faces different kinds of landscape complexities from start to finish. Thus, no one framework is expected to perform best during each step of the optimization process. To determine the best performing framework for a problem, a simple-minded approach would be to apply each of the 10 frameworks to solve each problem independently using $\text{SE}_{\max}$ high-fidelity evaluations, and then determine the specific framework which performs the best using an EMO metric, such as hypervolume or IGD. This will be computationally expensive, requiring 10 times more than the prescribed $\text{SE}_{\max}$. If each framework is allocated only 1/10 of $\text{SE}_{\max}$, they may be insufficient to find comparatively good solutions. A better approach would be use an adaptive switching strategy, in which the most suitable framework is chosen at every step se one such adaptive switching strategy.

We call a 'step' during the optimization process for assessing different metamodeling frameworks to choose the best-performing framework as an *epoch*. In each epoch, exactly H

new in-fill solutions are created irrespective of the metamodeling framework used, thereby consuming H high-fidelity SEs. Clearly, the maximum number of epochs allowable is $E_{\max} = \lceil \frac{SE_{\max} - N_0}{H} \rceil$ with a minor adjustment on the SEs used in the final epoch. At the beginning of each epoch (say, t-th epoch), we have an archive ($\mathcal{A}_t$) of N_t high-fidelity solutions. For the first epoch, these are all N_0 Latin hypercube sampled (LHS) solutions, and in each subsequent epoch, H new in-fill solutions are added to the archive. At the start of t-th epoch, each of the 10 frameworks are used to construct its respective metamodels using all N_t archive members. Then, a 10-fold cross-validation method (described in Section 5.2.2) is used with a suitable performance metric (described in Section 5.2.1) to determine the most suitable framework for the next epoch. A pseudo-code of the proposed ASM approach is provided in Algorithm 10.

5.2.1 Performance Metric for Framework Selection

To compare the performances among multiple surrogate models, mean squared error (MSE) has been widely used in literature [23]. For optimization algorithms, the regression methods that use MSE are known to be susceptible to outliers. For multiple objectives, different objectives and constraints may have different scaling. Our pilot study shows that even with the normalization of the objectives and constraints, the MSE metric does not always correctly evaluate the metamodels. Here, we propose a *selection error probability* (SEP) metric which is appropriate for an optimization task of selecting best performing frameworks more accurately. SEP is defined as the probability of making an error in correctly predicting the better of two solutions compared against each other using the constructed metamodels. Consider Figure 5.1, which illustrates an minimization task and comparison of three different population members pair-wise. The true function values are shown in solid blue, while the predicted

Algorithm 10: Adaptive Swithing Framework (ASM)

Input : Objectives: $[f_1, \ldots, f_m]^T$, Constraints: $[g_1, \ldots, g_J]^T$, n (variables), Initial sample size N_0, SE_{max} (maximum high-fidelity solution evaluations), Switching frameworks $\mathcal{M}_i$ for $i \in \{1 \ldots, S\}$ where S is the number of frameworks, parameters and functions of each framework Γ_i for $i \in \{1 \ldots, s\}$, Number of solutions per epoch u, Number of partitions for cross-validation K

Output: P_T

$t \leftarrow 0$;

$\mathrm{P}_t, \mathrm{F}_t, \mathrm{G}_t \leftarrow \emptyset$;

$\mathrm{P}_{new} \leftarrow \mathrm{LHS}(\rho, n)$// `Initial solutions`

$e \leftarrow |\mathrm{P}_{new}|$;

while *True* **do**

  // `high-fidelity evaluation of objectives`

  $F_{new} = \{f_i(\mathrm{P}_{new}),\ \forall i \in \{1, \ldots, M\}\}$;

  // `high-fidelity evaluation of constraints`

  $G_{new} = \{g_j(\mathrm{P}_{new}),\ \forall j \in \{1, \ldots, J\}\}$;

  // `merge to archive`

  $\mathrm{P}_{t+1}, \mathrm{F}_{t+1}, \mathrm{G}_{t+1} \leftarrow (\mathrm{P}_t \cup \mathrm{P}_{new}), (\mathrm{F}_t \cup \mathrm{F}_{new}), (\mathrm{G}_t \cup \mathrm{G}_{new})$;

  $e \leftarrow e + |\mathrm{P}_{new}|$// `total evaluations`

  break if $e \geq \mathrm{SE}_{max}$// `termination`

  Calculate $\{\mathrm{ASF}(.), \mathrm{ACV}(.),\ \mathcal{S}_5,\ \mathcal{S}_6\}$ etc. from P_{t+1}, F_{t+1} & G_{t+1} as per requirements of $\mathcal{M}_i,\ \forall i$;

  Create random K partition (training and test set) Q^k_{t+1} from $P_{t+1},\ \forall k \in \{1, \ldots, K\}$;

  for *k=1 to K* **do**

    for *i=1 to S* **do**

      $m_i \leftarrow$ Build corresponding metamodels for framework $\mathcal{M}_i$ using training set of Q^k_{t+1};

      $\mathrm{SEP}(k, i) \leftarrow$ Calculate selection-error probability for m_i with test set of Q^k_{t+1};

  $\mathcal{M}_B \leftarrow$ Identify best frameworks from SEP;

  $\mathcal{M}_b \leftarrow$ Randomly choose a framework from $\mathcal{M}_B$;

  $P_{new} \leftarrow$ Optimize framework $\mathcal{M}_b(m_b, \Gamma_b)$;

  if $|P_{t+1}| + |P_{new}| > SE_{max}$ **then**

    $P_{new} \leftarrow$ Randomly pick $\mathrm{SE}_{max} - |P_{t+1}|$ solutions from P_{new};

  $t \leftarrow t + 1$;

  // `end of epoch`

return $\mathrm{P}_T \leftarrow$ filter best solutions from P_{t+1}

function values are shown in dashed blue. When points x_1 and x_2 are compared based on predicted function, the prediction is correct, but when points x_1 and x_3 are compared, the prediction is wrong. Out of three pairwise comparisons, two predictions are correct and one is wrong, thereby making a selection error probability of 1/3 for this case. We argue that in an optimization procedure, it is the SEP which provides a better selection error than the actual function values, as the relative function values are important than the exact function values.

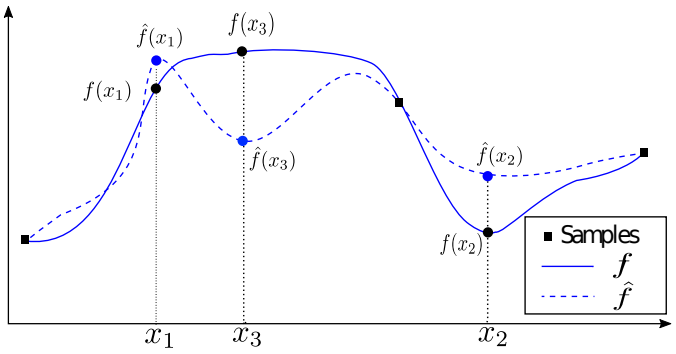

Figure 5.1: Selection Error Probability (SEP) concept is illustrated.

Mathematically, the SEP metric can be defined for n points as follows. For each of $N = \binom{n}{2}$ pairs of points (p and q), evaluate the selection error function ($E(p,q)$), which is one, if there is a mismatch between predicted winner and actual winner of p and q; zero, otherwise. Then, SEP is calculated as follows:

$$\text{SEP} = \frac{1}{N} \sum_{p=1}^{n-1} \sum_{q=p+1}^{n} E(p,q). \tag{5.9}$$

The definition of a 'winner' can be easily extended to multi-objective and constrained multi-objective optimization by considering the domination [7] and constraint-domination [3] status of two points p and q.

5.2.2 Selecting a Framework for an Epoch

A framework which has least SEP value is one of the best frameworks for performing the next epoch. We have performed 10-fold cross-validation in order to identify the best frameworks. After each epoch, H new in-fill points are evaluated using high-fidelity evaluations and added to the archive. In each fold of cross-validation, 90% solutions are used for constructing metamodels with respect to the competing frameworks. Then the corresponding frameworks are used to compare every pair (p and q) of the remaining 10% of archive points using the SEP metric. We apply constrained domination checks to identify the relationship between these two solutions. We then compare this relationship with the true relationship given by their high-fidelity values with the same constrained domination check. We calculate the selection error function ($E(p, q)$). SEP is computed using all pairs of test data. The above process is repeated 10 times by using different blocks of 90% points to obtain 10 different SEP values for each framework. This cross-validation procedure does not require any new solution evaluations, as the whole computations are performed based on the archive points and the predicted values. Thereafter, the best framework is identified based on the median SEP value of frameworks.

Finally, the Wilcoxon rank-sum test is performed between the best framework and all other frameworks. All frameworks within a statistical insignificance (having $p > 0.05$) are identified to obtain the set $\mathcal{M}_B$. Then a randomly chosen framework $\mathcal{M}_b$ is selected from $\mathcal{M}_B$ for the next epoch. Since each of these frameworks performs similarly in a sense of median performance, the choice of a random framework makes the ASM approach diverse with the probability of using different metamodeling landscapes in successive epochs. This procedure, in practice, prohibits the overall approach not to get stuck in similar metamodeling

frameworks for long, even it is one of the best performing frameworks.

5.3 Summary of the Chapter

In this chapter we made a brief discussion on different frameworks proposed in Chapter 2. Thereafter, we have proposed an adaptive switching based metamodeling (ASM) methodology by automatically choosing the most appropriate framework epoch-wise during the course of an optimization run. In order to choose the best framework in every epoch, we perform statistical tests based on a new acceptance criterion – selection error probability (SEP), which counts the correct pairwise relationships of objectives between two test solutions in a k-fold cross-validation test, instead of calculating the mean-squared error of metamodeled objective values from true values. We have observed that SEP is less sensitive to outliers and is much better suited for multi-objective constrained optimization. In each epoch, the ASM approach switches to an appropriate framework which then create a pre-specified number of in-fill points by using either an evolutionary single or multi-objective algorithm or by using a multi-modal or a niche-based real-parameter genetic algorithm.

Chapter 6

Experimental Results

In this chapter, we present the experimental results of the proposed frameworks and adaptive switching method (ASM) on 18 different test problems. In all cases, we have used trust region concept.

6.1 Test Problems

In this chapter we have performed our improved Metamodel-based multi-objective optimization algorithm on different set of test problems with low-budget. Without loss of generality, we assumed that the test problems are expensive in nature thus only a few hundreds (not more than 2000) solution evaluations can be carried out in practice. We have used ZDT test problems [1], DTLZ test problems [85], C2DTLZ2 [129], SRN [96], BNH [131], OSY [132], TNK [78], carside-impact [133] and Welded Beam [134].

6.2 Results And Discussion

We present the results of the ASM approach on 18 different test and engineering problems. The problems include two to five-objective, constrained and unconstrained problems. In order to get robust performance, we have included all 10 frameworks as options for switching in our ASM approach. The performance of ASM approach is compared with each framework

alone. We then compare ASM's performance with three recently suggested multi-objective metamodeling methods: MOEA/D-EGO [24], K-RVEA [19] and CSEA [25].

6.2.1 Parameter Settings

For two-objective problems, we use NSGA-II [37] for M1-2 and M2-2 frameworks. For problems with higher number of objectives, we use NSGA-III [129] procedure. Note that, other multi-objective evolutionary algorithms (e.g. MOEA/D [24] or RVEA [19]) can also be used. A population of size ($N = 100$) is used when the number of reference lines (H) is less than 100. Otherwise, the population size is set identical to H. Initial archive size is set according to Table 6.1. Other parameter settings are as follows: Number of generations $\tau = 300$, SBX crossover probability $p_c = 0.95$, polynomial mutation probability $p_m = 1/n$, distribution indices for SBX and mutation operators are $\eta_c = 20$ and $\eta_m = 20$, respectively. The number of reference points, $\text{SE}_{\max}$, resulting epochs for each problem are presented in Table 6.1.

6.2.2 Two-objective Unconstrained Problems

First, we apply our proposed methodologies to two-objective unconstrained problems: ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6. Table 6.2 presents the median IGD values of 11 runs for each framework applied standalone from start to end. In the absence of any constraint or having a single constraint, M1-1 and M2-1 are identical frameworks; so are M1-2 and M2-2, M3-1 and M4-1, M3-2 and M4-2. This is why we keep a blank under M2-1, M2-2, M4-1, M4-2 entries for unconstrained and single-constraint problems in the table. It is clear from the table that the ASM approach (right-most column) performs better or equivalent to all frameworks for

Table 6.1: Parameter values for 18 problems.

Problem	n	M	J	N_0	$\text{SE}_{\max}$	H	#epochs
ZDT1	10	2	0	100	500	21	20
ZDT2	10	2	0	100	500	21	20
ZDT3	10	2	0	100	500	21	20
ZDT4	5	2	0	100	1000	21	43
ZDT6	10	2	0	100	500	21	20
OSY	6	2	6	200	800	21	29
TNK	2	2	2	200	800	21	29
SRN	2	2	2	200	800	21	29
BNH	2	2	2	200	800	21	29
WB	4	2	4	300	1000	21	39
DTLZ2	7	3	0	500	1000	91	6
C2DTLZ2	7	3	1	700	1500	91	9
CAR	7	3	10	700	2000	91	15
DTLZ5	7	3	0	500	1000	91	6
DTLZ4	7	3	0	700	2000	91	15
DTLZ7	7	3	0	500	1000	91	6
DTLZ2-5	7	5	0	700	2500	210	9
C2DTLZ2-5	7	5	1	700	2500	210	9

all five ZDT problems, whereas M1-1 performs the best in the first four problems. M1-2 and M3-1 performs well in three test problems, whereas M6 performs the best in ZDT6 problem. Obtained non-dominated solutions of two-objective constrained and unconstrained problems of the median run are presented in Figure 6.1. We also show performance of other comparing algorithms: MOEA/D-EGO [24], K-RVEA [19], and CSEA [25] in the figure. It is apparent that ASM approach is able to find a better distributed and converged set of points than other methods for an identical number of SEs.

The epoch-wise proportion of usage of each framework over 11 runs of the ASM approach is shown in Figure 6.2 for all five ZDT problems. For ZDT1, standalone M1-1, M2-1, M3-1 and M4-1 perform in a statistically similar manner as shown in Table 6.2, but the ASM approach mostly restricts its epoch-wise choice on M1-1, M1-2, M2-1 and M2-2 and produces a similar performance. For ZDT2, only M1-1 and M1-2 perform well as a standalone frame-

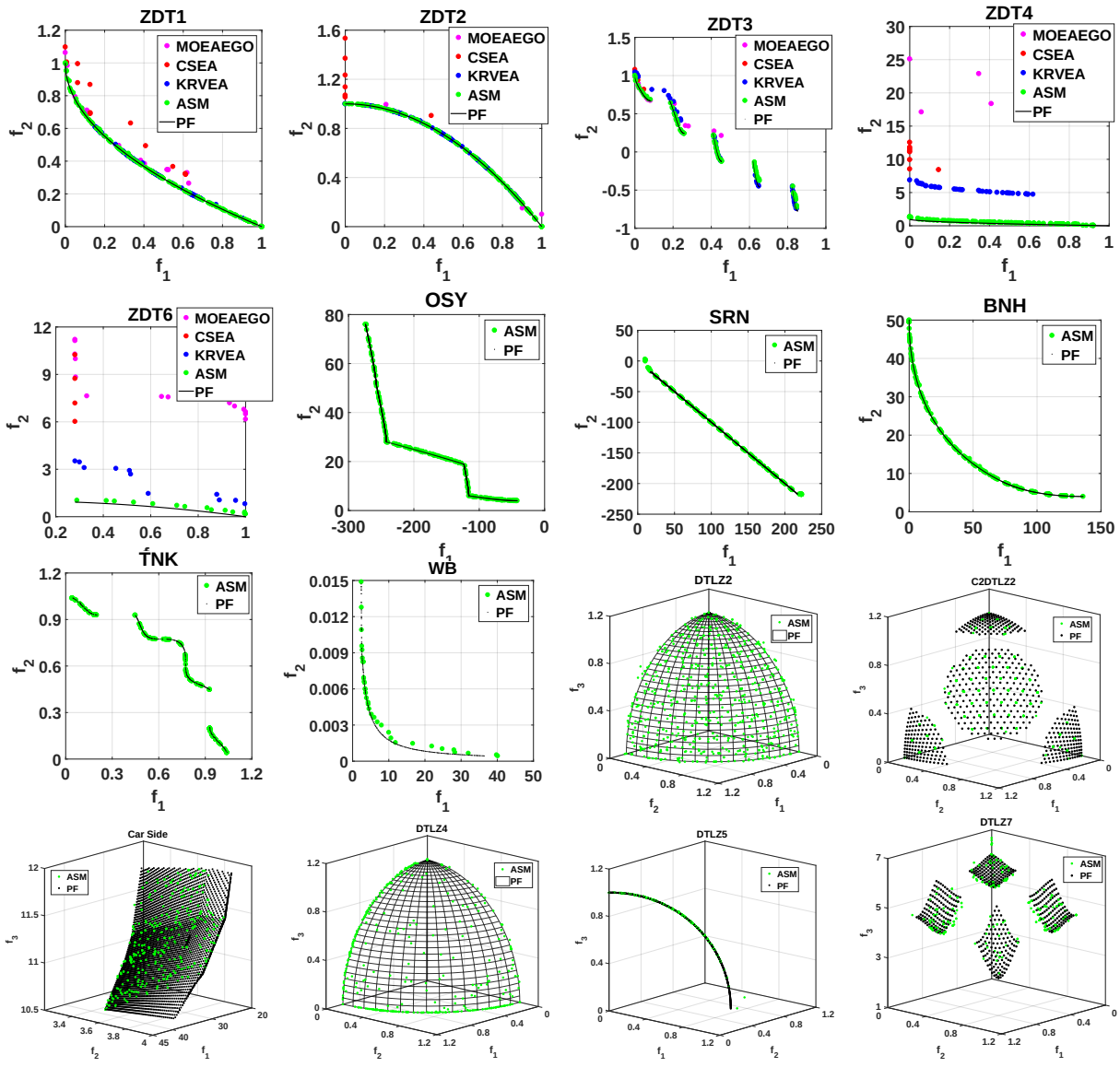

Figure 6.1: Non-dominated solutions of the final archive for the median run of ASM approach for 18 test problems. Algorithms CSEA, K-RVEA and MOEAD-EGO don't handle constrained problems, hence the results are not shown.

work, and ASM approach is able to pick these two frameworks along with M1-2 and M2-2 to produce the best performing result. Except in ZDT6, M1-1, M1-2, M1-2, and M2-2, for which objectives are independently modeled, turn to be dominating frameworks. However, for ZDT6, M3-2, M4-2 and M6 show their dominance. In ZDT4, almost all the frameworks are found to be switching between them early on, but settles with M1 and M2 frameworks at the latter part of the optimization runs. Switching among different frameworks performs well on all five problems. More such results can be found in the supplementary document on other problems.

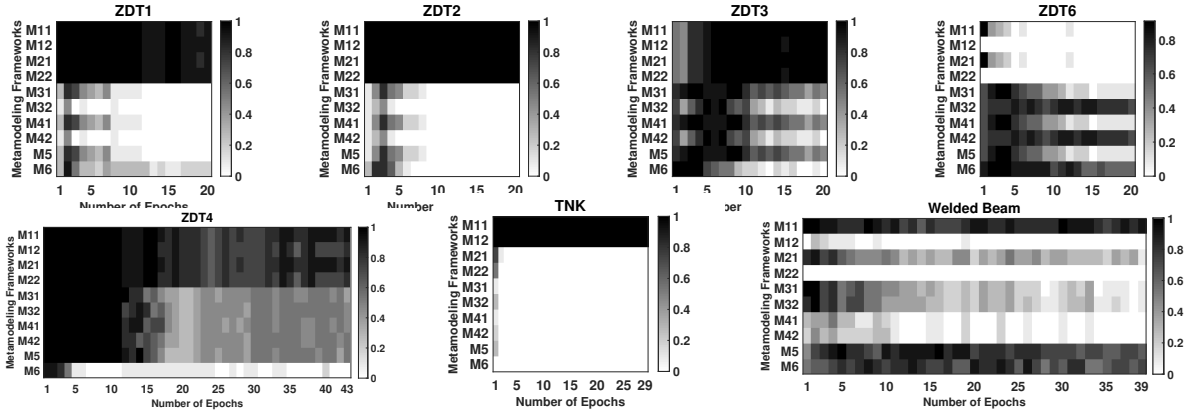

Figure 6.2: Epoch-wise proportion of usage of 10 frameworks in 11 runs of the ASM approach for ZDT problems, TNK, and welded beam design problems.

The switching patterns of frameworks for the median performing run for ZDT1, ZDT4 and ZDT6 are shown in Figure 6.3. For ZDT2, the ASM approach juggles mostly between M1 and M2 variants and produce the best performing result, even better than M1 and M2 alone. In ZDT4, the ASM approach alternates between eight frameworks in the beginning and settles with four of them (M3 and M4 variants) in the middle and then uses M3 variants at the end to produce statistically equivalent result to M1-1 alone. Interestingly, While as a standalone framework from start to end, M1-1 performs the best performance, the ASM approach does not use M1-1 in any of the epochs. The switching of different frameworks

from epoch to epoch is clear from these plots. More plots are provided in the supplementary document.

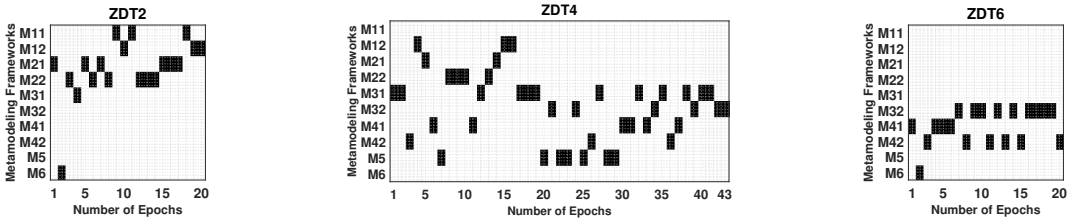

Figure 6.3: Switching among frameworks for the median IGD run of ASM approach for ZDT2, ZDT4 and ZDT6.

6.2.3 Two-objective Constrained Problems

Next, we apply ASM approach and all the frameworks separately to standard two-objective constrained problems: BNH, SRN, TNK, OSY, and the welded beam problem (WB) [3]. The ASM approach performs the best on three of the five problems, followed by M1-1 which performed best in two problems, however both these methods perform the best statistically on all five problems. Other individual frameworks do not perform so well. Figure 6.2 shows the epoch-wise utilization of different frameworks for TNK and WB in 11 runs. The plots for TNK shows that ASM almost always chooses M1-1 or M1-2 as the best-performing frameworks as supported by IGD values in Table 6.2. However, on WB problem, ASM approach selects M1-1, M5 and M6 in most of the epochs, despite poor performance of the latter two when applied in a standalone manner from start to end.

Table 6.2: IGD values obtained from all the individual frameworks and proposed combined switching algorithm for test problems are presented. The best performing framework and other statistically similar frameworks are marked in bold with their p-values in the second row.

Problem	M1-1	M2-1	M1-2	M2-2	M3-1	M4-1	M3-2	M4-2	M5	M6	ASM
ZDT1	**0.00090**	-	***0.00555***	-	***0.00447***	-	***0.00537***	-	-	0.01337	***0.00130***
	-	-	p= 0.4701	-	p= 0.4702	-	p=0.7928	-	-	p=8.1e-5	p=0.091
ZDT2	***0.00065***	-	***0.00062***	-	0.00568	-	0.00910	-	-	0.72366	**0.00055**
	p=0.2372	-	p=0.2372	-	p=8.1e-5	-	p=8.1e-5	-	-	p=8.1e-5	-
ZDT3	***0.00531***	-	**0.00212**	-	0.17123	-	0.19050	-	-	0.08315	***0.00391***
	p=0.325	-	-	-	p=8.1e-5	-	p=8.1e-5	-	-	p=8.1e-5	p=0.369
ZDT4	**0.28900**	-	5.43450	-	***0.29300***	-	0.43450	-	-	6.15510	***0.39992***
	-	-	p=8.1e-5	-	p=0.4307	-	p=0.0126	-	-	p=8.1e-5	p=0.1310
ZDT6	***0.37058***	-	0.48360	-	***0.24192***	-	0.47159	-	-	**0.21327**	***0.24440***
	p=0.2934	-	p=8.1e-5	-	p=0.8438	-	p=0.0013	-	-	-	p= 0.3933
OSY	***0.15323***	24.57940	0.18806	22.99990	6.26550	18.49200	4.77670	18.33760	45.18110	57.15870	**0.12110**
	p= 0.2301	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	-
TNK	**0.00073**	0.04383	***0.00082***	0.02849	0.01180	0.03332	0.01121	0.03743	0.03077	0.03990	***0.00080***
	-	p=8.1e-5	p=0.206	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=0.494
SRN	**0.13191**	4.17160	1.00930	0.92614	1.06120	1.20480	1.51360	1.48870	1.28450	2.41710	***0.13406***
	-	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=0.1891
BNH	***0.07885***	0.74425	***0.04630***	***0.04457***	0.23728	0.23923	0.32874	0.36600	0.23699	0.71300	**0.04176**
	p=0.0865	p=8.1e-5	p=0.5114	p=0.5994	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	-
WB	***0.13794***	0.55529	0.23159	0.84746	***0.16909***	0.88586	1.39250	3.40770	0.96166	1.41110	**0.08960**
	p=0.2933	p=8.1e-5	p=0.0126	p=8.1e-5	p=0.1007	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	-
DTLZ2	0.07870	-	**0.03340**	-	0.05377	-	0.05040	-	-	0.07736	***0.03701***
	p=8.1e-5	-	-	-	p=8.1e-5	-	p=8.1e-5	-	-	p=8.1e-5	p=0.562
C2DTLZ2	0.05130	-	***0.03355***	-	0.03493	-	***0.03190***	-	0.12403	0.04410	**0.03062**
	p=8.1e-5	-	p= 0.115	-	p=0.008	-	p=0.148	-	p=8.1e-5	p=8.1e-5	-
CAR	0.43510	0.43145	0.50119	**0.29817**	0.39809	0.42223	0.40494	0.44251	0.50061	0.55569	0.40110
	p=8.1e-5	p=8.1e-5	p=8.1e-5	-	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=8.1e-5
DTLZ5	0.01960	-	**0.00948**	-	0.01352	-	0.01537	-	-	0.05421	***0.01252***
	p=8.1e-5	-	-	-	p=8.1e-5	-	p=8.1e-5	-	-	p=8.1e-5	p=0.0605
DTLZ4	**0.05840**	-	***0.09024***	-	0.20668	-	0.12570	-	-	***0.08731***	***0.07934***
	-	-	p=0.1203	-	p=8.1e-5	-	p=8.1e-5	-	-	p=0.3933	p=0.425
DTLZ7	0.11808	-	***0.07664***	-	0.87172	-	1.26300	-	-	0.82989	**0.06529**
	p=0.0187	-	p=0.2122	-	p=8.1e-5	-	p=8.1e-5	-	-	p=8.1e-5	-
DTLZ2-5	0.21450	-	**0.03981**	-	0.14401	-	0.14403	-	-	0.11028	***0.04918***
	p=8.1e-5	-	-	-	p=8.1e-5	-	p=8.1e-5	-	-	p=8.1e-5	p=0.595
C2DTLZ2-5	0.17341	-	***0.03676***	-	0.15388	-	0.11669	-	0.29291	0.20842	**0.03441**
	p=8.1e-5	-	p=0.8541	-	p=8.1e-5	-	p=8.1e-5	-	p=8.1e-5	p=8.1e-5	-

6.2.4 Three and More Objective Constrained and Unconstrained Problems

Next, we apply all ten frameworks and ASM approach to three-objective optimization problems (DTLZ2, DTLZ4, DTLZ5 and DTLZ7) and also to two three-objective constrained problem (C2DTLZ2 and the car side impact problem CAR [129]). Table 6.2 shows that while M2-2 works uniquely the best on CAR, M1-2 and M3-2 on C2-DTLZ2, and M1-1, M1-

2 and M6 on DTLZ4, the performance of ASM approach is better or equivalent compared to all 10 problems.

The epoch-wise proportion of utilization of 10 frameworks in 11 runs are shown in Figure 6.4 for three and five-objective problems. It can be clearly seen that M3-1 to M6 frameworks are not usually chosen by the ASM approach on most of these problems, except for complex problems, such as DTLZ4. Switching has been confined between M1-1 to M2-2 for most problems, except in DTLZ4, in which all generative frameworks are found to be useful in certain stages during the optimization process. DTLZ works better with simultaneous frameworks M1-2 and M2-2.

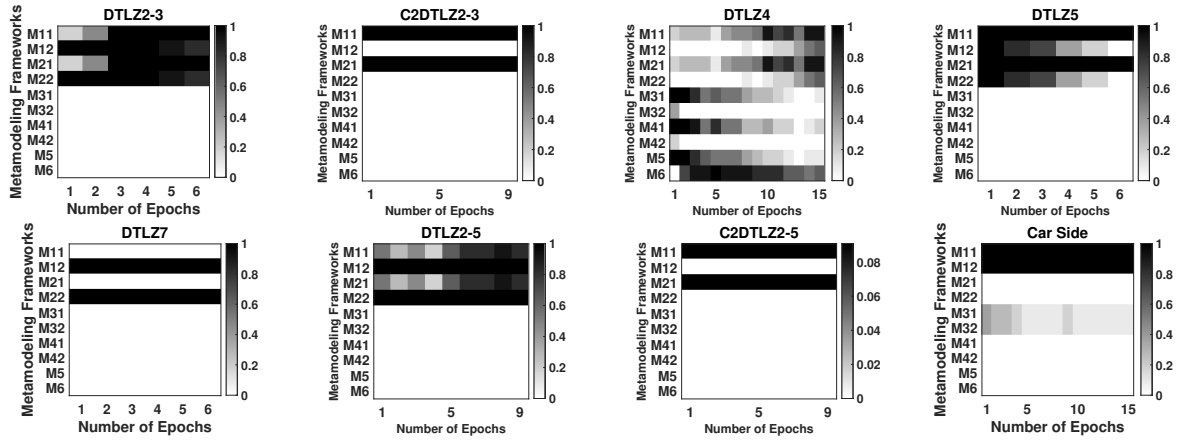

Figure 6.4: Epoch-wise proportion of usage of 10 frameworks in 11 runs of the ASM approach for three and five-objective problems.

On two five-objective unconstrained DTLZ2 and constrained C2-DTLZ2 problems, M1-2 alone and ASM approach perform the best with statistically significant difference with other frameworks. Constrained C2DTLZ2 problems use similar a switching pattern for three and five-objective version of the problem.

Table 6.3 calculates the rank of each of the 10 frameworks in solving 18 problems. The table shows that the ASM approach performs the best overall, followed by M1-2, M2-2 and M3-1 respectively. It indicates that overall, metamodeling of objectives independently is a

better approach for these problems. M6, although being the most efficient in the number of metamodels, performs the worst.

Table 6.3: Average rank of 10 frameworks and the ASM approach on 18 problems based on Wilcoxon rank-sum test.

M1-1	M2-1	M1-2	M2-2	M3-1	M4-1	M3-2	M4-2	M5	M6	ASM
3.66	6.16	2.88	3.00	4.55	5.44	6.22	6.94	6.33	8.55	**1.11**

6.3 Comparative Studies

Next, we examine the performance of adaptive switching metamodeling strategy by comparing them with a few recent algorithms, namely, MOEA/D-EGO [24], K-RVEA [19], and CSEA [25]. Algorithms are implemented in PlatEMO [135]. Since these three competing algorithms can only be applied to unconstrained problems, only ZDT and DTLZ problems are considered here. Identical parameters settings as those used with the ASM approach are used for the three competing algorithms. Table 6.4 presents the mean IGD value of each algorithm. The Wilcoxon rank-sum test results are also shown. It is clearly evident that ASM approach outperforms three competing methods, of which K-RVEA performs well only on two of the nine problems.

6.4 Switching Among Simultaneous Frameworks

We have run our ensemble based algorithm ASM by switching only simultaneous frameworks. The algorithm is denoted by S-ASM. We have obtained the ranks of the algorithm by compare with each individual frameworks as shown in Table 6.5.

Table 6.4: Median IGD on unconstrained problems using ASM approach, and MOEA/D-EGO, K-RVEA, and CSEA algorithms. DNC is denoted as 'Did not converge' within given time.

Problem	MOEA/D-EGO	K-RVEA	CSEA	ASM
ZDT1	0.05611	0.07964	0.95330	**0.00130**
	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=0.0910
ZDT2	0.04922	0.03395	1.01060	**0.00055**
	p=8.1e-5	p=8.1e-5	p=8.1e-5	-
ZDT3	0.30380	0.02481	0.94840	**0.00391**
	p=8.1e-5	p=8.1e-5	p=8.1e-5	-
ZDT4	73.25920	4.33221	12.71600	**0.39992**
	p=8.1e-5	p=8.1e-5	p=8.1e-5	-
ZDT6	0.51472	0.65462	5.42620	**0.24440**
	p=8.1e-5	p=8.1e-5	p=8.1e-5	p= 0.0612
DTLZ2	0.33170	0.0548	0.11420	**0.03701**
	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=0.157
DTLZ4	0.64533	**0.0449**	0.08110	0.07934
	p=8.1e-5	-	p=0.0022	p=0.0380
DTLZ5	0.26203	0.0164	0.03081	**0.01252**
	p=8.1e-5	p=8.1e-5	p=8.1e-5	p=0.211
DTLZ7	5.33220	**0.0531**	0.70520	**0.06529**
	p=8.1e-5	-	p=8.1e-5	p=0.1930
DTLZ2-5	0.31221	0.23031	DNC	**0.04918**
	p=8.1e-5	p=8.1e-5	DNC	-

6.5 Summary of the Chapter

In this chapter we have presented detailed experiments on test the problems using all the frameworks presented in this thesis. We have also presented results of adaptive switching method. On eighteen test and engineering problems having two to five objectives and multiple constraints, the ASM approach has been found to perform much better compared to each framework alone and also to three other existing metamodeling multi-objective algorithms. It has been observed that in most problems a switching between different M1 and M2 frameworks, in which objectives are independently modeled, has performed the best. Metamodeling of constraints in an aggregate manner or independently is not an important matter. However, for more complex problems, such as ZDT3, ZDT6, ZDT4, DTLZ4, and engineering

Table 6.5: Rank of five simultaneous frameworks and S-ASM for 18 problems.

Problem	M1-2	M2-2	M3-2	M4-2	M6	S-ASM
ZDT1	1	1	1	1	6	1
ZDT2	1	1	4	4	6	3
ZDT3	1	1	5	5	3	1
ZDT4	4	4	1	1	6	1
ZDT6	3	3	5	5	1	1
OSY	1	5	3	4	6	1
TNK	1	4	3	5	6	1
SRN	1	1	5	4	6	1
BNH	1	1	4	5	6	1
WB	1	3	4	6	5	1
DTLZ2	1	1	4	4	6	1
C2DTLZ2	1	1	1	1	6	1
CAR	5	1	3	4	6	1
DTLZ5	1	1	4	4	6	1
DTLZ4	1	1	5	5	1	1
DTLZ7	1	1	5	5	4	1
DTLZ2-5	1	1	5	5	4	1
C2DTLZ2-5	1	1	4	4	6	1
Average	1.50	1.78	3.67	4.00	5.05	**1.11**

design problems, all 10 frameworks, including M5 and M6, were involved at different stages of optimization. Interestingly, certain problems have preferred to pick generative frameworks only, while some others have preferred simultaneous frameworks. Clearly, further investigation is needed to decipher a detail problem-wise pattern of selecting frameworks, but this first study on statistics-based adaptive switching has clearly shown its advantage over each framework applied alone.

Chapter 7

Conclusion and Future Work

In this chapter, we make concluding remarks of this thesis and provide some future research directions for solving computationally expensive multi-objective optimization problems.

7.1 Conclusion

In this thesis, we have introduced a taxonomy for metamodel assisted multi-objective optimization algorithms. The taxonomy extends from single-objective optimization problems with and without constraints to multi-objective ones. Constraint handling under the same platform as handling of objectives has been introduce. Outside the popular strategies of handling each objectives and constraints separately, we have introduced few aggregated way by targeting the same set of Pareto-optimal solutions. Under six main categories, we have further classified the framework based on number of targeted optima in each step. This categorization leads to new simultaneous optimization frameworks that have never been proposed in the literature until this thesis proposal. We have performed systematic study of the frameworks by comparing them in a large number of test problems and few real world problems. The results suggest that a particular framework can excel in a particular problem thus each of the framework needs to be studied further. Parameters of all six proposed frameworks have not been fine-tuned for their best performance. Thus a detailed study is needed to understand what class of optimization problems are best suited for different frameworks.

To be particular, we have focused on the simultaneous optimization frameworks which find multiple near-optimal solutions in a single optimization. We have then proposed an adaptive switching based metamodeling (ASM) methodology that chooses the best framework during any stage of the optimization process. To select the best framework, we have proposed a new performance criterion called 'selection error probability' (SEP). Additionally, we have introduced trust region concepts along with performance metrics for good-region adaptation. We have developed faster non-dominated sorting method for efficient search in the model space. We have performed detailed experiments to 18 different test and real world problems. Our results show that we can achieve much better results in terms of convergece and diversity for expensive problems using only a fraction of function evaluations compare to state-of-the-art methods.

7.2 Future Work

In future, we would like to extend our work in the following way.

- Different machine learning models can be suitable for different stages of the search process. Therefore, we would like to extend our work by applying our algorithm to switch among different machine learning models for a particular modeling function or acquisition function.

- There is a good number of acquisition functions that can be used along with different metamodeling frameworks. For example, expected improvement, probability of improvement, hypervolume improvement, and others infill sampling criteria are proposed recently. We can also find the best acquisition functions for a particular class of optimization problems.

- In some problems, objectives and constraints may have heterogeneous computational complexity. We plan to extend our methodologies and applications to these problems as well.

- Trust region concepts have shown promises in solving low-budget multi-objective test problems. Since the number of function evaluations is very low for these problems, sophisticated trust region techniques can provide promising search region thereby helping the algorithm towards better convergence. In future, we would like to investigate non-linear dimensionality reduction techniques to accommodate non-spherical trust regions in high dimensional search space.

- Some theoretical analysis can be performed for simple objectives and constraints under some suitable assumptions to get error bound on the local solutions.

- Theoretical or empirical analysis in terms of number of variables of the problem can be presented in order to demonstrate the curse of dimensionality for modeling a function.

- In future, we would like to show the effectiveness of the uncertainty estimates provided by the Gaussian process.

- The trade-off between CPU time and accuracy among different frameworks and meta-models can be presented in future studies.

- Deep neural network (DNN) can also be used as a metamodel for high-dimensional problems. It would be quite challenging to observe the performance of DNN with such small amount of data.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, Apr 2002.

[3] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.

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

[5] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[6] K. Deb and R. Datta, "A fast and accurate solution of constrained optimization problems using a hybrid bi-objective and penalty function approach," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[7] K. Miettinen, *Nonlinear Multiobjective Optimization*. Kluwer, Boston, 1999.

[8] K. Deb, R. Hussein, P. Roy, and G. Toscano, "Classifying metamodeling methods for evolutionary multi-objective optimization: First results," in *Evolutionary Multi-Criterion Optimization EMO 2017*. Springer, 2017.

[9] K. Deb, R. Hussein, P. C. Roy, and G. Toscano, "A taxonomy for metamodeling frameworks for evolutionary multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2018.

[10] L. Shi and K. Rasheed, "A Survey of Fitness Approximation Methods Applied in Evolutionary Algorithms," in *Computational Intelligence in Expensive Optimization Problems*. Springer Berlin Heidelberg, 2010, vol. 2, pp. 3–28.

[11] A. Forrester, A. Sobester, and A. Keane, *Engineering design via surrogate modelling: A practical guide*. John Wiley & Sons, 2008.

[12] K. Shimoyama, K. Sato, S. Jeong, and S. Obayashi, "Updating kriging surrogate models based on the hypervolume indicator in multi-objective optimization," *Journal of Mechanical Design*, vol. 135, no. 9, p. 094503, 2013.

[13] A. Cassioli and F. Schoen, "Global optimization of expensive black box problems with a known lower bound," *Journal of Global Optimization*, 2013.

[14] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of global optimization*, vol. 21, no. 4, 2001.

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

[16] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with gaussian process model," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 456–474, 2010.

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

[18] F. A. C. Viana, R. T. Haftka, and L. T. Watson, "Efficient global optimization algorithm assisted by multiple surrogate techniques," *Journal of Global Optimization*, 2013.

[19] 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*, vol. 22, no. 1, pp. 129–142, 2018.

[20] D. Zhao and D. Xue, "A multi-surrogate approximation method for metamodeling," *Engineering with Computers*, 2011.

[21] K. Bhattacharjee, H. Singh, and T. Ray, "Multi-objective optimization using an evolutionary algorithm embedded with multiple spatially distributed surrogates," *The American Society of Mechanical Engineers*, vol. 138, no. 9, pp. 135–155, 2016.

[22] H. Wang, Y. Jin, and J. Doherty, "Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2664–2677, Sept 2017.

[23] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen, "A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms," *Soft Computing*, vol. 23, no. 9, pp. 3137–3166, May 2019. [Online]. Available: https://doi.org/10.1007/s00500-017-2965-0

[24] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive Multiobjective Optimization by MOEA/D With Gaussian Process Model," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, june 2010.

[25] L. Pan, C. He, Y. Tian, H. Wang, X. Zhang, and Y. Jin, "A classification based surrogate-assisted evolutionary algorithm for expensive many-objective optimization," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2018.

[26] M. Hüsken, Y. Jin, and B. Sendhoff, "Structure optimization of neural networks for evolutionary design optimization," *Soft Computing*, vol. 9, no. 1, pp. 21–28, Jan 2005.

[27] M. Pilt and R. Neruda, "Improving many-objective optimizers with aggregate meta-models," in *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*, Dec 2011, pp. 555–560.

[28] M. N. Le, Y. S. Ong, Y. Jin, and B. Sendhoff, "A unified framework for symbiosis of evolutionary mechanisms with application to water clusters potential model design," *IEEE Computational Intelligence Magazine*, vol. 7, no. 1, pp. 20–35, Feb 2012.

[29] F. Li, X. Cai, and L. Gao, "Ensemble of surrogates assisted particle swarm optimization of medium scale expensive problems," *Applied Soft Computing*, vol. 74, pp. 291 – 305, 2019.

[30] C. Jin, A. K. Qin, and K. Tang, "Local ensemble surrogate assisted crowding differential evolution," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, May 2015, pp. 433–440.

[31] G. Matheron, "Principles of geostatistics," *Economic Geology*, vol. 58, no. 8, pp. 1246–1266, Dec. 1963.

[32] D. Krige, "A statistical approach to some basic mine valuation problems on the Witwatersrand," *Journal of the Chemical, Metallurgical and Mining Engineering Society of South Africa*, pp. 119–139, 1951.

[33] J. Sacks, W. Welch, T. Mitchell, and H. Wynn, "Design and Analysis of Computer Experiments," *Statistical Science*, pp. 409–423, 1989.

[34] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *J. of Global Optimization*, 1998.

[35] A. P. Wierzbicki, "The use of reference objectives in multiobjective optimization," in *Multiple criteria decision making theory and application*. Springer, 1980, pp. 468–486.

[36] R. E. Steuer, *Multiple Criteria Optimization: Theory, Computation and Applica- tion*. New York: Wiley, 1986.

[37] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II," *IEEE Transactions on Evolutionary Computation*, no. 2, pp. 182–197, April 2002.

[38] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, pp. 577–601, 2014.

[39] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, pp. 712–731, 2007.

[40] P. Rajak, U. Tewary, S. Das, B. Bhattacharya, and N. Chakraborti, "Phases in Zn-coated Fe analyzed through an evolutionary meta-model and multi-objective Genetic Algorithms," *Computational Materials Science*, pp. 2502–2516, 2011.

[41] S. Zapotecas Martínez and C. Coello Coello, "MOEA/D Assisted by RBF Networks for Expensive Multi-objective Optimization Problems," in *GECCO'2013*. ACM.

[42] A. Arias-Montano, C. Coello Coello, and E. Mezura-Montes, "Multi-objective Airfoil Shape Optimization using a Multiple-surrogate Approach," in *IEEE Congress on Evolutionary Computation*, 2012.

[43] A. Turco, "MetaHybrid: Combining Metamodels and Gradient-Based Techniques in a Hybrid Multi-Objective Genetic Algorithm," in *Learning and Intelligent Optimization, 5th International Conference, LION 5*. Rome, Italy: Springer., 2011, pp. 293–307.

[44] S. Zapotecas Martínez and C. A. Coello Coello, "A Multi-Objective Meta-Model Assisted Memetic Algorithm with Non Gradient-Based Local Search," in *GECCO'2010*. Portland, Oregon, USA: ACM Press, 2010, pp. 537–538.

[45] S. Zapotecas Martínez and C. Coello Coello, "A Multi-objective Meta-model Assisted Memetic Algorithm with non Gradient-based Local Search," ser. GECCO '10. ACM, 2010.

[46] N. Namura, K. Shimoyama, and S. Obayashi, "Kriging Surrogate Model Enhanced by Coordinate Transformation of Design Space Based on Eigenvalue Decomposition," in *8th International Conference, EMO 2015*. Guimarães, Portugal: Springer., 2015, pp. 321–335.

[47] P. C. Roy and K. Deb, "High dimensional model representation for solving expensive multi-objective optimization problems," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, July 2016, pp. 2490–2497.

[48] 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*, vol. 30, 2010.

[49] F. Bittner and I. Hahn, "Kriging-Assisted Multi-Objective Particle Swarm Optimization of permanent magnet synchronous machine for hybrid and electric cars," in *2013 IEEE International*, 2013.

[50] J. Beck, D. Friedrich, S. Brandani, and E. S. Fraga, "Multi-objective optimisation using surrogate models for the design of VPSA systems," *Computers & Chemical Engineering*, 2015.

[51] R. Carrese, A. Sobester, H. Winarto, and X. Li, "Swarm Heuristic for Identifying Preferred Solutions in Surrogate-Based Multi-Objective Engineering Design," *AIAA Journal*, pp. 1437–1449, 2011.

[52] A. Lombardi, D. Ferrari, and L. Santos, "Aircraft Air Inlet Design Optimization via Surrogate-Assisted Evolutionary Computation," in *8th International Conference, EMO 2015*. Guimarães, Portugal: Springer., 2015, pp. 313–327.

[53] K. Shimoyama, S. Jeong, and S. Obayashi, "Kriging-Surrogate-Based Optimization Considering Expected Hypervolume Improvement in Non-Constrained Many-Objective Test Problems," in *CEC'2013*. Cancún, México: IEEE Press, 2013, pp. 658–665.

[54] J. Sreekanth and B. Datta, "Multi-objective management of saltwater intrusion in coastal aquifers using genetic programming and modular neural network based surrogate models," *Journal of Hydrology*, pp. 245–256, 2010.

[55] A. Husain, K.-D. Lee, and K.-Y. Kim, "Enhanced multi-objective optimization of a dimpled channel through evolutionary algorithms and multiple surrogate methods," *International Journal for Numerical Methods in Fluids*, pp. 742–759, 2011.

[56] N. Namura, S. Obayashi, and S. Jeong, "Surrogate-Based Multi-Objective Optimization and Data Mining of Vortex Generators on a Transonic Infinite-Wing," in *CEC'2013*. Cancún, México: IEEE Press, 2013, pp. 2910–2917.

[57] S. Kanyakam and S. Bureerat, "Comparative Performance of Surrogate-Assisted MOEAs for Geometrical Design of Pin-Fin Heat Sinks," *Journal of Applied Mathematics*, 2012.

[58] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff, "Generalizing Surrogate-Assisted Evolutionary Computation," *IEEE Transactions on Evolutionary Computation*, pp. 329–355, 2010.

[59] D. Verbeeck, F. Maes, K. D. Grave, and H. Blockeel, "Multi-Objective Optimization with Surrogate Trees," in *GECCO'2013*. New York, USA: ACM Press, 2013, pp. 679–686.

[60] N. Stander, "An Adaptive Surrogate-Assisted Strategy for Multi-Objective Optimization," in *World Congress on Structural and Multidisciplinary Optimization*, Orlando, Florida, 2013.

[61] J. Y. J. Chow and A. C. Regan, "A surrogate-based multiobjective metaheuristic and network degradation simulation model for robust toll pricing," *Optimization and Engineering*, vol. 15, pp. 137–165, 2014.

[62] S. A. Kyriacou, V. G. Asouti, and K. C. Giannakoglou, "Efficient PCA-driven EAs and metamodel-assisted EAs, with applications in turbomachinery," *Engineering Optimization*, vol. 46, pp. 895–911, 2014.

[63] E. Rigoni and A. Turco, "Metamodels for Fast Multi-Objective optimization: Trading Off Global Exploration and Local Exploitation," in *SEAL 2010*. Kanpur, India: Springer. Lecture Notes in Computer Science Vol. 6457, 2010, pp. 523–532.

[64] A. Rosales Pérez, "Surrogate-Assisted Evolutionary Multi-Objective Full Model Selection," Ph.D. dissertation, Instituto Nacional de Astrofísica, Óptica y Electrónica, Tonantzintla, Puebla, México, January 2016.

[65] A. Rosales-Pérez, C. A. Coello Coello, J. A. Gonzalez, C. A. Reyes-García, and H. Jair Escalante, "A Hybrid Surrogate-Based Approach for Evolutionary Multi-Objective Optimization," in *CEC'2013*. Cancún, México: IEEE Press, 2013, pp. 2548–2555.

[66] I. Loshchilov, M. Schoenauer, and M. Sebag, "A Mono Surrogate for Multiobjective Optimization," in *GECCO'2010*. Portland, Oregon, USA: ACM Press, 2010, pp. 471–478.

[67] Y. Zhang, S. Hu, J. Wu, Y. Zhang, and L. Chen, "Multi-objective optimization of double suction centrifugal pump using Kriging metamodels," *Advances in Engineering Software*, pp. 16–26, 2014.

[68] M. N. Le, Y. S. Ong, S. Menzel, C.-W. Seah, and B. Sendhoff, "Multi co-objective evolutionary optimization: cross surrogate augmentation for computationally expensive problems," in *CEC'2012*. Brisbane, Australia: IEEE Press, 2012, pp. 2871–2878.

[69] R. F. Coelho, J. Lebon, and P. Bouillard, "Hierarchical stochastic metamodels based on moving least squares and polynomial chaos expansion," *Structural and Multidisciplinary Optimization*, pp. 707–729, 2011.

[70] R. F. Coelho and P. Bouillard, "Multi-Objective Reliability-Based Optimization with Stochastic Metamodels," *Evolutionary Computation*, vol. 19, pp. 525–560, 2011.

[71] S. Zapotecas Martínez and C. A. Coello Coello, "A Memetic Algorithm with Non Gradient-Based Local Search Assisted by a Meta-Model," in *11th International Conference, Proceedings, Part I*. Kraków, Poland: Springer, 2010, pp. 576–585.

[72] H. K. Singh, T. Ray, and W. Smith, "Surrogate assisted Simulated Annealing (SASA) for constrained multi-objective optimization," in *CEC'2010*. Barcelona, Spain: IEEE Press, 2010, pp. 4202–4208.

[73] J. Martínez-Frutos and D. H. Pérez, "Kriging-based infill sampling criterion for constraint handling in multi-objective optimization," pp. 97–115, 2016.

[74] P. Singh, I. Couckuyt, F. Ferranti, and T. Dhaene, "A Constrained Multi-Objective Surrogate-Based Optimization Algorithm," in *CEC'2014*. Beijing, China: IEEE Press, 2014, pp. 3081–3087.

[75] I. Tsoukalas and C. Makropoulos, "Multiobjective optimisation on a budget: Exploring surrogate modelling for robust multi-reservoir rules generation under hydrological uncertainty," *Environmental Modelling & Software*, 2015.

[76] R. Hussein and K. Deb, "A generative kriging surrogate model for constrained and unconstrained multi-objective optimization," in *GECCO'2016*. ACM Press, 2016.

[77] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM J. on Optimization*, 1998.

[78] M. Tanaka, H. Watanabe, Y. Furukawa, and T. Tanino, "Ga-based decision support system for multicriteria optimization," in *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, vol. 2, Oct 1995, pp. 1556–1561 vol.2.

[79] I. Das and J. Dennis, "Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems," *SIAM J. on Optimization*, 1998.

[80] B. Beachkofski and R. Grandhi, "Improved Distributed Hypercube Sampling," in *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA, 2002*.

[81] K. Deb, R. Hussein, P. Roy, and G. Toscano, "A taxonomy for metamodeling frameworks for evolutionary multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, in press.

[82] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, Apr 2002.

[83] C. C. Tutum and K. Deb, *A Multimodal Approach for Evolutionary Multi-objective Optimization (MEMO): Proof-of-Principle Results*. Cham: Springer International Publishing, 2015, pp. 3–18.

[84] K. Deb, M. Abouhawwash, and J. Dutta, "An optimality theory based proximity measure for evolutionary multi-objective and many-objective optimization," in *EMO'2015*. Springer, pp. 18–33.

[85] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable Test Problems for Evolutionary Multiobjective Optimization," in *Evolutionary Multiobjective Optimization*. Springer Berlin Heidelberg, 2005, pp. 105–145.

[86] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

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

[88] M. T. M. Emmerich, A. H. Deutz, and J. W. Klinkenberg, "Hypervolume-based expected improvement: Monotonicity properties and exact computation," in *2011 IEEE Congress of Evolutionary Computation (CEC)*, 2011, pp. 2147–2154.

[89] 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*, vol. 22, no. 1, pp. 129–142, 2018.

[90] P. C. Roy and K. Deb, "High dimensional model representation for solving expensive multi-objective optimization problems," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 2490–2497.

[91] N. M. Alexandrov, J. E. Dennis, R. M. Lewis, and V. Torczon, "A trust-region framework for managing the use of approximation models in optimization," *Structural optimization*, vol. 15, no. 1, pp. 16–23, 1998.

[92] G. Pedrielli and S. Ng, *G-STAR: A new kriging-based trust region method for global optimization*. United States: Institute of Electrical and Electronics Engineers Inc., 1 2017.

[93] J. hyun Ryu and S. Kim, "A derivative-free trust-region method for biobjective optimization," *SIAM Journal on Optimization*, vol. 24, no. 1, pp. 334–362, 2014.

[94] J. Fliege and A. I. F. Vaz, "A method for constrained multiobjective optimization based on sqp techniques," *SIAM Journal on Optimization*, vol. 26, no. 4, pp. 2091–2119, 2016.

[95] L. While, L. Bradstreet, and L. Barone, "A fast way of calculating exact hypervolumes," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 86–95, 2012.

[96] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, Sep. 1994.

[97] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," in *Evolutionary Methods for*

Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), K. Giannakoglou *et al.*, Eds. International Center for Numerical Methods in Engineering (CIMNE), 2002, pp. 95–100.

[98] J. Knowles and D. Corne, "The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 1, 1999, p. 105 Vol. 1.

[99] D. Corne, J. D. Knowles, and M. J. Oates, "The pareto envelope-based selection algorithm for multi-objective optimisation," in *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, ser. PPSN VI. London, UK, UK: Springer-Verlag, 2000, pp. 839–848.

[100] P. Roy, M. Islam, K. Murase, and X. Yao, "Evolutionary path control strategy for solving many-objective optimization problem," *Cybernetics, IEEE Transactions on*, vol. 45, no. 4, pp. 702–715, April 2015.

[101] X. Zhang, Y. Tian, and Y. Jin, "A knee point-driven evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 6, pp. 761–776, Dec 2015.

[102] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 694–716, Oct 2015.

[103] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, Oct 2016.

[104] A. O. Hero and G. Fleury, "Pareto-optimal methods for gene ranking," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 38, no. 3, pp. 259–275, 2004.

[105] J. Handl and J. Knowles, "Exploiting the trade-off — the benefits of multiple objectives in data clustering," in *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, ser. EMO'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 547–560.

[106] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *Data Engineering, 2001. Proceedings. 17th International Conference on*, April 2001, pp. 421–430.

[107] K. Leyton-Brown and Y. Shoham, *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*, 1st ed. Morgan and Claypool Publishers, 2008.

[108] D. Romik, *The Surprising Mathematics of Longest Increasing Subsequences*, 1st ed. Cambridge University Press, 2015.

[109] F. F. Yao, "On Finding the maximal elements in a set of plane vectors," Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana Champaign, Illinois, Tech. Rep. 667, 1974.

[110] F. Nielsen, "Output-sensitive peeling of convex and maximal layers," *Information Processing Letters*, vol. 59, no. 5, pp. 255 – 259, 1996.

[111] S. Tang, Z. Cai, and J. Zheng, "A fast method of constructing the non-dominated set: Arena's principle," in *Proceedings of the 2008 Fourth International Conference on Natural Computation - Volume 01*, ser. ICNC '08. IEEE Computer Society, 2008, pp. 391–395.

[112] K. McClymont and E. Keedwell, "Deductive sort and climbing sort: New methods for non-dominated sorting," *Evol. Comput.*, vol. 20, no. 1, pp. 1–26, Mar. 2012.

[113] H. Wang and X. Yao, "Corner sort for pareto-based many-objective optimization," *Cybernetics, IEEE Transactions on*, vol. 44, no. 1, pp. 92–102, Jan 2014.

[114] H. Fang, Q. Wang, Y.-C. Tu, and M. F. Horstemeyer, "An efficient non-dominated sorting method for evolutionary algorithms," *Evol. Comput.*, vol. 16, no. 3, pp. 355–384, Sep. 2008.

[115] M. Jensen, "Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 5, pp. 503–515, Oct 2003.

[116] F.-A. Fortin, S. Grenier, and M. Parizeau, "Generalizing the improved run-time complexity algorithm for non-dominated sorting," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '13. New York, NY, USA: ACM, 2013, pp. 615–622.

[117] "A provably asymptotically fast version of the generalized Jensen algorithm for non-dominated sorting," in *Parallel Problem Solving from Nature - PPSN XIII*, ser. Lecture Notes in Computer Science, T. Bartz-Beielstein, J. Branke, B. Filipic, and J. Smith, Eds. Springer International Publishing, 2014, vol. 8672.

[118] J. L. Bentley, "Multidimensional divide-and-conquer," *Commun. ACM*, vol. 23, no. 4, pp. 214–229, Apr. 1980.

[119] J. L. Bentley, K. L. Clarkson, and D. B. Levine, "Fast linear expected-time alogorithms for computing maxima and convex hulls," in *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '90. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1990, pp. 179–187.

[120] H. T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *J. ACM*, vol. 22, no. 4, pp. 469–476, Oct. 1975.

[121] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "An efficient approach to nondominated sorting for evolutionary multiobjective optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 19, no. 2, pp. 201–213, April 2015.

[122] P. Gustavsson and A. Syberfeldt, "A new algorithm using the non-dominated tree to improve non-dominated sorting," *Evolutionary Computation*, vol. 26, no. 1, pp. 89–116, 2017.

[123] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 97–112, Feb 2018.

[124] A. Jaszkiewicz and T. Lust, "Nd-tree-based update: a fast algorithm for the dynamic non-dominance problem," *IEEE Transactions on Evolutionary Computation*, vol. PP, no. 99, pp. 1–1, 2018.

[125] K. Li, K. Deb, Q. Zhang, and Q. Zhang, "Efficient nondomination level update method for steady-state evolutionary multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2838–2849, Sept 2017.

[126] M. Drozdk, Y. Akimoto, H. Aguirre, and K. Tanaka, "Computational cost reduction of nondominated sorting using the m-front," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 659–678, Oct 2015.

[127] S. Huband, P. Hingston, L. Barone, and L. While, "A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit," *Evolutionary Computation, IEEE Transactions on*, vol. 10, pp. 477–506, 2006.

[128] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, 2000.

[129] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 4, pp. 577–601, Aug 2014.

[130] P. Roy, R. Hussein, and K. Deb, "Metamodeling for multimodal selection functions in evolutionary multi-objective optimization," in *GECCO'2017*. ACM Press, 2017.

[131] T. T. Binh and U. Korn, "Mobes: A multiobjective evolution strategy for constrained optimization problems," in *In Proceedings of the Third International Conference on Genetic Algorithms (MENDEL97*, 1997, pp. 176–182.

[132] A. Osyczka and S. Kundu, "A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm," *Structural optimization*, vol. 10, no. 2, pp. 94–99, Oct 1995. [Online]. Available: https://doi.org/10.1007/BF01743536

[133] 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*, vol. 18, no. 4, pp. 602–622, Aug 2014.

[134] K. Deb, A. Pratap, and S. Moitra, "Mechanical component design for multiple ojectives using elitist non-dominated sorting ga," in *Parallel Problem Solving from Nature PPSN VI*, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 859–868.

[135] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "Platemo: A MATLAB platform for evolutionary multi-objective optimization," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017.